11th Innovations in Theoretical Computer Science Conference

ITCS 2020, January 12–14, 2020, Seattle, Washington, USA

Edited by

Thomas Vidick

LIPIcs – Vol. 151 – ITCS 2020 www.dagstuhl.de/lipics

Editors

Thomas Vidick
California Institute of Technology, Pasadena, CA, USA
vidick@caltech.edu

ACM Classification 2012
Mathematics of computing; Theory of computation

ISBN 978-3-95977-134-4

Published online and open access by
Schloss Dagstuhl – Leibniz-Zentrum für Informatik GmbH, Dagstuhl Publishing, Saarbrücken/Wadern, Germany. Online available at https://www.dagstuhl.de/dagpub/978-3-95977-134-4.

Publication date
January, 2020

Bibliographic information published by the Deutsche Nationalbibliothek
The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available in the Internet at https://portal.dnb.de.

Digital Object Identifier: 10.4230/LIPIcs.ITCS.2020.0

LIPIcs – Leibniz International Proceedings in Informatics

LIPIcs is a series of high-quality conference proceedings across all fields in informatics. LIPIcs volumes are published according to the principle of Open Access, i.e., they are available online and free of charge.

ISSN 1868-8969

https://www.dagstuhl.de/lipics

Contents

Regular Papers

11th Innovations in Theoretical Computer Science Conference (ITCS 2020).
Editor: Thomas Vidick
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Preface

The papers in this volume were presented at the 11th Innovations in Theoretical Computer Science (ITCS 2020) conference. The conference was held at the University of Washington in Seattle, WA, USA, January 12-14, 2020. ITCS seeks to promote research that carries a strong conceptual message, for instance, introducing a new concept or model, opening a new line of inquiry within traditional or cross-interdisciplinary areas,introducing new techniques, or making novel connections between existing areas and ideas. The conference format is single-session and aims to promote the exchange of ideas between different areas of theoretical computer science and with other disciplines. The call for papers welcomed all submissions, whether aligned with current theory of computation research directions or deviating from them. A record 204 submissions were received. Of these, the program committee selected 86 papers. I would like to thank the authors of all submissions, whether accepted or not, for their interest in ITCS.

The program committee consisted of 32 members (plus the chair): Nikhil Bansal, CWI + TU Eindhoven; Nir Bitansky, Tel-Aviv University; Clement Canonne, Stanford; Timothy Chan, University of Illinois at Urbana-Champaign; Edith Cohen, Google and Tel-Aviv University; Shaddin Dughmi, University of Southern California; Sumegha Garg, Princeton; Ankit Garg, Microsoft research; Ran Gelles, Bar-Ilan University; Elena Grigorescu, Purdue; Tom Gur, University of Warwick; Sandy Irani, UC Irvine; Dakshita Khurana, University of Illinois at Urbana-Champaign; Antonina Kolokolova, Memorial University of Newfoundland.; Pravesh Kothari, Carnegie Mellon University; Rasmus Kyng, Harvard; Katrina Ligett, Hebrew University; Nutan Limaye, IIT Bombay; Pasin Manurangsi, UC Berkeley; Tamara Mchedlidze, Karlsruhe Institute of Technology; Dana Moshkovitz, UT Austin; Jelani Nelson, UC Berkeley; Merav Parter, Weizmann Institute; Krzysztof Pietrzak, IST Austria; Elaine Shi, Cornell; Piyush Srivastava, Tata Institute of Fundamental Research, Mumbai; Li-Yang Tan, Stanford; Madhur Tulsiani, TTIC; Gregory Valiant, Stanford; Virginia Vassilevska Williams, MIT; Ronald de Wolf, CWI and University of Amsterdam; David Woodruff, Carnegie Mellon University.

I wish to express my heartfelt thanks to them for agreeing to join the committee as well as for investing a great deal of time and effort to evaluate the submissions. I am also grateful to the many subreviewers who assisted with the reviewing process. The local organizer was Shayan Oveis Gharan from University of Washington. I would like to thank him very much for his service. I'm also grateful to Umesh Vazirani, chair of the ITCS Steering Committee, and the entire Steering Committee for their help and guidance in shaping the format of the conference. Finally, I would like to thank all the presenters and the audience at ITCS for making ITCS a wonderful experience.

Thomas Vidick
ITCS 2020 Program Chair
California Institute of Technology
Pasadena, CA, USA

11th Innovations in Theoretical Computer Science Conference (ITCS 2020).
Editor: Thomas Vidick

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

ITCS 2020 Conference Organization

Program Chair: Thomas Vidick (Caltech)

Local Organization: Shayan Oveis Gharan (University of Washington)

Steering Committee Chair: Umesh Vazirani (UC Berkeley)

Steering Committee
Sanjeev Arora, Princeton
Manuel Blum, Carnegie Mellon
Bernard Chazelle, Princeton
Irit Dinur, Weizmann
Oded Goldreich, Weizmann
Shafi Goldwasser, MIT and Weizmann
Richard Karp, Berkeley
Robert Kleinberg, Cornell University
Ueli Maurer, ETH
Silvio Micali, MIT
Christos Papadimitriou, Berkeley
Michael Rabin, Harvard
Omer Reingold, Stanford
Tim Roughgarden, Stanford
Madhu Sudan, Harvard
Leslie Valiant, Harvard
Umesh Vazirani, Berkeley
Thomas Vidick, Caltech
Avi Wigderson, IAS
Andy Yao, Tsinghua

Program Committee:
Nikhil Bansal, CWI + TU Eindhoven
Nir Bitansky, Tel-Aviv University
Clement Canonne, Stanford
Timothy Chan, University of Illinois at Urbana-Champaign
Edith Cohen, Google and Tel-Aviv University
Shaddin Dughmi, University of Southern California
Sumegha Garg, Princeton
Ankit Garg, Microsoft research
Ran Gelles, Bar-Ilan University
Elena Grigorescu, Purdue
Tom Gur, University of Warwick
Sandy Irani, UC Irvine
Dakshita Khurana, University of Illinois at Urbana-Champaign
Antonina Kolokolova, Memorial University of Newfoundland.
Pravesh Kothari, Carnegie Mellon University
Rasmus Kyng, Harvard
Katrina Ligett, Hebrew University
Nutan Limaye, IIT Bombay
Pasin Manurangsi, UC Berkeley

11th Innovations in Theoretical Computer Science Conference (ITCS 2020).
Editor: Thomas Vidick
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Program Committee (continued):

Tamara Mchedlidze, Karlsruhe Institute of Technology
Dana Moshkovitz, UT Austin
Jelani Nelson, UC Berkeley
Merav Parter, Weizmann Institute
Krzysztof Pietrzak, IST Austria
Elaine Shi, Cornell
Piyush Srivastava, Tata Institute of Fundamental Research, Mumbai
Li-Yang Tan, Stanford
Madhur Tulsiani, TTIC
Gregory Valiant, Stanford
Virginia Vassilevska Williams, MIT
Ronald de Wolf, CWI and University of Amsterdam
David Woodruff, Carnegie Mellon University

Additional Reviewers:

Geoffroy Couteau	Muhammad Haris Mughees	Anat Paskin-Cherniavsky
Anthony Leverrier	Matthew Hastings	Arturs Backurs
Amir Abboud	Lijie Chen	Alexander Block
Igor Shinkar	Prahladh Harsha	Warut Suksompong
Christian Stricker	Bill Fefferman	Shalev Ben-David
Ben Reichardt	Wei Zhan	Daniel Dadush
Karthik C. S.	Dylan Quintana	Peter Manohar
Young-San Lin	Brendan Lucier	Vincent Cohen-Addad
Chethan Kamath	S. Matthew Weinberg	Zhiyi Huang
Justin Thaler	Gillat Kol	Sahil Singla
Ashish Chiplunkar	David Eppstein	Rafael Oliveira
Sushant Sachdeva	Brian Bullins	Manish Raghavan
Alessandro Chiesa	Ashish Chiplunkar	Christopher Jung
Jeroen Zuiddam	Josh Alman	Raghunath Tewari
Orr Fischer	Amir Abboud	Michael Chapman
Prahladh Harsha	Suprovat Ghoshal	Thatchaphol Saranurak
Shay Solomon	Matheus V. X. Ferreira	Artem Tsikiridis
Aaron Roth	Minshen Zhu	Hessam Mahdavifar
Ariel Schvartzman	Guus Regts	Jacob Focke
Spencer Gordon	Rahul Savani	Chethan Kamath
Mika Göös	Noah Fleming	Sushant Sachdeva
Hariharan Narayanan	Maris Ozols	Andreas Winter
Matthew Weinberg	Rajesh Chitnis	Hossein Esfandiari
Vatsal Sharan	Kate Donahue	Gal Yona
Ofir Geri	Yakov Babichenko	Stephen Fenner
Viswanath Nagarajan	Ola Svensson	Greg Kuperberg
Dor Minzer	Pooya Hatami	D. Ellis Hershkowitz
Kai-Min Chung	Arnab Bhattacharyya	Michael Joswig
Stephane Gaubert	Srinivasan Arunachalam	Alexander Belov
Pc Discussion	Karen Klein	Nick Spooner

Additional Reviewers (continued):

Mika Goos
Pekka Orponen
Biaoshuai Tao
Uriel Feige
Prashant Vasudevan
Tyler Helmuth
Abhradeep Guha Thakurta
Preetum Nakkiran
Przemysław Uznański
Omri Weinstein
Yuichiro Kamada
Subhonmesh Bose
Stefano Tessaro
Marcelo Ariel Fernandez
Talya Eden
Gregory Gutin
Nathan Manohar
Eshan Chattopadhyay
Chandra Chekuri
Goran Zuzic
Kai-Min Chung
Kshipra Bhawalkar
Anindya De
Shachar Lovett
Roi Livni
Giulio Malavolta
Arnaud Labourel
Christian Ikenmeyer
Gilad Asharov
Brett Hemenway
Venkata Gandikota
Joachim Spoerhase
Taisuke Yasuda
Nicolas Resch
Rob van Stee
Philipp Kindermann
Jouke Witteveen
Ilan Komargodski
Yuval Filmus
Paul Duetting
Onkar Bhardwaj
Kv Subrahmanyam
Thibaut Horel
Tom van der Zanden
Sanket Kanjalkar
Yin Tat Lee

Anupam Das
Or Sheffet
Hung Nguyen
Shay Mozes
Aram Harrow
Dana Randall
Gal Yona
Badih Ghazi
Rahul Santhanam
Vaggos Chatziafratis
Kavitha Telikepalli
Aaron Schild
Vladimir Podolskii
Nicole Wein
Yury Makarychev
Eunjung Kim
Feng-Hao Liu
Lijie Chen
Inbal Talgam-Cohen
Benjamin Niedermann
Noah Golowich
Nima H.
Sitan Chen
Kshipra Bhawalkar
Viswanath Nagarajan
Alex Lombardi
Cole Franks
Bruno Loff
Akshayaram Srinivasan
Carmine Ventre
Yoram Moses
Chen Dan
Arnold Filtser
Arnab Bhattacharyya
Meirav Zehavi
Rohit Gurjar
Oxana Poburinnaya
Karthik C. S.
Florian Speelman
Umang Bhaskar
Edith Elkind
Sepehr Assadi
John Peebles
Michael Walter
Vassilis Zikas
Huy Nguyen

Lisa Yang
Pc Discussion
Huy Nguyen
Wei-Kai Lin
Uma Girish
Vitaly Feldman
Michael P. Kim
Joshua Brody
Avishay Tal
Josh Alman
Christian Konrad
Jingcheng Liu
Manish Raghavan
Marc Roth
Konstantin Makarychev
John Wilmes
Stefan Dziembowski
Ryan Williams
Marc Vinyals
Thomas Steinke
Rahul Jain
Meng-Tsung Tsai
Srikanth Srinivasan
Steven Wu
Lisa Hellerstein
Sariel Har-Peled
Abhiram Natarajan
Michael A. Forbes
Saikrishna Badrinarayanan
Irit Dinur
Viswanath Nagarajan
Zachary Friggstad
Di Wang
Manish Purohit
Ran Cohen
Michael P. Kim
Karen Klein
Omer Paneth
Shalev Ben-David
Anilesh Kollagunta Krishnaswamy
Zeyu Guo
Debmalya Panigrahi
Moses Charikar
Jacob Leshno
Yassine Hamoudi
Carsten Baum

Additional Reviewers (continued):

Amit Agarwal
Rajmohan Rajaraman
Ankit Singh Rawat
Nika Haghtalab
Varun Kanade
Urmila Mahadev
Ofir Geri
Marco Carmosino
Ola Svensson
Ke Wu
Kuan-Yi Ho
Marshall Ball
Mary Wootters
Tuomas Hakoniemi
Jong Chan Lee
Ron Rosenthal
Mariana Raykova
Vedat Alev
Noah Golowich
Viet Tung Hoang
Ami Paz
Tami Tamir
Xiaorui Sun
Andrea Lincoln
Anupam Prakash
Vitaly Feldman
Ido Shahaf
Gilad Asharov
Govind Ramnarayan
Karl Bringmann
Oded Lachish
Venkatesan Guruswami
Ola Svensson
Yuval Dagan
Yupeng Zhang
Brendan Lucier
Hila Dahari
Stefano Tessaro
Marco Carmosino
Brynmor Chapman
Antigoni Polychroniadou
Michael Walter

Guy Kortsarz
Sivakanth Gopi
Bo Waggoner
Zhenjian Lu
Zvika Brakerski
Mrinal Kumar
Boi Faltings
Sahil Singla
Omri Weinstein
Ilan Komargodski
Joel Alwen
Antigoni Polychroniadou
Atri Rudra
Sasank Mouli
Heiko Röglin
Irit Dinur
Qiang Tang
Ron Rosenthal
Cameron Musco
Rio LaVigne
Sidhanth Mohanty
Samson Zhou
Constantine Caramanis
Carlos Santos
Suhail Sherif
Akshayaram Srinivasan
Andy Drucker
Huijia Lin
Rishiraj Bhattacharyya
Arturs Backurs
Themis Gouleakis
Bo Waggoner
Marek Adamczyk
Riad Wahby
Rasmus Pagh
Qiang Tang
Tianren Liu
Sebastian Faust
Igor Olivera
Debayan Gupta
Jop Briët
Omer Paneth

Shubhang Kulkarni
Mary Wootters
Pathikrit Basu
Michael Hamann
Stacey Jeffery
Marco Carmosino
Yang Liu
Marek Adamczyk
Aviad Rubinstein
Akshayaram Srinivasan
Fermi Ma
Nicolas Resch
Iordanis Kerenidis
Rahul Santhanam
Yuval Filmus
Aarushi Goel
Rajesh Jayaram
Tali Kaufman
Bruce Kapron
Greg Bodwin
Goutham Rajendran
Rahul Santhanam
David Eppstein
Srinivasan Arunachalam
Badih Ghazi
Eshan Chattopadhyay
Yilei Chen
Jeremy Fineman
John Miller
Talya Eden
Sivakanth Gopi
Zhiyi Huang
Ainesh Bakshi
Justin Thaler
Ilya Razenshteyn
Peter Gazi
Bruce Kapron
Prabhanjan Ananth
Yael Hitron
Rotem Tsabary
Sivakanth Gopi

Hardness Amplification of Optimization Problems

Elazar Goldenberg
The Academic College of Tel Aviv Yaffo, Israel
elazargo@mta.ac.il

Karthik C. S.
Tel Aviv University, Israel
karthiks@mail.tau.ac.il

Abstract

In this paper, we prove a general hardness amplification scheme for optimization problems based on the technique of direct products.

We say that an optimization problem Π is direct product feasible if it is possible to efficiently aggregate any k instances of Π and form one large instance of Π such that given an optimal feasible solution to the larger instance, we can efficiently find optimal feasible solutions to all the k smaller instances. Given a direct product feasible optimization problem Π, our hardness amplification theorem may be informally stated as follows:

If there is a distribution $\mathcal{D}$ over instances of Π of size n such that
every randomized algorithm running in time $t(n)$ fails to solve
Π on $\frac{1}{\alpha(n)}$ fraction of inputs sampled from $\mathcal{D}$,
then, assuming some relationships on $\alpha(n)$ and $t(n)$,
there is a distribution $\mathcal{D}'$ over instances of Π of size $O(n \cdot \alpha(n))$ such that
every randomized algorithm running in time $\frac{t(n)}{poly(\alpha(n))}$ fails to solve
Π on $99/100$ fraction of inputs sampled from $\mathcal{D}'$.

As a consequence of the above theorem, we show hardness amplification of problems in various classes such as NP-hard problems like Max-Clique, Knapsack, and Max-SAT, problems in P such as Longest Common Subsequence, Edit Distance, Matrix Multiplication, and even problems in TFNP such as Factoring and computing Nash equilibrium.

2012 ACM Subject Classification Theory of computation → Computational complexity and cryptography

Keywords and phrases hardness amplification, average case complexity, direct product, optimization problems, fine-grained complexity, TFNP

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.1

Related Version A full version of the paper is available at `http://arxiv.org/abs/1908.10248`.

Funding *Karthik C. S.*: Most of this work was done while the author was a student at Weizmann Institute of Science and was supported by Irit Dinur's ERC-CoG grant 772839. This work also received support from the Israel Science Foundation (grant number 552/16) and the Len Blavatnik and the Blavatnik Family foundation.

Acknowledgements We would like to thank Amir Abboud, Irit Dinur, and Eylon Yogev for discussions and comments.

1 Introduction

The widely believed conjecture $\mathsf{P} \neq \mathsf{NP}$ asserts that the class NP cannot be decided efficiently on the worst-case. That is, no polynomial time algorithm can decide the satisfiability of a CNF formula on *every* instance. However, the worst case hardness of NP still does not clarify its average-case hardness: how hard is to decide the satisfiability on a uniformly random instance.

Studying the average-case hardness of NP has a two-fold motivation. First, it may provide a more meaningful explanation than worst-case complexity about the intractability of NP-hard instances actually encountered in practice. In other words, if NP is hard only on the worst-case, then the theory of worst-case complexity that has been extensively developed over the last fifty years, might not be a good reflection of reality. Second, hardness on average is the cornerstone of modern cryptography as the security of any nontrivial cryptosystem requires some computational problem to be average-case hard (for some nice distribution). Additionally, showing average-case hardness for functions is a stepping stone towards proving strong derandomization results and the construction of pseudorandom generators.

The study of hardness amplification is the task of connecting the worst-case and average-case hardness. More specifically, based on a worst-case hardness (assumption) one would like to prove the average-case hardness of the problem.

1.1 Utopic Theorem of Hardness Amplification

A utopic theorem in the context of hardness amplification would assert that if a function is hard in the worst-case then it implies the average-case hardness for the same function against algorithms with essentially the same running time complexity. More formally it would look as follows:

▶ **Utopic Theorem 1** (A Utopic Hardness Amplification Theorem). *Let $\{f_n\}_{n\in\mathbb{N}}$ be a family of functions. Assume that every algorithm running in time $t(n)$, fails to compute f_n on at least $\gamma(n)$ fraction of inputs. Then there exists a family $\{g_n\}_{n\in\mathbb{N}}$ of functions, such that every algorithm running in time $t'(n)$, fails to compute g_n on at least $\gamma'(n)$ fraction of inputs.*

Ideally, we would like to achieve the above amplification for the following parameters[1]

1. $\gamma(n) = O(1/2^n)$ *and* $\gamma'(n) = 1/2 - O(1/2^n)$,
2. $t'(n) \approx t(n)$,
3. $\{f_n\}_{n\in\mathbb{N}} = \{g_n\}_{n\in\mathbb{N}}$.

We briefly elaborate here why we would like the above three setting of parameters in our utopic hardness amplification theorem. Item 1 would yield a worst-case to average-case reduction, and therefore extend all the lower bounds and hardness results that have been achieved in the theory of worst-case complexity for f to the average-case complexity of g. In fact, achieving $\gamma'(n) = 1/2 - O(1/2^n)$ would imply that no algorithm running in time $t'(n)$ can do much better than randomly guessing the output. Item 2 would imply that our worst-case complexity lower bounds meaningfully translate to lower bounds in the average-case. Item 3 expresses the notion of self-reducibility: if we are interested in understanding the average complexity of a problem, our hardness amplification theorem should enable us to do so by analyzing the worst-case complexity of the *same* problem. In summary, obtaining a hardness amplification result satisfying the three items is in a sense an attempt to bridge the gap between theory and practice. Finally, we remark that our utopic theorems would gain more importance if the family of functions for which we show hardness amplification are natural (in some broad sense).

Specifically, if we prove such a theorem for the family of deciding satisfiability of CNF formulas, then we get that the assumption that P $\neq$ NP implies that every polynomial time algorithm fails to decide satisfiability on slightly more that half of the CNF formulas –

[1] In order to succinctly specify the desirable parameters of a hardness amplification theorem, we assume here that f_n and g_n are Boolean functions.

a highly non-trivial and very desirable result that would pave the way for the construction of one-way functions from (weak) worst-case assumptions. However, as we wake up from the dream of a utopia, one may wonder if such a result can even be achieved [7].

Remarkably, nearly three decades ago, Lipton [26, 9] proved the above type of (utopic) theorem for the function of computing the permanent (a #P-complete problem) against probabilistic polynomial time algorithms. Trevisan and Vadhan [32] following a line of works [3, 24, 29] proved such an amplification result for PSPACE and almost proved such an amplification result for the class EXP (they couldn't achieve Item 3). For the class NP we are far from proving a strong hardness amplification result, and there are some known barriers while trying to convert worst-case NP-hardness into average-case hardness (see e.g. [6]). More recently, strong hardness amplification results have been proved for functions in P [4, 16, 17]. We also note that hardness amplification results have also been shown for one-way functions [34, 15, 5].

Given the above state-of-the-art picture, we raise a few natural questions and address them in this paper. There are many problems that are hard in the worst-case but easy on average. For example, 3-coloring is a well-known NP-hard problem, but it is an easy exercise to show that it can be solved in linear time with high probability on a random graph. This motivates us to distinguish within worst-case hard problems as to which of them remain hard on average. One way to go about this task is to identify which worst-case hard problem admits a hardness amplification theorem.

For which problems can we amplify hardness?
Can we identify a mathematical structure that allows us to amplify hardness?

The latter question has been implicitly addressed in literature (for example, if the problem has algebraic structure like in the case of computing permanent [26] or counting k-cliques [17]), but are quite specific and not broad enough to capture the class of problems that we believe are hard on average. In Section 1.2.1 we address the above two questions.

Next, we turn our attention to NP-hard problems. In a beautiful paper, O'Donnell [28] initiated the study of (non-uniform) hardness amplification in NP. His result was improved by [19] who showed that if for some function in NP (with n inputs) we have that any $s(n)$ size circuit fails to compute the function correctly on $1/poly(n)$ fraction of the inputs then, the hardness can be amplified to show that there is some function in NP such that any $s(\sqrt{n})^{\Omega(1)}$ size circuit fails to compute the function on $1/2 - 1/s(\sqrt{n})^{\Omega(1)}$ fraction of the inputs. However, the best uniform hardness amplification results (against algorithms as opposed to circuits) that have been achieved do not match the parameters of [19]: Trevisan [31, 8] improving on his previous work [30] showed that we can amplify hardness from $1/poly(n)$ to $1/2 - 1/polylog(n)$ for NP against randomized polynomial time algorithms (later extended to deterministic algorithms in [18]). However, it is important to note that all these hardness amplification results are for decision problems, and this leads us to our next question, do we gain anything by moving to search problems, or more precisely to the focus of this paper, to optimization problems?

Can we improve our hardness amplification results for optimization problems?
Can we prove stronger uniform hardness amplification results for MaxSAT*?*

Arguably, optimization problems are as natural as decision problems, but are strictly harder from the point of view of computational complexity. Does this mean we can either give simpler proofs of hardness amplification for optimization problems or prove stronger results? We address the above questions in Section 1.2.2.

We now shift our focus to the class P. As mentioned earlier, we have strong worst-case to average-case results established for problems in P [4, 17]. The drawback however is that they are all for counting problems. This is indeed inherent as the underlying technique these works use are the same as the one used to show worst-case to average-case reduction for the permanent problem. While counting the number of k-cliques (the problem considered in [17]) is a natural problem, and therefore hardness amplification for that problem is interesting, it still leaves the door open for proving hardness amplification for the search problem of just finding one k-clique in a graph (an easier problem and thus harder to amplify hardness).

Can we prove hardness amplification results for natural search problems in P*?*

Moreover, there is a barrier [1] to using the algebraic techniques of [4, 17] to obtain hardness amplification for important problems studied in fine-grained complexity such as computing the Longest Common Subsequence (LCS) and Edit Distance (Edit-Distance) for a pair of strings. In particular, if these string problems can be represented using low-degree polynomials, then we could obtain small speedups by using the polynomial method [11], which would imply new circuit lower bounds [2]. This suggests we might need to look beyond these algebraic techniques for proving hardness amplification for these string problems. Is there a different technique to prove hardness amplification in P? We address these aforestated questions in Section 1.2.3.

1.2 Our Results

Our main contribution is a general hardness amplification theorem for optimization problems which we state in Section 1.2.1. Next, we apply our main theorem to various problems. In Section 1.2.2 we state our hardness amplification theorems for various NP-hard problems such as Knapsack and MaxSAT. In Section 1.2.3 we state our hardness amplification theorems for various string problems in P such as LCS and Edit-Distance. Finally, in Section 1.2.4 we state our hardness amplification theorems for various problems in TFNP (believed to not be in P) such as Factoring and computing Nash equilibrium.

1.2.1 Hardness Amplification of Optimization Problems

Aggregation is a key tool in the field of hardness amplification. If a function f is hard to compute on a tiny fraction of the domain, then, intuitively, computing multiple instances of f in one shot should be hard on a larger fraction of the inputs. More formally, for a function $f : [N] \to \Sigma$ and $k \in \mathbb{N}$, its k-direct product encoding is defined as a function $f^{(k)} : [N]^k \to \Sigma^k$ mapping each tuple $(x_1, \ldots, x_k)$ into $(f(x_1), \ldots, f(x_k))$. Using standard techniques one can show a "direct product theorem" stating that if f is hard against $t(n)$ running-time algorithms on $\alpha(n)$-fraction of the domain, then $f^{(k)}$ is hard against $t'(n)$ running-time algorithms on $\approx k \cdot \alpha(n)$-fraction of its domain. But in order to utilize such a direct product result, we need to be able to stitch k-instances into a single (larger) instance. To address this task we introduce the following notion of direct product feasibility.

▶ **Definition 1** (Direct Product Feasibility; Informal statement)**.** *Let* Π *be an optimization problem. We say that* Π *is* (S, T)*-direct product feasible*[2] *if the exists a pair of deterministic algorithms* (Gen, Dec) *satisfying the following:*

[2] In the formal definition of direct product feasibility, it is defined for a pair of optimization problems (Π, Λ) for technical reasons which will be addressed later in Section 1.2.3. In the case $\Pi = \Lambda$ we formally call it as self direct product feasible and this notion coincides with the informal definition given here. For most of the applications given in this paper, self direct product feasibility notion suffices.

- Gen *takes as input* k *instances* $(I_1, \ldots, I_k)$ *of* Π *each of size* n *and outputs an instance* I' *of* Π *of size* $S(n,k)$.
- Dec *gets as input* $(I_1, \ldots, I_k)$, *the instance* I' *which is the output of* Gen *on input* $(I_1, \ldots, I_k)$, *an optimal solution for* I', *and* $i \in [k]$. *It outputs an optimal solution for the instance* I_i.
- *The running time of* Gen *and* Dec *is bounded by* $T(n,k)$.

Our main theorem is about hardness amplification for an arbitrary direct product feasible problem Π. In particular we show that if Π is hard against $t(n)$ running time randomized algorithms on a tiny fraction of the domain, then Π is hard on a much larger fraction of the domain against randomized algorithms with a similar running time.

▶ **Theorem 2** (Informal Statement). *Let* Π *be* (S,T)*-direct product feasible. Let* $\mathcal{D}(n)$ *be an efficiently samplable distribution over the instances of* Π *of size* n. *Assume the following:*

- *Any* $t(n)$ *running-time algorithm with success probability at least* $2/3$ *fails to compute an optimal solution on at least* $\alpha(n)$*-fraction of the inputs sampled from* $\mathcal{D}$.
- *Fix* $k = poly((\alpha(n))^{-1})$. *Then we have* $T(n,k) = o(t(n))$.
- *We can (deterministically) decide the optimality of a given solution to any instance in* $o(t(n))$ *time.*

Then there exists an efficiently samplable distribution $\mathcal{D}'(S(n,k))$ *over instances of* Π *of size* $S(n,k)$ *such that every* $t(n)$ *running-time algorithm with success probability at least* $2/3$ *fails to compute an optimal solution on at least* 99% *of the inputs sampled from* $\mathcal{D}'$.

Naturally, the distribution $\mathcal{D}'$ is defined as follows: Draw k independent samples $I_1, \ldots, I_k$ from $\mathcal{D}$, and output $\mathsf{Gen}(I_1, \ldots, I_k)$. The proof of our main theorem is based on a reduction using an oracle access to an algorithm that solves $\mathcal{D}'$ on 1% of the inputs, we convert it into an algorithm solving $\mathcal{D}$ on greater than $1 - \alpha(n)$ fraction of the inputs. The reduction is uniform, so in case that the algorithms $(\mathsf{Gen}, \mathsf{Dec})$ are uniform we get a uniform hardness amplification result.

Another key point is that our hardness amplification is a self-reduction, i.e., if a problem is somewhat hard against one distribution $\mathcal{D}$, then the same problem is much harder against a different distribution $\mathcal{D}'$.

To the best of our knowledge, this is the first result to study hardness amplification for optimization problems. It opens avenues to prove results in various subclasses as we will see in subsequent subsections.

1.2.2 Hardness Amplification for NP-hard Problems

In the NP world, we generalize the results of [28, 31] to optimization problems. In particular we show that if MaxSAT is hard to solve on $1/poly(n)$ fraction of the inputs of samples drawn from some samplable distribution $\mathcal{D}$. Then there exists a samplable distribution $\mathcal{D}'$ such that solving MaxSAT on $\mathcal{D}'$ is hard on at least $99/100$-fraction of the samples.

▶ **Theorem 3** (Informal Statement). *Let* $\mathcal{D}(n)$ *be a distribution over* 3-CNF *formulas with* n *variables and* $poly(n)$ *clauses, such that for every randomized algorithm* $\mathcal{A}$ *running in time* $poly(n)$, *we have:*

$$\Pr_{\Psi \sim \mathcal{D}}[\mathcal{A} \text{ finds a optimal assignment for } \Psi \text{ w.p. } \geq 2/3] \leq 1 - 1/poly(n).$$

Then there exists a distribution $\mathcal{D}'(n')$ over 3-CNF *formulas with n' variables poly(n') clauses, such that for every randomized algorithm $\mathcal{A}'$ running in time poly(n'), we have:*

$$\Pr_{\Psi' \sim \mathcal{D}'}[\mathcal{A}' \text{ finds a optimal assignment for } \Psi' \text{ w.p.} \geq 2/3] \leq 0.01.$$

Moreover, if $\mathcal{D}(n)$ is poly(n)-samplable then $\mathcal{D}'(n')$ is poly(n')-samplable.

Observe that the failure probability on $\mathcal{D}'$ is much larger than in [28, 31] and can even tend to 0 for a proper choice of our parameters. This can be achieved since we deal with optimization problems instead of decision problems.

We also remark that our reduction and the proof correctness are much simpler, and in particular we do not rely the hard core set lemma [20], a powerful and non-trivial key tool in the previous known proofs.

Our result easily extends into other NP-hard problems such as finding the largest clique in a graph, or finding smallest dominating set or vertex cover of a graph, etc.

However, there are other NP-hard problems for which establishing a hardness amplification result through Theorem 2 is not easy. A special highlight is that of proving such a result for the Knapsack problem, as it isn't immediately clear if it's direct product feasible for reasonable range of parameters. This is because, for the Knapsack problem, when we aggregate instances in the natural way, optimal solutions of one instance may interfere with other instances. Nonetheless, with some care, the direct product feasibility of Knapsack problem was established.

The Exponential Time Hypothesis (ETH) [22, 23, 10] asserts that that we cannot decide whether a given 3-CNF is satisfiable in time which is sub-exponential in the number of variables. That is a worst case assumption, and it raises a natural question arises: Can we prove stronger hardness amplification result based on ETH? In fact, can we prove a worst case to an average case hardness amplification based on ETH?

Our next theorem is a step towards proving such a worst-case to an average case reduction for MaxSAT.

▶ **Theorem 4** (Informal Statement). *Let $\mathcal{D}(n)$ be a distribution over* 3-CNF *instances with n variables and $O(n)$-clauses, such that for every randomized algorithm $\mathcal{A}$ running in time $2^{o(n)}$, we have:*

$$\Pr_{\Psi \sim \mathcal{D}}[\mathcal{A} \text{ finds an optimal assignment for } \Psi \text{ w.p.} \geq 2/3] \leq 1 - \frac{1}{2^{o(n)}}.$$

Then there exists a distribution $\mathcal{D}'(n')$ over 3-CNF *instances with n' variables and $2^{o(n')}$ clauses, such that for every polynomial time randomized algorithm $\mathcal{A}'$, we have:*

$$\Pr_{\Psi' \sim \mathcal{D}'}[\mathcal{A}' \text{ finds an optimal assignment for } \Psi' \text{ w.p.} \geq 2/3] \leq 0.01.$$

Heally et al. [19] proved a similar result for the non-uniform case. Our result is stronger in the sense that we use a weaker assumption: we rely on the ETH that is assuming that every *uniform* algorithm fails on $1/2^{o(n)}$-fraction of inputs. While Heally et al. use similar assumption against non-uniform algorithms.

1.2.3 Hardness Amplification in P

We investigate hardness amplification in P and can show results for string problems, such as LCS and Edit-Distance, which were not possible in previous works.

▶ **Theorem 5** (Informal statement). *Fix $\varepsilon > 0$. Let $\mathcal{D}(n)$ be an efficiently samplable distribution over the instances of* LCS/Edit-Distance *of length n. Assume that any $n^{2-\varepsilon}$ running-time algorithm with success probability at least $2/3$ fails to compute an optimal alignment on at least $1/n^{o(1)}$-fraction of the inputs sampled from $\mathcal{D}$. Then for some $\varepsilon' > 0$ there exists an efficiently samplable distribution $\mathcal{D}'(n^{1+o(1)})$ over instances of* LCS/Edit-Distance *of size $n^{1+o(1)}$ such that every $n^{2-\varepsilon'}$ running-time algorithm with success probability at least $2/3$ fails to compute an optimal solution on at least 99% of the inputs sampled from $\mathcal{D}'$.*

Recall from earlier in this section that Abboud [1] had pointed out a barrier to obtaining a result such as above, through algebraic techniques. Another similarity search problem that is studied along with LCS and Edit-Distance, is the problem of computing the Fréchet distance between two (discrete) curves. Strangely, this problem resists all natural approaches to show that it is direct product feasible. Therefore, it is an interesting question as to whether it is possible to show that it is direct product feasible (for relevant range of parameters) or whether it is a candidate for a problem that is not direct product feasible.

Additionally, we show hardness amplification for a very different kind of problem, that of computing the product of two matrices. We highlight this problem, as it does not directly follow from our main theorem (i.e., Theorem 2). Elaborating, a detail that was brushed under the carpet while discussing Theorem 2 was that, given an instance of an optimization problem and a candidate solution, we need to able to efficiently compute the value of the objective of the candidate solution for that instance. This naturally holds for all the problems considered in this paper except the task of computing the product of two matrices, i.e., we do not know a way to *deterministically* verify if the product of two matrices is equal to a given third matrix, which is significantly faster than actually multiplying the two given matrices and checking if it's equal to the third matrix [25, 33]. Nonetheless, we modify the proof of Theorem 2 to handle this issue.

1.2.4 Hardness Amplification in TFNP

Total problems (with not necessarily efficient verification of totality) are essentially equivalent to Optimization problems. The class TFNP is special as it is in an informal sense the intersection of Search NP and Optimization problems. Problems in TFNP capture problems in various areas such as game theory, cryptography, computational geometry, etc. We show that our general theorem can be applied to TFNP problems as well, and as an example show it for the Factoring problem and the End of a Line problem. The latter hardness amplification result directly implies the hardness amplification of various problems in game theory such as computing an approximate Nash equilibrium.

1.3 Open Problems

Our work leaves open several questions. We state a few of them below.

1.3.1 Stronger Hardness Amplification

In Theorem 4 we showed that if MaxSAT is hard to compute on $1 - 1/2^{o(n)}$-fraction of inputs for sub-exponential time algorithm, then there exists a distribution on which it is hard on a constant fraction of inputs for algorithms running in time $n^{\omega(1)}$. A natural open question is the following:

Can we improve Theorem 4 and get hardness amplified against sub-exponential time algorithms (instead of super-polynomial time algorithms)?

It seems to us that derandomized direct product theorems may serve as the key tool to address the above question (for example, see [21]). In particular, if one can prove a (strongly) derandomized version of [14] then it might be possible to both aggregate sub-exponentially many instances succinctly and sample from the (derandomized) direct product distribution efficiently.

1.3.2 Direct Product Feasibility

In this paper, we were able to show direct product feasibility for certain problems quite easily (for example, see Theorems 3 and 5), but had to work harder to prove them for some other problems (for example, Knapsack and Matrix Multiplication), and in some problem(s) were unable to establish the property of direct product feasibility (for example, computing Fréchet distance). This leads us to the following question.

Can we pinpoint what property of a problem makes it possible to establish direct product feasibility?

1.3.3 Gap Amplification versus Hardness Amplification

Direct Product theorems are key ingredients for both gap amplification and hardness amplification. Also, there are many philosophical similarities in the techniques known in literature of the aforementioned two kinds of amplifications. Thus we can ask the following (ambitious) question:

Can we obtain a trade-off between gap amplification and hardness amplification?

In particular, can we show that if one problem is hard to approximate on worst case within some factor $\alpha > 0$, then it is hard to approximate within a factor $\alpha/100$ on average? We note here that Feige [13], did answer the converse of this question, i.e., he used average case hardness assumptions to prove hardness of approximation results for various problems in NP.

It seems to us that analyzing the operation of performing a small perturbation on the given instance may be the right direction to proceed. Elaborating, consider a (worst case) hard distribution over gap instances of some problem. If we build a new distribution, which samples from the aforementioned distribution, then performs a small perturbation on the sampled gap instance, and outputs the perturbed instance, then we would still retain most of the gap in the instance sampled from the new distribution, but on the other hand, the fraction of instances on which it is hard to solve the problem should increase significantly. It would be interesting if this intuition/approach could be made to work.

A related question is to ask if we can improve our result in Theorem 4 (for example, by making progress on the question detailed in Section 1.3.1) using Gap-ETH [12, 27] (instead of ETH)?

1.3.4 Average Case Hard Problems in P

In this paper, we looked at average case hardness of some problems in P against some efficiently sampleable distribution but one can ask if we can achieve more.

Can we show for some natural problem in P *that it is hard to solve for the uniform distribution?*

Another important question stemming from cryptography [4] is whether we can construct a *fine-grained* one way function from worst case assumptions?

2 Proof Overview

We provide a proof overview for our hardness amplification result for the problem of finding the maximum clique in a graph and then in the subsequent section we will show how our general result (i.e., Theorem 2) would follow.

2.1 Hardness Amplification for Max Clique

To illustrate the main ideas behind our scheme let us focus on MaxCLIQUE, the problem of finding the largest clique in a given graph G.

Assume the existence of a distribution $\mathcal{D}$ over graphs on n vertices which is somewhat hard to compute. That is for every *randomized* algorithm $\mathcal{A}$ running in time $poly(n)$, we have

$$\Pr_{G\sim\mathcal{D}}[\mathcal{A} \text{ finds max-clique in } G \text{ w.p. } \geq 2/3] \leq 1 - 1/n. \tag{1}$$

We would like to prove the existence of a new distribution $\mathcal{D}'$ over graphs on $poly(n)$ vertices which is much harder to compute. That is, for every randomized algorithm $\mathcal{A}'$ running in time $poly(n')$, we have:

$$\Pr_{G'\sim\mathcal{D}'}[\mathcal{A}' \text{ finds max-clique in } G' \text{ w.p. } \geq 2/3] \leq 0.01. \tag{2}$$

Moreover if $\mathcal{D}$ is $poly(n)$-time samplable, then so is $\mathcal{D}'$.

Construction of New Distribution

$\mathcal{D}'$ samples a graph H as follows:

1. Independently sample $G_1, \ldots, G_k$ from $\mathcal{D}$, where $k = poly(n)$.
2. Define $V(H) = V(G_1) \dot\cup \cdots \dot\cup V(G_k)$.
3. For every $i \in [k]$, connect the vertices in $V(G_i)$ using the original edges in G_i.
4. For every $i, j \in [k]$ such that $i \neq j$, insert all the possible edges between G_i and G_j.
5. Output H.

Clearly, if $\mathcal{D}$ is $poly(n)$-time samplable, then so is $\mathcal{D}'$. Now assume for sake of contradiction, that there exists $\mathcal{A}'$ running in time $poly(n')$, violating Equation (2). We show the existence of an algorithm $\mathcal{A}$ running in time $poly(n)$ violating Equation (1).

The algorithm $\mathcal{A}$ on input graph G with n vertices is defined as follows:

1. Let $\mathcal{S}$ be an empty set.
2. Repeat following $O(n)$ times.
   a. Pick randomly $i \in [k]$.
   b. Independently sample $G_1, \ldots, G_{i-1}, G_{i+1}, \ldots G_k$ from $\mathcal{D}$.
   c. Construct H setting G_i to be G.
   d. Find clique in H using $\mathcal{A}'$.
   e. Restrict clique in H to the vertices of G and add it to $\mathcal{S}$.
3. Output the largest clique in $\mathcal{S}$.

Clearly, the running time of $\mathcal{A}$ is $poly(n)$, as $n' = poly(n)$ and the running time of $\mathcal{A}'$ is $poly(n')$. Our first observation is that for any graph H constructed by $\mathcal{A}$, and for every $i \in [k]$ the restriction of a maximal clique in H into G_i, is a maximal clique for G_i.

Let $\mathcal{A}_0$ be one iteration of step 2 of $\mathcal{A}$. If we show that $\mathcal{A}_0$ outputs maximum clique w.p. $\Omega(1/n)$ on $1 - 1/n$ fraction of samples from $\mathcal{D}$ then, $\mathcal{A}$ outputs maximum clique w.p. $2/3$ on $1 - 1/n$ fraction of samples from $\mathcal{D}$.

Now, observe that if instead of planting the given input graph G as the i-th subgraph of H, we were planting a uniformly random sample of $\mathcal{D}$, then we get a graph H which is drawn according to $\mathcal{D}'$. Consequently, if that was the case, then the success probability of $\mathcal{A}_0$ was equal the probability of $\mathcal{A}'$ and we were done.

Let $\mathcal{D}'_G$ denote the marginal distribution over H, where the graph G is planted at a random coordinate $i \in [k]$. We conclude the proof by showing that for $1 - 1/n$-fraction of instances G drawn from $\mathcal{D}$ we have:

$$\begin{aligned}&\Pr_{G' \sim \mathcal{D}'_G}\left[\mathcal{A}' \text{ finds max-clique in } G' \text{ w.p. } \geq 2/3\right] \\ \geq\ &\frac{\Pr_{G' \sim \mathcal{D}'}\left[\mathcal{A}' \text{ finds max-clique in } G' \text{ w.p. } \geq 2/3\right]}{2}.\end{aligned}$$

Towards this goal we use a result by Feige and Kilian [14] that was proven in the context of parallel repetition. Under minor manipulations their result can be stated as follows:

Let X be a universe and $\mathcal{T}$ be a distribution over X. Let $f : X^k \to \{0,1\}$. Define

$$\mu = \mathop{\mathbb{E}}_{x^k \sim \mathcal{T}^k}\left[f\left(x^k\right)\right],$$

$$\mu_x = \mathop{\mathbb{E}}_{i \in [k], x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_k \sim \mathcal{T}}\left[f(x_1, \ldots, x_{i-1}, x, x_{i+1}, \ldots x_k)\right].$$

$$\Pr_{x \sim \mathcal{T}}\left[|\mu_x - \mu| \geq k^{-1/6}\right] \leq k^{-1/6}, \tag{3}$$

To conclude the result, set X as the set of graphs with n vertices, and $\mathcal{T}$ be the distribution $\mathcal{D}$. We have $\mathcal{D}' = \mathcal{D}^k$. Define $f : X^k \to \{0,1\}$ by:

$$f(G') = 1 \iff \mathcal{A}' \text{ finds a maximal clique in } G \text{ w.p. } \geq 2/3.$$

In these notations,

$$\begin{aligned}\mu &= \Pr_{G' \sim \mathcal{D}'}\left[\mathcal{A}' \text{ finds max-clique in } G' \text{ w.p. } \geq 2/3\right] \\ \mu_x &= \Pr_{G' \sim \mathcal{D}'_G}\left[\mathcal{A}' \text{ finds max-clique in } G' \text{ w.p. } \geq 2/3\right].\end{aligned}$$

By an application of (3), and a proper choice of k, we get that for all but at most $k^{1/6}$-fraction of graphs G drawn according to $\mathcal{D}$, the success probability of $\mathcal{A}'$ on $\mathcal{D}'_G$ is $\Omega(n)$, as claimed.

2.2 Abstraction

In the previous subsection, we showed the main ingredients used for proving hardness amplification for the task of finding a maximal clique in a given graph. What were the properties of MaxCLIQUE that we utilized to prove the result?

One property that we used was that if we are given k input graphs $G_1, \ldots, G_k$, there exists an efficient way to construct a large graph H such that a maximal clique in H induces a maximal clique on each of the graphs G_i. The second property was that given a maximal clique in H there exists an efficient algorithm to construct a maximal clique on each of the graphs G_i.

These two properties are captured in Definition 1: The first property of a problem Π being Direct Product feasible is the existence of an efficient algorithm Gen stitching k instances $I_1, \ldots, I_k$ of Π into a larger instance I' of Π, such that: an optimal solution for I' induces an optimal solution for each of the instances I_i. The second property the existence of an efficient algorithm Dec converting an optimal solution for I' into an optimal solution of I'.

Once we show Π is Direct Product feasible then the rest of the proof goes through. Indeed, assuming the existence of a distribution $\mathcal{D}$ on instances of Π for which any efficient algorithm fails to compute on $1 - 1/n$ fraction of inputs, we define the distribution $\mathcal{D}', {\mathcal{D}'}_I$ as follows:

- $\mathcal{D}'$ is the k-product distribution of $\mathcal{D}$, where we pick k random samples from $\mathcal{D}'$ independently.
- ${\mathcal{D}'}_I$ is the distribution where we pick uniformly at random $i \in [k]$, and independently sample $I_1, \ldots, I_{i-1}, I_{i+1}, \ldots I_k$ from $\mathcal{D}$. Finally, we construct I' by setting I_i to be I.

Now we can use [14] to show that for most instances $I \sim \mathcal{D}$ to connect the success probability of $\mathcal{A}'$ on $\mathcal{D}'$ and ${\mathcal{D}'}_I$, to conclude the proof.

Remark about Direct Product results and Hardness Amplification

The direct product lemma at the heart of most hardness amplification results is the XOR lemma [34]. But here we critically use the fact the problem is total, so at the surface at least, our results are incomparable to the hardness amplification results for NP and EXP obtained via XOR lemmas.

References

1. Amir Abboud. Personal communication, 2019.
2. Amir Abboud, Thomas Dueholm Hansen, Virginia Vassilevska Williams, and Ryan Williams. Simulating branching programs with edit distance and friends: or: a polylog shaved is a lower bound made. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 375–388, 2016. doi:10.1145/2897518.2897653.
3. László Babai, Lance Fortnow, Noam Nisan, and Avi Wigderson. BPP has subexponential time simulations unless EXPTIME has publishable proofs. *Computational Complexity*, 3:307–318, 1993. doi:10.1007/BF01275486.
4. Marshall Ball, Alon Rosen, Manuel Sabin, and Prashant Nalini Vasudevan. Average-case Fine-grained Hardness. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2017, pages 483–496, New York, NY, USA, 2017. ACM. doi:10.1145/3055399.3055466.
5. Andrej Bogdanov and Alon Rosen. Input Locality and Hardness Amplification. *J. Cryptology*, 26(1):144–171, 2013. doi:10.1007/s00145-011-9117-y.
6. Andrej Bogdanov and Luca Trevisan. Average-Case Complexity. *Foundations and Trends in Theoretical Computer Science*, 2(1), 2006. doi:10.1561/0400000004.
7. Andrej Bogdanov and Luca Trevisan. On Worst-Case to Average-Case Reductions for NP Problems. *SIAM J. Comput.*, 36(4):1119–1159, 2006. doi:10.1137/S0097539705446974.
8. Joshua Buresh-Oppenheim, Valentine Kabanets, and Rahul Santhanam. Uniform Hardness Amplification in NP via Monotone Codes. *Electronic Colloquium on Computational Complexity (ECCC)*, 13(154), 2006. URL: http://eccc.hpi-web.de/eccc-reports/2006/TR06-154/index.html.
9. Jin-yi Cai, Aduri Pavan, and D. Sivakumar. On the Hardness of Permanent. In *STACS 99, 16th Annual Symposium on Theoretical Aspects of Computer Science, Trier, Germany, March 4-6, 1999, Proceedings*, pages 90–99, 1999. doi:10.1007/3-540-49116-3_8.
10. Chris Calabro, Russell Impagliazzo, and Ramamohan Paturi. A Duality between Clause Width and Clause Density for SAT. In *21st Annual IEEE Conference on Computational Complexity (CCC 2006), 16-20 July 2006, Prague, Czech Republic*, pages 252–260, 2006. doi:10.1109/CCC.2006.6.

11 Timothy M. Chan and Ryan Williams. Deterministic APSP, Orthogonal Vectors, and More: Quickly Derandomizing Razborov-Smolensky. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1246–1255, 2016. doi:10.1137/1.9781611974331.ch87.

12 Irit Dinur. Mildly exponential reduction from gap 3SAT to polynomial-gap label-cover. *ECCC*, 23:128, 2016. URL: http://eccc.hpi-web.de/report/2016/128.

13 Uriel Feige. Relations between average case complexity and approximation complexity. In *Proceedings on 34th Annual ACM Symposium on Theory of Computing, May 19-21, 2002, Montréal, Québec, Canada*, pages 534–543, 2002. doi:10.1145/509907.509985.

14 Uriel Feige and Joe Kilian. Two-Prover Protocols - Low Error at Affordable Rates. *SIAM J. Comput.*, 30(1):324–346, 2000. doi:10.1137/S0097539797325375.

15 Oded Goldreich, Russell Impagliazzo, Leonid A. Levin, Ramarathnam Venkatesan, and David Zuckerman. Security Preserving Amplification of Hardness. In *31st Annual Symposium on Foundations of Computer Science, St. Louis, Missouri, USA, October 22-24, 1990, Volume I*, pages 318–326, 1990. doi:10.1109/FSCS.1990.89550.

16 Oded Goldreich and Guy N. Rothblum. Worst-case to Average-case reductions for subclasses of P. *Electronic Colloquium on Computational Complexity (ECCC)*, 24:130, 2017. URL: https://eccc.weizmann.ac.il/report/2017/130.

17 Oded Goldreich and Guy N. Rothblum. Counting t-Cliques: Worst-Case to Average-Case Reductions and Direct Interactive Proof Systems. In *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 77–88, 2018. doi:10.1109/FOCS.2018.00017.

18 Parikshit Gopalan and Venkatesan Guruswami. Hardness amplification within NP against deterministic algorithms. *J. Comput. Syst. Sci.*, 77(1):107–121, 2011. doi:10.1016/j.jcss.2010.06.008.

19 Alexander Healy, Salil P. Vadhan, and Emanuele Viola. Using Nondeterminism to Amplify Hardness. *SIAM J. Comput.*, 35(4):903–931, 2006. doi:10.1137/S0097539705447281.

20 Russell Impagliazzo. Hard-Core Distributions for Somewhat Hard Problems. In *36th Annual Symposium on Foundations of Computer Science, Milwaukee, Wisconsin, USA, 23-25 October 1995*, pages 538–545, 1995. doi:10.1109/SFCS.1995.492584.

21 Russell Impagliazzo, Valentine Kabanets, and Avi Wigderson. New Direct-Product Testers and 2-Query PCPs. *SIAM J. Comput.*, 41(6):1722–1768, 2012. doi:10.1137/09077299X.

22 Russell Impagliazzo and Ramamohan Paturi. On the Complexity of k-SAT. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001. Preliminary version in CCC'99. doi:10.1006/jcss.2000.1727.

23 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which Problems Have Strongly Exponential Complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001. Preliminary version in FOCS'98. doi:10.1006/jcss.2001.1774.

24 Russell Impagliazzo and Avi Wigderson. $P = BPP$ if E Requires Exponential Circuits: Derandomizing the XOR Lemma. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing, El Paso, Texas, USA, May 4-6, 1997*, pages 220–229, 1997. doi:10.1145/258533.258590.

25 Marvin Künnemann. On Nondeterministic Derandomization of Freivalds' Algorithm: Consequences, Avenues and Algorithmic Progress. In *26th Annual European Symposium on Algorithms, ESA 2018, August 20-22, 2018, Helsinki, Finland*, pages 56:1–56:16, 2018. doi:10.4230/LIPIcs.ESA.2018.56.

26 Richard J. Lipton. New Directions In Testing. In *Distributed Computing And Cryptography, Proceedings of a DIMACS Workshop, Princeton, New Jersey, USA, October 4-6, 1989*, pages 191–202, 1989. doi:10.1090/dimacs/002/13.

27 Pasin Manurangsi and Prasad Raghavendra. A Birthday Repetition Theorem and Complexity of Approximating Dense CSPs. *CoRR*, abs/1607.02986, 2016. arXiv:1607.02986.

28 Ryan O'Donnell. Hardness amplification within NP. *J. Comput. Syst. Sci.*, 69(1):68–94, 2004. doi:10.1016/j.jcss.2004.01.001.

29 Madhu Sudan, Luca Trevisan, and Salil P. Vadhan. Pseudorandom Generators without the XOR Lemma. *J. Comput. Syst. Sci.*, 62(2):236–266, 2001. doi:10.1006/jcss.2000.1730.

30 Luca Trevisan. List-Decoding Using The XOR Lemma. In *44th Symposium on Foundations of Computer Science (FOCS 2003), 11-14 October 2003, Cambridge, MA, USA, Proceedings*, pages 126–135, 2003. doi:10.1109/SFCS.2003.1238187.

31 Luca Trevisan. On uniform amplification of hardness in NP. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005*, pages 31–38, 2005. doi:10.1145/1060590.1060595.

32 Luca Trevisan and Salil P. Vadhan. Pseudorandomness and Average-Case Complexity Via Uniform Reductions. *Computational Complexity*, 16(4):331–364, 2007. doi:10.1007/s00037-007-0233-x.

33 Virginia Vassilevska Williams and R. Ryan Williams. Subcubic Equivalences Between Path, Matrix, and Triangle Problems. *J. ACM*, 65(5):27:1–27:38, 2018. doi:10.1145/3186893.

34 Andrew Chi-Chih Yao. Theory and Applications of Trapdoor Functions (Extended Abstract). In *23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, USA, 3-5 November 1982*, pages 80–91, 1982. doi:10.1109/SFCS.1982.45.

Smooth and Strong PCPs

Orr Paradise
University of California, Berkeley, CA, USA
http://people.eecs.berkeley.edu/~orrp/
orrp@eecs.berkeley.edu

Abstract

Probabilistically checkable proofs (PCPs) can be verified based only on a constant amount of random queries, such that any correct claim has a proof that is always accepted, and incorrect claims are rejected with high probability (regardless of the given alleged proof). We consider two possible features of PCPs:

- A PCP is *strong* if it rejects an alleged proof of a correct claim with probability proportional to its distance from some correct proof of that claim.
- A PCP is *smooth* if each location in a proof is queried with equal probability.

We prove that all sets in $\mathcal{NP}$ have PCPs that are both smooth and strong, are of polynomial length, and can be verified based on a constant number of queries. This is achieved by following the proof of the PCP theorem of Arora, Lund, Motwani, Sudan and Szegedy (*JACM*, 1998), providing a stronger analysis of the Hadamard and Reed–Muller based PCPs and a refined PCP composition theorem. In fact, we show that any set in $\mathcal{NP}$ has a smooth strong *canonical* PCP of Proximity (PCPP), meaning that there is an efficiently computable bijection of $\mathcal{NP}$ witnesses to correct proofs. This improves on the recent construction of Dinur, Gur and Goldreich (*ITCS*, 2019) of PCPPs that are strong canonical but inherently non-smooth.

Our result implies the hardness of approximating the satisfiability of *"stable"* 3CNF formulae with *bounded variable occurrence*, where *stable* means that the number of clauses violated by an assignment is proportional to its distance from a satisfying assignment (in the relative Hamming metric). This proves a hypothesis used in the work of Friggstad, Khodamoradi and Salavatipour (*SODA*, 2019), suggesting a connection between the hardness of these instances and other stable optimization problems.

2012 ACM Subject Classification Theory of computation → Interactive proof systems

Keywords and phrases Interactive and probabilistic proof systems, Probabilistically checkable proofs, Hardness of approximation

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.2

Related Version A full version of the paper is available on the Electronic Colloquium on Computational Complexity as TR-19-023, https://eccc.weizmann.ac.il/report/2019/023/.

Acknowledgements This work was done during my time at the Weizmann Institute of Science. It originated in a question of Irit Dinur, and I am grateful to her. My heartfelt appreciation goes to Oded Goldreich for guiding me through all stages of this work, from communication of Irit's question to me and up to this very write-up. Many thanks to Madhu Sudan and Oded for providing the vector-valued low-degree test (Appendix C). I also thank Elad Granot and Roei Tell for the helpful discussions, and Amey Bhangale, Tom Gur, and Eylon Yogev for suggesting improvements to the write-up. I wish to thank an anonymous reviewer for pointing out an issue with the smoothness of the construction of Section 5.3 as it appeared in a previous version.

1 Introduction

A *probabilistically checkable proof system (PCP)* offers verification based only on a tiny amount of random locations in an alleged proof. It is *complete* and *sound*: correct claims have a proof that is always accepted, and incorrect claims are rejected with high probability

regardless of the alleged proof. The study of these systems culminated in the PCP theorem ([3, 2]), stating that membership in any set in $\mathcal{NP}$ can be verified by reading a *constant* number of random locations from a PCP of *polynomial* length.

While soundness guarantees that incorrect claims are rejected with high probability, what about *incorrect proofs* for *correct claims*? By definition, a proof of a claim is incorrect if it is not always accepted by the probabilistic verifier. But how often is it rejected? In a *strong* PCP, an alleged proof is rejected with probability proportional to its distance from a correct proof.

Strong PCPs are intuitively appealing: simply put, it is desirable to seek verification procedures that are sensitive to the correctness of the given claim *as well as the given proof*. From the perspective of property testing, the verifier of a strong PCP can be viewed as a (proximity oblivious) tester for the property of being a correct proof. In addition, strong PCPs have been used to construct better locally testable codes (see Section 1.3.1).

One immediately wonders if a *strong PCP theorem* holds as well; that is, whether any set in $\mathcal{NP}$ admits a *strong* PCP of polynomial length and constant query complexity. Dinur et al. answer this in the positive in a recent work [10],[1] however, their construction is inherently *non-smooth*, in the sense that certain locations in the proof are much more likely to be read than others.

A PCP is smooth if each location in its proof is equally likely to be read by the verifier. We expect natural PCPs to have smooth verifiers since, intuitively-speaking, we expect the verifier to treat all parts of the proof equally. Concretely, smooth PCPs are *tolerant* of errors, as a few corrupt locations in a correct proof still give high acceptance probability.[2] Prior works considered smoothness in the context of locally decodable codes (see Section 1.3.2).

Before moving on to the main result, let us motivate why smooth and strong PCPs are particularly natural in tandem. Fix a correct claim and consider two innate measures of the "incorrectness" of a proof: its probability of being rejected by the verifier and its distance from a correct proof. Denoting the first by ρ and the second by δ, a strong PCP guarantees that $\rho = \Omega(\delta)$. On the other hand, the tolerance of a smooth, constant-query PCP implies that $\rho = O(\delta)$. Thus, for a constant-query PCP that is both smooth and strong these measures coincide (up to a constant).

This work presents a construction of simultaneously smooth and strong PCPs of polynomial length for any set in $\mathcal{NP}$, verifiable by reading a constant number of bits from the proof. Specifically, we reanalyze and enhance the construction used in [3, 2] (with proof composition as in [6]) to obtain smooth and strong PCPs. The enhancements include the introduction of multi-piece PCPs, a smooth and strong-preserving transformation of these to single-piece PCPs, and a new composition theorem for smooth and strong PCPs.

Our result implies the hardness of approximating the satisfiability of *stable* 3CNF formulae with *bounded variable occurrence*, where *stability* means that the number of clauses violated by an assignment is proportional to its distance from a satisfying assignment (in the relative Hamming metric). We believe that the hardness of approximating 3SAT even under stability guarantees is related to the hardness of other stable optimization problems. Friggstad et al. provide evidence to this in [12], as they show that this result implies the hardness of approximating *perturbation-stable Euclidean k-means* (see Section 1.3.3).

---

[1] For technical reasons, their result is stated only for the class $\mathcal{UP} \subseteq \mathcal{NP}$, but can be easily adapted to suit all of $\mathcal{NP}$.

[2] Indeed, tolerance is (oppositely) related to strong PCPs which guarantee detection of errors in a correct proof. See next paragraph.

1.1 Main notions

In this section we formally define *strong PCPs* and *smooth PCPs*. But first, a reminder of standard PCPs and their basic properties.

▶ **Definition 1.1** (PCP). *A* probabilistically checkable proof system (PCP) *for a set* $S \subseteq \{0,1\}^*$ *is a probabilistic polynomial-time oracle machine* V*, called a* verifier *and denoted* V*, that satisfies the following conditions:*

- Completeness: *For all* $x \in S$ *there exists a proof* $\pi \in \{0,1\}^*$ *such that the verifier* V *accepts explicit input* x *and proof oracle* π *with probability* 1*.*
- Soundness: *For all* $x \notin S$ *and proof oracle* $\pi \in \{0,1\}^*$*, the verifier* V *rejects explicit input* x *and proof oracle* π *with probability at least* $1/2$*.*[3]

The maximal number of random coin tosses made by verifier V *on inputs of length* n *is its* randomness complexity*, denoted* $r(n)$*. The maximal number of queries made by the verifier* V *on inputs of length* n *is its* query complexity*, denoted* $q(n)$*.*

A PCP is *nonadaptive* if it determines all queries solely by its random coins and explicit input. All PCPs in this work are nonadaptive, and furthermore, they query the same number of bits regardless of the sampled coin sequence.

Notice that Definition 1.1 doesn't mention the *length* of the proof itself. That is because the number of possible locations the verifier might read in the proof can be upper-bounded based on its randomness and query complexities: a nonadaptive PCP that tosses r random coins and then makes q queries can query at most $q \cdot 2^r$ different locations in the proof. Thus, from here on we will ignore the proof length and focus on the randomness and query complexities.

1.1.1 Strong PCPs

Strong PCPs are PCPs that reject incorrect proofs even for correct claims with probability proportional to the distance of such proofs from correct ones. Throughout this work, *distance* refers to the relative Hamming distance: For a fixed alphabet Σ (one can think of $\{0,1\}$, but we will use different alphabets later), the *relative hamming distance* between strings $x, y \in \Sigma^n$ equals the fraction of locations on which they differ, and is denoted $\delta(x,y)$. If $\delta(x,y) < d$ then x is said to be *d-close* to y, and if $\delta(x,y) \geq d$ then x is *d-far* from y. The distance of a string x from a set $T \subseteq \{0,1\}^*$ is defined to be $\delta(x,T) := \min_{x' \in T \cap \{0,1\}^{|x|}} \delta(x,x')$, with the minimum over the empty set defined to be 1.

▶ **Definition 1.2** (Strong PCP). *A* strong PCP *for membership in set* S *with* strongness parameter $\alpha \in (0,1]$ *is a probabilistic polynomial-time oracle machine, called a* verifier *and denoted* V*, that satisfies the following conditions:*

- Completeness: *For all* $x \in S$ *there exists a proof* $\pi \in \{0,1\}^*$ *such that the verifier* V *accepts explicit input* x *and proof oracle* π *with probability* 1*. Such a proof* π *is called a* correct proof for x*.*
- Strong soundness:
  - *If* $x \notin S$ *then* x *has no correct proof.*
  - *Let* $\mathscr{P}(x)$ *denote the set of correct proofs for* x*. Then, the verifier rejects explicit input* x *and proof oracle* π *with probability at least* $\alpha \cdot \delta(\pi, \mathscr{P}(x))$*.*

[3] The constant $1/2$ can be replaced with any other constant $\alpha \in (0,1)$.

Note that strong soundness implies standard soundness, i.e. rejection of instances $x \notin S$ with constant probability regardless of the given proof oracle, because for these instances the verifier rejects with probability $\alpha \cdot \delta(x, \emptyset) = \alpha$.

1.1.2 Smooth PCPs

Smoothness is a straightforward notion and is defined for any oracle Turing machine. To us, oracles always have finite domains, and we associate the oracle $f\colon [n] \to \{0,1\}$ with an n-bit string $f(1)\cdots f(n)$.

▶ **Definition 1.3** (Smooth oracle machine). *A probabilistic oracle Turing machine M is* smooth *if for any explicit input and oracle, the probability that M queries each location of its oracle (in any of its queries) is equal. That is, given access to oracle f and letting $Q(j)$ be the event that M queries location j of f in any of its queries, it holds that $\mathbb{P}[Q(j)] = \mathbb{P}[Q(j')]$ for every $j, j' \in [|f|]$, where $|f|$ denotes the length of the oracle f.*

Additional discussion on smoothness and a nearly-equivalent definition can be found in Appendix B.

1.2 Contributions

1.2.1 Smooth and Strong PCPs for $\mathcal{NP}$

The main contribution is a proof of the following result.

▶ **Theorem 1.4** (Main result). *Every set in $\mathcal{NP}$ has a smooth and strong PCP with logarithmic randomness and constant query complexities.*

At this point one might wonder: A strong PCP rejects incorrect proofs with probability proportional to their distance from correct proofs – but *who are these correct proofs*? In the case of Theorem 1.4, we can say that the correct proofs for any fixed instance are obtained by a polynomial-time computable bijection of $\mathcal{NP}$-witnesses to correct proofs.

▶ **Theorem 1.5** (Main result, strengthened). *Fix a set $S \in \mathcal{NP}$, and let $\mathscr{W}(x)$ denote the set of $\mathcal{NP}$-witnesses for an instance x of S. Then, S has a PCP as in Theorem 1.4 with a polynomial-time computable* canonical proof strategy $\Pi : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ *such that for every $x \in \{0,1\}^*$, $\Pi(x, \cdot)$ is a bijection between the set $\mathscr{W}(x)$ and the set of correct proofs $\mathscr{P}(x)$.*

1.2.2 Hardness of approximation

Recall that the PCP theorem implies that for some $\rho \in (0,1)$, it is $\mathcal{NP}$-hard to distinguish 3CNF formulas that are satisfiable from ones in which any assignment violates at least a ρ fraction of the clauses. Theorem 1.4 yields a similar result for 3CNF formulas that are *"stable"* and have each variable occurring in a bounded number of clauses. Here *stability* means that the number of clauses violated by an assignment is (at least) proportional to its distance from a correct assignment (in the relative Hamming metric). Formally,

▶ **Definition 1.6** ((α, b)-`stable3SAT`). *A 3CNF formula φ is* α-stable *if any assignment that that is δ-far from a satisfying assignment violates at least an $\alpha\delta$ fraction of clauses in φ. A formula has* b-bounded-occurrence *if any variable occurs in at most b clauses. For constants α and b, the promise problem (α, b)-`stable3SAT` is distinguishing b-bounded-occurrence 3CNF formulas that are α-stable and satisfiable from ones in which any assignment violates at least an α fraction of the clauses.*

One motivation for our interest in stable and bounded-occurrence formulas is that they exhibit an interesting structural property. For a fixed satisfiable formula, we consider two natural measures of the "cost" (i.e., "badness") of an assignment. The first and most common one is the *fraction of clauses* violated by the assignment. The second is the *fraction of variables* on which the assignment disagrees with the closest satisfying assignment (i.e. the relative Hamming distance of the assignment from the set of satisfying assignments). We denote the first by δ and the second by ρ. Now, stable formulas have $\delta = \Omega(\rho)$, while bounded-occurrence formulas satisfy $\delta = O(\rho)$, since changing the value of a variable affects a bounded number of clauses. Hence, these two measures coincide for stable and bounded-occurrence formulas (up to a constant factor); the fraction of clauses unsatisfied by an assignment approximately reflects its distance from a satisfying assignment. Theorem 1.5 implies a hardness of approximation result for such formulas.

▶ **Corollary 1.7.** *There exist $\alpha \in (0,1)$ and $b \in \mathbb{N}$ such that (α, b)-*`stable3SAT` *is $\mathcal{NP}$-hard. Furthermore, it is $\mathcal{NP}$-hard under parsimonious Karp reductions.*

The proof of Corollary 1.7 are deferred to Appendix A. We proceed with a discussion of two of its implications.

Consider a *distance oracle* that, given a 3CNF formula φ and assignment σ, returns the (relative Hamming) distance of σ from the set of satisfying assignments of φ if φ is satisfiable, and answers arbitrarily otherwise. Efficiently finding a satisfying assignment given such an oracle can be done by greedily minimizing the distance returned by the oracle. What if instead we are given access to an *approximate* distance oracle, which returns the distance of an assignment from the set of satisfying assignments *up to some multiplicative constant*? Corollary 1.7 and the observation that precedes it imply that an approximate distance oracle is not enough to find even an approximately satisfying assignment; that is, that for some constant $\alpha \in (0,1)$, finding an assignment that satisfies more than an α-fraction of clauses is $\mathcal{NP}$-hard even when given access to an approximate distance oracle. This is because, as observed, the answers of an approximate distance oracle can be efficiently emulated for stable and bounded-degree formulas, and Corollary 1.7 asserts $\mathcal{NP}$-hardness of finding an approximately satisfying assignment for such formulas.

In addition to the aforementioned intrinsic motivation for the study of stable and bounded occurrence instances, Corollary 1.7 implies a hypothesis used in the recent work of Friggstad et al. [12, Hypothesis 1], yielding the first hardness of approximation result for perturbation-stable Euclidean k-means. More on this in Section 1.3.3.

1.2.3 Smooth and Strong Canonical PCPs of Proximity for $\mathcal{NP}$-relations

PCPs of Proximity (abbreviated *PCPPs*, aka *assignment testers*), introduced in [11, 6], are PCPs placed on an even tighter budget, with access to their input accounted for in their query complexity. Since PCPPs cannot read the entirety of their input oracle, they aren't able to distinguish inputs in the set from inputs *close* to being in the set. As such, PCPPs should satisfy a relaxed notion of soundness that requires them to reject (with high probability) only input oracles *far* from correct ones.

More generally, PCPPs verify membership of a *pair* $(x; y)$ in a *relation* $R \subseteq \{0,1\}^* \times \{0,1\}^*$, when given explicit (i.e. unaccounted) access to x and oracle (i.e. accounted) access to y, as well as access to a proof oracle.

▶ **Definition 1.8** (PCP of Proximity (PCPP)). *A* PCP of Proximity system (PCPP) *for relation* $R \subseteq \{0,1\}^* \times \{0,1\}^*$ *with* proximity parameter $\delta > 0$ *is a probabilistic polynomial-time oracle machine, called a* verifier *and denoted* V *such that the following hold:*

- Completeness: *If* $(x, y) \in R$ *then there exists a proof* π *such that the verifier* V *accepts explicit input* x*, input oracle* y *and proof oracle* π *with probability* 1*.*
- Soundness: *If* y *is* δ*-far from* $\{y' : (x, y') \in R\}$*, then for any proof oracle* π*, the verifier rejects explicit input* x*, input oracle* y *and proof oracle* π *with probability at least* $1/2$*.*

Strong canonical PCPPs

PCPP soundness is somewhat reminiscent of strong soundness, but note that in the former rejection probability is related to the distance of the *input oracle* from being correct, rather than the distance of the *proof oracle* from being correct (here we think of the explicit input as fixed).[4] Indeed, the adaptation of strong soundness to the setting of proximity verification, i.e. *strong PCPPs*,[5] combines these two requirements: a *strong PCPP* is required to reject with probability proportional the *maximum* between the distance of the input oracle y to a correct input oracle y', and the proof oracle π to a correct proof oracle π' for y'.

Actually, we won't bother to formally define strong PCPPs, because we show the existence of even *stronger* (pun intended) constructs. Our PCPPs have a *canonical* transformation of correct inputs to correct proofs, meaning that for each correct input $(x; y)$, our PCPPs have a unique *canonical proof* $\Pi(x; y)$ that is always accepted by the verifier, whereas all other strings are rejected with nonzero probability.[6] But with what probability? Right, the maximum between the distance of the input oracle y to a correct input oracle y', and the proof oracle π to the canonical proof for x and y' (i.e. $\delta(\pi, \Pi(x; y'))$).

▶ **Definition 1.9** (Strong canonical PCPP). *A* strong canonical PCPP *for relation* R *with* strongness parameter $\alpha \in (0, 1]$ *is a probabilistic polynomial-time oracle machine, called a* verifier *and denoted* V*, coupled with a polynomial-time computable* canonical proof strategy *denoted* $\Pi\colon \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$*, such that when the verifier is given explicit input* x *and access to an* input oracle y *and a* proof oracle π*, the following hold:*

- Canonical completeness: *The verifier accepts with probability* 1 *if and only if* $(x; y) \in R$ *and* π *is the corresponding canonical proof, i.e.* $\pi = \Pi(x; y)$*.*
- Strong canonical soundness: *Let* $R(x) \coloneqq \{y' : (x; y') \in R\}$*. Then, the verifier rejects with probability at least*

$$\alpha \cdot \min_{y' \in R(x)} \{\max\left(\delta(y, y'), \delta(\pi, \Pi(x; y'))\right)\}$$

In particular, if $R(x)$ *is empty then the verifier rejects with probability* α*.*

Again, strong canonical soundness implies standard PCPP soundness, e.g. rejection of any input y that is 0.1-far from $R(x)$ with constant probability, because in this case the verifier rejects with probability at least $\alpha \cdot \min\{0.1, 1\}$, where α is the (constant) strongness parameter.

---

[4] Furthermore, Definition 1.9 is *proximity-oblivious*, in the sense that rejection probability grows with the distance of the oracles from being correct, whereas Definition 1.8 offers rejection with constant probability of inputs whose distance from being correct is farther than some constant.

[5] Not to be confused with the related [17][Definition 5.7], which we will soon refer to as *strong canonical PCPPs*. See also Footnote 7.

[6] This is the bijection mentioned in Theorem 1.5.

▶ Remark 1.10. Previously (e.g. [17, 15, 10]), strong canonical PCPPs were considered only for *unambiguous relations*; that is, only for relations $R \subseteq \{0,1\}^* \times \{0,1\}^*$ for which $|R(x)| \leq 1$ for any x. Our definition is more general as it does not place a restriction on the relation R.[7] Furthermore, Definition 1.9 allows the verifier to take an explicit input (as in [6]), whereas past works studied PCPPs that have access to input and proof oracles but no (auxiliary) explicit input.

Smooth PCPPs

PCPP verifiers are oracle machines that have *two* oracles, and we say that a PCPP is smooth if it is smooth on each of its oracle. Formally, we generalize Definition 1.3 to suit a *t-oracle* Turing machine for any constant $t \in \mathbb{N}$, which is a machine that has access to a t oracles (where t is a constant).

▶ **Definition 1.11** (Smooth multi-oracle machine)**.** *A probabilistic t-oracle Turing machine M is* smooth *if for any explicit input and oracles, the probability that M queries each location of each of its oracle (in any of its queries) is equal. That is, given access to oracles $f_1, \ldots, f_t$ and letting $Q_\ell(i,j)$ be the event that the ith query of M is to location j of f_ℓ, it holds that $\mathbb{P}\left[\bigcup_{i=1}^{q} Q_\ell(i,j)\right] = \mathbb{P}\left[\bigcup_{i=1}^{q} Q_\ell(i,j')\right]$ for each $\ell \in [t]$ and every $j, j' \in [|f_\ell|]$, where $|f_\ell|$ denotes the length of the ℓth oracle f_ℓ.*

We prove the following theorem.

▶ **Theorem 1.12.** *Every $\mathcal{NP}$-relation has a smooth and strong canonical PCPP with logarithmic randomness complexity and constant query complexities.*

Notice that a PCPP for relation R yields a PCP for the set $S_R \coloneqq \{x : \exists y \ (x;y) \in R\}$: a proof that $x \in S_R$ (in the PCP for S_R) is composed of some y such that $(x;y) \in R$, followed a proof (in the PCPP for R) that $(x;y) \in R$. Furthermore, the PCP for S_R retains the strongness of the PCPP for R and, under a reweighing of the input oracle (presented in Section 2), smoothness is retained as well. Therefore, Theorem 1.5 follows from Theorem 1.12.

1.3 Related work

1.3.1 Strong (canonical) soundness

The term *strong* in the definition of strong PCPs is inspired by *strong locally testable codes (strong LTCs)*, which are codes whose local test rejects with probability proportional to the distance of the input from the code. In fact, strong canonical PCPPs have seen numerous uses in works on strong LTCs, as follows.

Goldreich and Sudan [17] defined strong canonical PCPPs in their work on strong LTCs,[8]and constructed such PCPPs for certain linear codes. An extension of this initial construction saw use by Gur and Rothblum [19] as they obtained strong LTCs that allow for a relaxed notion of local decoding (of [6, Section 4.2]). Goldreich et al. [15] later improved these

[7] In fact, our work is the first to make a semantic distinction between *strong* and *strong canonical*, recognizing strong canonical PCPPs as special type of strong PCPPs in which there is an efficient *canonical* transformation between $\mathcal{NP}$-witnesses and proofs, in addition to strong soundness. In previous works, the terms *strong PCPP* and *strong canonical PCPP* were used synonymously, which is consistent with our distinction, given that these works considered PCPPs only for unambiguous relations. See Section 1.3.1 for more on previous works using strong canonical PCPPs.

[8] See Footnote 7 for a warning on the usage of the terms *strong* and *strong canonical* in previous works.

codes, again utilizing strong canonical PCPPs. Gur et al. [18] employed strong canonical PCPPs in their work on relaxed locally correctable codes. We stress that all these works featured PCPPs for linear subspaces (e.g. linear codes).

As mentioned, Dinur et al. [10] characterized *unambiguous* $\mathcal{NP}$ *(*$\mathcal{UP}$*)* in terms of strong canonical PCPPs, under the original and more restricted definition of strong canonicality generalized in this work (see Remark 1.10).

In [14, Section 13.2.2], strong PCPs are presented from the perspective of property testing as *locally testable proofs*, in analogy to locally testable codes.

1.3.2 Smoothness

Like strongness, smoothness too has its roots in coding theory, with the work Katz and Trevisan [21] examining *smooth locally decodable codes*. These are codes whose local decoding algorithm reads each bit in an alleged codeword with *approximately* equal probability (cf. Definition 1.3, which requires the probability to be *exactly* equal). Since its inception, this property has appeared in numerous works relating to locally decodable codes, e.g. [16, 24, 13]). Goldreich and Sudan [17, Definition 5.14] considered a similar feature for their Linear Inner Proof Systems (LIPS), which are fundamentally different from PCPs.

To avoid possible confusion, it is worth pointing out that the smoothness referred to in this work is as in the aforementioned works in coding theory, and *not* as in the *smooth label cover* of Khot [22].

The PCPPs of Theorem 1.12 are used in the recent work of Alman and Chen [1] which shows an explicit construction of rigid matrices using an $\mathcal{NP}$ oracle. They make use only of the smoothness of these PCPPs, but not the additional strong canonicality feature.[9]

1.3.3 Hardness of perturbation-stable Euclidean k-means

The hardness of approximating bounded-degree `stable3SAT` (Corollary 1.7) is the starting point of the first hardness of approximation result for perturbation-stable instances of Euclidean k-means, due to Friggstad et al. [12]. This connection between `stable3SAT` and perturbation-stable problems is an interesting direction for future research, so we provide a brief description of their result.

The study of optimization on perturbation-stable instances was initiated by Bilu and Linial [8] and Awasthi et al. [4] as a way of focusing on instances that can "occur in reality" (to quote the former). We consider the *Euclidean k-means* problem and its perturbation-stable instances, which are defined as follows:

- An instance of the k-means problem consists of $k \in \mathbb{N}$, *dimension* $d \in \mathbb{N}$, a *metric* $\mu : \mathbb{R}^d \times \mathbb{R}^d \to [0, \infty)$, *data points* $X \subseteq \mathbb{R}^d$, and *candidate centers* $C \subseteq \mathbb{R}^d$. The objective is to choose *centers* $S \subseteq C$ such that $|S| = k$ so as to minimize $\mathbb{P}_{x \in X} \left[\min_{c \in S} (\mu(x, c))\right]$. An instance is *Euclidean* if μ is the Euclidean metric.
- Fix $\gamma > 1$ and metric μ. A γ-perturbation of μ is a function $\mu' : \mathbb{R}^d \times \mathbb{R}^d \to [0, \infty)$ such that for any $x \neq y \in \mathbb{R}^d$,

  $$1 \leq \frac{\mu'(x, y)}{\mu(x, y)} \leq \gamma$$

  Notice that μ' is not necessarily a metric.

[9] As opposed to [12] (see Section 1.3.3) which uses both features simultaneously.

- For some fixed $\gamma > 1$, an instance (k, d, X, C, μ) of k-means is *γ-perturbation-stable* if it has a unique optimal solution $S^* \subseteq C$, and for any γ-perturbation μ' of μ, S^* is the optimal solution of the related instance (k, d, X, C, μ'). The *γ-perturbation-stable Euclidean k-means* problem is the k-means problem as previously described, under the promise that instances are *Euclidean* and are γ-perturbation-stable.

While solving (general, non-stable) Euclidean k-means is known to be $\mathcal{NP}$-hard to approximate [5], Awasthi et al. showed that introducing some perturbation-stability makes the problem easy; namely, they show that $(\sqrt{2}+3)$-stable Euclidean k-means can be solved exactly in polynomial time. Could the introduction of *any* amount of perturbation-stability render Euclidean k-means easy, or even just easy to approximate? The answer is *no*, as demonstrated by Friggstad et al. [12]. Their result is conditioned on an hypothesis asserting hardness of approximation of *bounded-occurrence and* `stable3SAT` (see Section 1.2.2), which is implied by Corollary 1.7.

▶ **Theorem 1.13** ([12]). *Assuming Corollary 1.7, there are $\gamma > 1$ and $\varepsilon > 1$ such that γ-perturbation-stable Euclidean k-means cannot be approximated within factor ε, unless $\mathcal{RP} = \mathcal{NP}$.*

1.4 Proof outline

To prove Theorem 1.12 we construct a PCPP with the necessary properties for the *circuit valuation* relation, denoted CircuitVal, which consists of all pairs $(C; y)$ such that circuit C accepts when given y as input. Then, any $\mathcal{NP}$-relation R has a PCPP (with the same properties) that, given explicit input x, efficiently computes a circuit C_x such that $(x; y) \in R$ if and only if C_x accepts y, and then runs the PCPP of CircuitVal on explicit input C_x and the same input and proof oracles. Following are highlights of our smooth and strong canonical PCPP for CircuitVal.

Multi-piece PCPPs (Section 2)

The PCPP for CircuitVal will be a variant of the PCPP presented in [3, 2, 6]. However, even a high-level inspection of this construction reveals that it is not at all smooth: for example, one building block (the Hadamard-based PCPP, mentioned below) consists of two "pieces" from which the verifier samples uniformly random locations, with the first piece substantially shorter than the second (so its bits are queried far more often). Indeed, this PCPP is not smooth when viewed as a single proof, but when partitioned into two *proof-pieces* (given as two proof-piece oracles), the verifier is smooth *as a three-oracle machine* (one input oracle and two proof-piece oracles). We present a transformation of multi-piece PCPPs to single-piece ones that *simultaneously preserves smoothness and strong canonicality*. This is done by replacing each proof-piece with a list of copies, so that the length of each list of copies is roughly the same.[10]

[10] To be clear, the Hadamard-based PCPP is but one example of non-smoothness (or rather, multi-piece smoothness) in the construction of [3, 2, 6]; the Reed–Muller-based PCPP, as well as PCPP composition, exhibit similar traits. Thus, our transformation of multi-piece PCPPs to single-piece PCPPs is used by all components of our construction, not just by the Hadamard-based PCPP.

Composing smooth strong canonical PCPPs (Section 3)

The run of a nonadaptive PCPP verifier can be viewed as a two-step process: first, it tosses some random coins and generates a *residual (decision) circuit* and *query locations* based on the coins it tossed, and then it queries its oracles and accepts or rejects according to the residual circuit's computation of their answers. A strong canonical and *robust* PCPP is such that, in expectation, the distance of its oracles' answers from satisfying the residual circuit reflects the distance of the oracles (in their entirety) from correct ones (i.e. a correct input oracle and a canonical proof oracle). Our *composition theorem* asserts that for a smooth strong canonical and robust PCPP, replacing the residual circuit's computation with an additional probabilistic verification (by an *inner* smooth strong canonical PCPP) yields a smooth strong canonical PCPP.

With a composition theorem at hand, we turn to the construction of smooth strong canonical and robust PCPPs, whose composition gives the PCPP postulated by Theorem 1.12.

The Hadamard-based smooth strong canonical PCPP (Section 4)

This is the Hadamard-based PCPP presented in [2], used as the innermost PCPP of the composition. Its proofs are based on the Hadamard encoding of the input oracle, and its verifier checks that the proof oracle encodes the input oracle (a "consistency check"), and uses the structure of the Hadamard code to check that the input oracle satisfies the (explicitly given) circuit. To show strong canonicality, we consider three cases:

Case 1: Both the input and proof oracles are close to correct ones. That is, the input oracle y is close to an input y' that is accepted by the (explicitly given) circuit, and the proof oracle π is close to its canonical proof oracle Π' of y'. The verifier checks consistency by performing a *strong codeword test* on the proof oracle, and checks that the decoding of the proof oracle agrees with the input oracle on a random bit. Strong testability means precisely that the first check rejects with probability $\Omega(\pi, \Pi')$, and the local decoding of a random location rejects with probability $\Omega(y, y')$.

Case 2: The input oracle is far from any correct input oracle. Then, standard PCPP soundness guarantees rejection with high probability.

Case 3: The proof oracle is far from the canonical proof of the input oracle. If the proof oracle is close to the canonical proof for some input $y' \neq y$ then the consistency check between the proof oracle and the input oracle rejects with probability $\Omega(\delta(y, y'))$. Otherwise the proof oracle is far from any canonical proof, and is therefore rejected with high probability by the strong codeword test.

Lastly, as mentioned above, we observe that it is smooth as a two-piece PCPP, and can therefore be transformed to a single-piece strong canonical PCPP.

The Reed–Muller-based smooth strong canonical robust PCPP (Section 5)

This PCPP is used in the outer layers of the composition, so we must show that it is smooth and *robust* strong canonical. Again, smoothness amounts to observing that it is multi-piece smooth. The analysis of its strong canonicality follows the same lines of the Hadamard-based PCPP, this time relying on a generalization of the strong Reed–Muller codeword test (provided by Oded Goldreich and Madhu Sudan in Appendix C). We first present a smooth strong canonical robust PCPP whose proofs are over a large alphabet, and then show that alphabet reduction (encoding each letter in an error correcting code) preserves smoothness and strong canonicality.

2 Multi-piece PCPPs

As said in Section 1.4, several of the PCPPs in the [2] construction are not smooth per se, but they spread their queries smoothly on each of a few significant *proof-pieces*. Such *multi-piece* PCPPs are required to satisfy a stricter form of strong canonicality, in which the rejection probability is related to the maximum between *each proof-piece oracle* to its corresponding proof-piece in a correct proof (and, as usual, the distance of the input oracle y to a satisfying input oracle y'). Motivated by this observation, we define *multi-piece strong canonical PCPPs*.

▶ **Definition 2.1.** *For a constant t, a* t-piece strong canonical PCPP system *for relation R with* strongness parameter $\alpha \in (0,1]$ *is a probabilistic polynomial-time oracle machine, called a* verifier *and denoted V, coupled with a sequence of polynomial-time computable* canonical proof-piece strategies*, denoted $\Pi_1, \ldots, \Pi_t \colon \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$, such that when the verifier is given explicit input x and access to an* input oracle y *and a* proof-piece oracles $\pi_1, \ldots, \pi_t$*, the following hold:*

- Canonical completeness: *The verifier accepts with probability* 1 *if and only if $(x;y) \in R$ and π_i is the corresponding canonical proof-piece, i.e. $\pi_i = \Pi_i(x;y)$, for each $i \in [t]$.*
- Strong canonical soundness: *Let $R(x) \coloneqq \{y' : (x;y') \in R\}$. Then, the verifier rejects with probability at least*

$$\alpha \cdot \min_{y' \in R(x)} \{\max\left(\delta(y,y'), \delta(\pi_1, \Pi_1(x;y')), \ldots, \delta(\pi_t, \Pi_t(x;y'))\right)\}$$

In particular, if $R(x)$ is empty, then the verifier rejects with probability α.

For each $i \in [t]$, the ith proof-piece length complexity*, denoted $\ell_i(n)$, is the length of the ith proof piece. The* ith proof-piece query complexity*, denoted $q_i(n)$, is the number of queries the verifier issues to the ith proof-piece given an input of length n.*[11]

2.1 From multi-piece to single-piece

We present a smoothness-preserving transformation of strong canonical multi-piece PCPPs to strong canonical PCPPs that use a single proof oracle. This transformation is not only convenient (for example, composition of multi-piece PCPPs gives one a multi-headache), but is also necessary for Theorem 1.12 which asserts the existence of *single-piece* smooth strong canonical PCPPs for every $\mathcal{NP}$ relation.

▶ **Lemma 2.2.** *Suppose that, for some constant t, a relation R has a* t-piece *smooth strong canonical PCPP with strongness parameter α, randomness complexity $r(n)$, and query complexity $q(n)$. Then, R has a* single-piece *smooth strong canonical PCPP with strongness parameter $\alpha/3$, randomness complexity $O(r(n) + \log q(n))$ and query complexity $O(q(n))$.*

Proof. Notice that if proof-piece lengths vary significantly, then simply concatenating the proof-pieces will not do, because bits of shorter pieces are sampled with higher probability than bits of longer pieces. Instead, each proof-piece is replaced with a list of copies such that each list is of equal length (up to a factor related to the proof-piece's query complexity). That is, the number of copies in the ith list is proportional to q_i/ℓ_i, which, by multi-piece smoothness, equals the probability that a bit in the ith proof-piece is queried by the multi-piece verifier.

[11] *Tedious comment:* We require the number of queries that the verifier issues to each proof-piece to depend only on the explicit input's length. This requirement is met by the all PCPPs used in this work.

The single-piece verifier emulates the multi-piece verifier on a random choice of proof-pieces (each proof-piece sampled uniformly from its list of alleged copies), and checks consistency of the copies in each list. Some care must be taken so that the consistency check does not harm smoothness (for example, checking consistency against a fixed copy would result in its bits being queried more often than others).

Strong canonicality of the single-piece verifier then follows from examining two possible cases: The first is that the given lists are noticeably inconsistent (i.e., there is a large discrepancy between the alleged copies), in which case the consistency check rejects with good probability. Otherwise, the lists are almost entirely consistent (i.e., each list of the given proof essentially consists of copies of some proof-piece), and then strong canonicality follows from strong canonicality of the multi-piece verifier.

Following is a detailed description and analysis of the construction. Let V be the postulated t-piece smooth strong canonical PCPP verifier with canonical proof-piece strategies $\Pi_1, \ldots, \Pi_t \colon \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$. We construct a single-piece PCPP $\overline{V}$ and start by describing its canonical proof strategy $\overline{\Pi}$. Fixing $(x, y) \in R$, we use the following notation:

- r denotes the number of random coins tossed by V when given x as explicit input.
- q_i denotes the number of queries V makes to the ith proof-piece oracle when given x as explicit input. The total number of queries V makes to all proof-piece oracles is $q \coloneqq \sum_{i=1}^{t} q_i$
- $\Pi_i \coloneqq \Pi_i(x; y)$ denotes the ith canonical proof-piece of $(x; y)$, and $\ell_i \coloneqq |\Pi_i|$ denotes its length.

Now, let $\overline{\Pi_i}$ be a list of $m_i \coloneqq (q_i + 1) \prod_{j \neq i} \ell_j$ copies of Π_i.[12] Indeed, m_i is proportional to the probability that a certain fixed bit of Π_i is sampled, namely q_i/ℓ_i.[13] The canonical proof of $(x; y)$ in the single-piece PCPP $\overline{V}$ is then the concatenation of all $\overline{\Pi_i}$'s.

We describe the run of the new verifier $\overline{V}$ given explicit input x and access to input oracle y and proof oracle $\overline{\pi} = (\overline{\pi_1}, \ldots, \overline{\pi_t})$, where $\overline{\pi_i} = (\pi_i^1, \ldots, \pi_i^{m(i)})$ is a list of $m(i)$ strings, each of length ℓ_i:

1. *Emulation:* For each $i \in [t]$, sample a uniformly random index $c_i \in [m_i]$. Next, emulate V on explicit input x, input oracle y and proof-piece oracles $\pi_1^{c_1}, \ldots, \pi_t^{c_t}$, rejecting if the emulation rejected. Let J_i denote the set of locations that V queries in $\pi_i^{c_i}$.
2. *Consistency check:* For each $i \in [t]$, sample uniformly from the remaining copies $c_i' \in [m_i] \setminus \{c_i\}$ and a uniformly random $j_i \in J_i$ and check that $\pi_i^{c_i'}[j_i] = \pi_i^{c_i}[j_i]$. That is, check that $\pi_i^{c_i}$ and $\pi_i^{c_i'}$ agree on a uniformly random location from the locations queried by the emulated verifier V in Step 1.[14]

   (Note that smoothness of V implies that j_i is uniformly distributed in $[\ell_i]$ (as detailed in Appendix B.2), and that only $\pi_i^{c_i'}[j_i]$ needs to be queried in this step.)

The single-piece verifier $\overline{V}$ uses t more queries than the emulated (multi-piece) verifier V since the consistency check requires querying an additional location from each list. The number of random coins is upper-bounded by $O(t^2 \cdot (r + \log q)) = O(r + \log q)$.

---

[12] The value $q_i + 1$ is used instead of q_i to account for the additional query made by the consistency check (Step 2 in the description of $\overline{V}$).

[13] Letting $m_i \coloneqq (q_i + 1)L/\ell_i$ for any L that is a common multiple of $\{\ell_i\}_{i \in [t]}$ would have worked as well.

[14] Indeed, the consistency check can be implemented in several other ways, for example without reusing the emulation copy c_i in the consistency check (i.e. checking consistency between c_i' and some c_i''), or by uniformly sampling j_i from all of $[\ell_i]$ rather than from J_i. However, other implementations require setting m_i to other (less informative) values.

Canonical completeness follows by observing that a proof is accepted with probability 1 if and only if it is formed of consistent lists (i.e. each list holds copies of some proof-piece), such that the proof-pieces in each list form a canonical proof in the multi-piece PCPP. We prove the remaining properties:

Smoothness

Fix a location in the proof oracle $\overline{\pi}$, which is the jth location of $\pi_i^{k_i}$ for some $i \in [t]$, $k_i \in [m(i)]$ and $j \in [\ell_i]$. This location is queried if and only if one of two disjoint events occur:

- In Step 1, $k_i = c_i$ *and* $j_i \in J_i$, i.e. the jth location of $\pi_i^{c_i}$ was queried by the emulated verifier. By multi-piece smoothness of the emulated verifier V, this event occurs with probability $\frac{1}{m_i} \cdot \frac{q_i}{\ell_i}$.
- In Step 1, $k_i \neq c_i$ *and* $j_i \in J_i$. In addition, in Step 2, $k_i = c_i'$ *and* the jth location is chosen from J_i (i.e. from the set of all locations queried by the verifier). This event occurs with probability $(1 - \frac{1}{m_i}) \cdot \frac{q_i}{\ell_i} \cdot \frac{1}{m_i - 1} \cdot \frac{1}{q_i} = \frac{1}{m_i} \cdot \frac{1}{\ell_i}$.

All in all, we have that the probability that this location is queried is

$$\frac{1}{m_i} \cdot \frac{q_i}{\ell_i} + \frac{1}{m_i} \cdot \frac{1}{\ell_i} = \frac{q_i + 1}{m_i \cdot \ell_i} = \frac{1}{\prod_{i=1}^{t} \ell_i}$$

Therefore, each bit in $\overline{\pi}$ is queried with equal probability.

Strong canonical soundness

Let α be the strongness parameter of the multi-piece verifier V. We show that the (single-piece) PCPP $\overline{V}$ has strongness parameter $\alpha/3$. Fix explicit input x, input oracle y and proof oracle $\overline{\pi} = (\overline{\pi_1}, \ldots, \overline{\pi_t})$, where $\overline{\pi_i}$ is purported to contain m_i copies of the ith canonical proof-piece for $(x; y)$. If $R(x)$ is empty, then the emulated verifier (and therefore the single-piece verifier) rejects with probability $\alpha \cdot 1$ for any choice of (alleged) copies c_i, so we may focus on the case that $R(x) \neq \emptyset$. Let $y' \in R(x)$ be a minimizer of ρ, defined

$$\rho \coloneqq \max\left(\delta(y, y'), \delta\left(\overline{\pi_1}, \overline{\Pi}_1(x; y')\right), \ldots, \delta\left(\overline{\pi_t}, \overline{\Pi}_t(x; y')\right)\right) \geq \max\left(\delta(y, y'), \delta\left(\overline{\pi}, \overline{\Pi}(x; y')\right)\right) \quad (1)$$

It suffices to show that the verifier rejects with probability at least $\frac{\alpha}{3} \cdot \rho$. Assume wlog that the maximum in the left hand side of Equation (1) is obtained in the first proof-piece oracle, so $\rho = \max\left(\delta(y, y'), \delta(\overline{\pi}_1, \overline{\Pi}_1(x; y'))\right)$. We proceed by examining the case that the first proof-piece has noticeable inconsistency, and the case where it is almost entirely consistent.

Case 1: The first proof-piece list is noticeably inconsistent: $\mathbb{E}_{c_1 \neq c_1' \in [m_1]}\left[\delta(\pi_1^{c_1}, \pi_1^{c_1'})\right] \geq \rho/3$. Smoothness of the emulated verifier implies that a uniformly random location from the set of locations queried in $\overline{\pi}_1$ is distributed uniformly in $[\ell_i]$ (as detailed in Appendix B.2). Hence, the probability that the consistency check rejects equals the expected distance between two distinct (alleged) copies sampled uniformly, which is assumed to be at least $\rho/3$.

Case 2: The first proof-piece list is almost entirely consistent: $\mathbb{E}_{c_1 \neq c_1' \in [m_1]}\left[\delta(\pi_1^{c_1}, \pi_1^{c_1'})\right] < \rho/3$. By an averaging argument, there exists a "typical" copy such that all other (alleged) copies in first-proof piece list are close to it in expectation; namely, there is an $a \in [m_1]$ such that $\mathbb{E}_{c_1 \in [m_1]}\left[\delta(\pi_1^{c_1}, \pi_1^{a})\right] \leq \rho/3$. We argue that since the alleged copies are close to the typical copy (in expectation), the multi-piece verifier rejects with probability similar to that of the strong canonical single-piece verifier given the typical copy as proof.

For each c_1, let $y^{c_1} \in R(x)$ be a minimizer of $\max\left(\delta(y, y^{c_1}), \delta(\pi_1^{c_1}, \Pi_1^{c_1})\right)$ where $\Pi_1^{c_1} := \Pi_1(x; y^{c_1})$. Using strong canonicality of the emulated verifier, we can lower-bound the probability that the single-piece verifier rejects in the emulation check by

$$\begin{aligned}
&\mathop{\mathbb{E}}_{c_1 \in [m_1]}\left[\alpha \cdot \max\left(\delta(y, y^{c_1}), \delta(\pi_1^{c_1}, \Pi_1^{c_1})\right)\right] \\
&\quad \geq \alpha \cdot \mathop{\mathbb{E}}_{c_1}\left[\max\left(\delta(y, y^{c_1}), \delta(\pi_1^{a}, \Pi_1^{c_1})\right)\right] - \alpha \cdot \mathop{\mathbb{E}}_{c_1}\left[\delta(\pi_1^{c_1}, \pi_1^{a})\right] \\
&\quad \geq \alpha \cdot \max\left(\delta(y, y^{a}), \delta(\pi_1^{a}, \Pi_1^{a})\right) - \alpha \cdot \frac{\rho}{3} \qquad (2)
\end{aligned}$$

where the second inequality used the minimality of y^a, which means that for any c_1 it holds that $\max\left(\delta(y, y_1^c), \delta(\pi_1^a, \Pi_1^{c_1})\right) \geq \max\left(\delta(y, y^a), \delta(\pi_1^a, \Pi_1^a)\right)$. We can also lower-bound the distance of the typical copy π_1^a from its corresponding canonical proof piece Π_1^a by

$$\begin{aligned}
\delta(\pi_1^a, \Pi_1^a) &\geq \mathop{\mathbb{E}}_{c_1 \in [m_1]}\left[\delta(\pi_1^{c_1}, \Pi_1^a)\right] - \mathop{\mathbb{E}}_{c_1 \in [m_1]}\left[\delta(\pi_1^{c_1}, \pi_1^a)\right] \qquad (3) \\
&\geq \mathop{\mathbb{E}}_{c_1 \in [m_1]}\left[\delta(\pi_1^{c_1}, \Pi_1^a)\right] - \frac{\rho}{3} = \delta\big(\overline{\pi}_1, \overline{\Pi}_1(x; y^a)\big) - \frac{\rho}{3}
\end{aligned}$$

Combining Equations (2) and (3), we have that the verifier rejects with probability at least

$$\alpha \cdot \max\left(\delta(y, y^a), \delta\big(\overline{\pi}_1, \overline{\Pi}_1(x; y^a)\big) - \frac{\rho}{3}\right) - \alpha \cdot \frac{\rho}{3} \geq \alpha \cdot \rho - \alpha \cdot \frac{2\rho}{3} = \alpha \cdot \frac{\rho}{3}$$

where the inequality is because $y^a \in R(x)$ is not necessarily the minimizer of Equation (1), i.e. it could be that $y^a \neq y'$ (and then $\max\left(\delta(y, y^a), \delta(\overline{\pi}_1, \overline{\Pi}_1(x; y^a))\right) \geq \rho$). ◀

3 Composing smooth strong canonical PCPPs

In this section, we adapt the composition theorem of Ben-Sasson et al. [6] to the strong canonical setting.

We can think of a run of nonadaptive PCPP verifier as a two-step process: first, it tosses some random coins and generates a *residual (decision) circuit* and *query locations* based on the coins it tossed, and then it queries its oracles and feeds their answers to the residual circuit, accepting or rejecting accordingly. Hence, the verifier accepts if and only if the residual circuit and oracle answers are in CircuitVal. *PCPP composition* replaces the naive verification of this claim of membership (in CircuitVal) by a probabilistic verification. That is, an *inner verifier* probabilistically checks that the oracles' answers satisfy the *outer verifier's* residual decision circuit. The resulting *composite verifier* accepts or rejects according to the inner verifier's decision.

The strong inner verifier rejects with probability that is proportional to the distance of the oracles' answers from satisfying the outer residual circuit. As such, this distance should reflect the distance of the outer oracles (in their entirety) from correct ones. In other words, if the outer oracles are far from correct ones, the outer verifier's queries should not only be rejected, but be *far* from being accepted by its residual circuit. In such a case we say that the outer verifier is *robust*.

As in the composition of [6], when it comes to randomness and query complexities, the composite verifier enjoys the best of both worlds: broadly speaking, its query complexity is inherited from the inner verifier, and its randomness complexity is (mostly) determined by the outer verifier. So, composing an outer verifier of low randomness complexity with an inner verifier that issues a few queries yields a composite verifier with low randomness *and* query complexities. The contribution of this section is in showing that, in addition, such composition can be made to preserve smoothness and strong canonicality.

3.1 Strong canonical robust PCPPs

First, we describe the run of a nonadaptive verifier as a two-step process: first sampling random coins, then querying its oracles and computing a decision. Formally,

▶ **Definition 3.1** (PCPPs, restated)**.** *Given explicit input x and oracle access input y and proof π, a (nonadaptive) PCPP verifier for relation R of randomness complexity $r(n)$ and query complexity $q(n)$ runs as follows:*

1. Sample. *The verifier uniformly samples a* coin sequence $c \in \{0,1\}^{r(|x|)}$*. Based on x and c, the verifier generates* query locations $I := I_c := (i_1, \ldots, i_{q(|x|)})$ *and* residual circuit $D := D_c$*. Note that I contains the locations of queries to both y and π.*
2. Query and compute. *The verifier queries oracles y and π according to the query locations I. Denoting the answers to these queries by $y\pi[I]$, the verifier then computes $D(y\pi[I])$ and outputs 1 ("Yes") if and only if $y\pi[I]$ satisfies D.*

The event that the verifier V accepts, i.e. "$V^{y,\pi}(x) = 1$", is equal by definition to "$y\pi[I_c] \in \mathtt{Sat}(D_c)$", where $\mathtt{Sat}(D_c)$ denotes the set of satisfying inputs of D_c, and the randomness in both events being the coin sequence c. The decision complexity *$d(n)$ is the maximal size of a residual circuit generated when the verifier is given explicit input of length n.*

Strong canonical *robustness* guarantees that the expected distance of the oracles' answers from satisfying the residual circuit is proportional to the distance of these oracles from being correct, and is formally defined as follows.

▶ **Definition 3.2** (Strong canonical robust PCPPs)**.** *A strong canonical PCPP V for relation R with canonical proof strategy $\Pi\colon \{0,1\}^* \to \{0,1\}^*$ is a strong canonical* robust *PCPP* (RPCPP) *with strongness parameter $\alpha \in (0,1]$ if, in addition to the conditions of Definition 1.8, it satisfies* strong canonical robust soundness*:*

- *For any $(x;y)$ such that $(x;y) \notin R$ it holds that $\Pi(x;y) = \emptyset$. When given explicit input x, input oracle y and proof oracle π, the expected distance of the bits queried by V from being accepted by the residual circuit is at least*

$$\alpha \cdot \min_{y' \in R(x)} \{\max\left(\delta(y,y'), \delta(\pi, \Pi(x;y'))\right)\} \tag{4}$$

That is,

$$\mathop{\mathbb{E}}_{c}\left[\delta(y\pi[I_c], \mathtt{Sat}(D_c))\right] \geq \begin{cases} \alpha \cdot \min_{y' \in R(x)} \{\max\left(\delta(y,y'), \delta(\pi, \Pi(x;y'))\right)\} & \text{if } R(x) \neq \emptyset \\ \alpha & \text{if } R(x) = \emptyset \end{cases}$$

▶ Remark 3.3. This definition differs from a natural adaptation of the original definition of robustness ([6, Definition 2.6]) in that it requires the *expected* distance to be large, rather than require the distance be large with high probability. Markov's inequality implies that Definition 3.2 is stronger.

3.2 The composition theorem

Following the two-step description of a run of a PCPP verifier in Definition 3.1, a PCPP verifier accepts explicit input x, input oracle y and proof oracle π if and only if the answers received from its proof oracle (denoted $y\pi[I]$) satisfy its residual circuit D. In a nutshell, PCPP composition is done by replacing the verification of the claim "$y\pi[I]$ *satisfies* D" with a probabilistic verification by an *inner PCPP verifier*.

Before turning to the theorem and its proof, we define an additional property we require from outer verifiers.

▶ **Definition 3.4** (Residual circuit distance). *For some constant $\Delta > 0$, we say that a PCPP verifier V has* residual circuit distance Δ *if for any input x and coin sequence c, any two distinct inputs satisfying the residual circuit D_c are at least Δ-far apart.*

In later sections, we show that the outer PCPP whose composition yields Theorem 1.12 satisfies this additional property for some constant $\Delta > 0$. Hence, it suffices to prove the composition theorem for PCPPs with constant residual circuit distance.

▶ **Theorem 3.5.** *Assume there are $\alpha_{\text{out}}, \alpha_{\text{in}}, \Delta \in (0,1]$ and $r_{\text{out}}, r_{\text{in}}, d_{\text{out}}, d_{\text{in}}, q_{\text{in}} \colon \mathbb{N} \to \mathbb{N}$ such that the following holds:*

- *The relation R has a smooth strong canonical RPCPP, denoted V_{out}, with residual circuit distance Δ, strongness parameter α_{out} and randomness and decision complexities r_{out} and d_{out} respectively.*
- CircuitVal *has a strong canonical PCPP, denoted V_{in}, with strongness parameter α_{in} and randomness, query and decision complexities r_{in}, q_{in} and d_{in} respectively.*

Then, the relation R has a strong canonical PCPP, denoted V_{comp}, with the following properties:

- *Randomness complexity $r_{\text{out}} + r_{\text{in}} \circ d_{\text{out}}$.*
- *Query complexity $q_{\text{in}} \circ d_{\text{out}}$.*
- *Decision complexity $d_{\text{in}} \circ d_{\text{out}}$*
- *Strongness parameter $\alpha_{\text{out}} \cdot \alpha_{\text{in}} \cdot \Delta/4$*

Furthermore, if V_{in} is smooth then R has a smooth strong canonical PCPP with strongness shrinking by a third (to $\alpha_{\text{out}} \cdot \alpha_{\text{in}} \cdot \Delta/12$). If V_{in} is robust (resp. has residual circuit distance) then V_{comp} is robust (resp. has residual circuit distance).

Notice that smoothness of the outer verifier is used for strong canonicality of the composite verifier.

Proof of Theorem 3.5. We affix the term *outer*, *inner* or *composite* when discussing components of the *outer*, *inner* or *composite* verifiers. For example, an *inner canonical proof* is a proof obtained from the canonical proof strategy of the inner PCPP.

We start by describing the canonical proof strategy of the composite PCPP, which consists of two proof-piece oracles: an *outer proof-piece* denoted Π, and an *inner proof-piece* denoted T. For any explicit input x and input oracle y with $(x; y) \in R$, the outer proof-piece is the outer canonical proof that $(x; y) \in R$ (for V_{out}), and the inner proof-piece is the concatenation (over all possible coin sequences c of the outer verifier) of the inner canonical proof that $y\Pi[I_c]$ satisfies D_c (for V_{in}), which we denote by T_c. With these in mind, we proceed by describing the composite verifier.

▶ **Algorithm 3.5.1** (PCPP composition [6, Section 2.4]). The composite PCPP system has verifier V_{comp} that takes explicit input x, input oracle y, and two proof-piece oracles: an outer proof-piece π, and an inner proof-piece $\tau = (\tau_c)_c$, with c ranging over all possible coin sequences of the outer verifier. It runs as follows:

1. Emulates the *Sample* step of the outer verifier, obtaining a coin sequence c, outer query locations I_c and outer residual circuit D_c. (V_{comp} does not issue any queries yet!)
2. Emulates the inner verifier with explicit input D_c, input oracle $y\pi[I_c]$ and proof oracle τ_c. V_{comp} accepts if and only if the inner verifier accepted.

Since Algorithm 3.5.1 is the same composition used in the proof of [6, Theorem 2.7], the composite PCPP enjoys all properties described there, and in particular it is a PCPP with the required complexities. Canonical completeness follows from the canonical completeness of the outer and inner verifiers, and so we turn to prove strong canonical soundness. Then,

if V_{in} is smooth then V_{comp} is two-piece smooth, and we may apply Lemma 2.2 to obtain a smooth strong canonical composite PCPP for R while losing an additional factor $1/3$ in strongness.

Fix an explicit input x, input oracle y, and alleged proof-piece oracles and π and τ, where π corresponds to the outer proof-piece and $\tau \coloneqq (\tau_c)_c$ corresponds to the inner proof-piece. If $R(x)$ is empty then we may refer to the standard soundness analysis of the composite PCPP, so we focus on the case that $R(x)$ is nonempty. Let y' be a satisfying input that minimizes the maximal distance between the given oracles and correct oracles; that is, the maximum between $\delta(y, y')$, $\delta(\pi, \Pi)$ and $\delta(\tau, T)$, where Π and T are the canonical outer and inner proof-pieces for y'. Denote the distance between the given oracles and the correct oracles by $\delta_y \coloneqq \delta(y, y')$, $\delta_\pi \coloneqq \delta(\pi, \Pi)$ and $\delta_\tau \coloneqq \delta(\tau, T) = \mathbb{E}_c\left[\delta(\tau_c, T_c)\right]$. We shall show that the verifier rejects with probability $\alpha_{\text{out}} \cdot \alpha_{\text{in}} \cdot \Delta \cdot \max\left(\delta_y, \delta_\pi, \delta_\tau\right)/4$. The analysis considers two cases.

Case 1: $\max\left(\delta_y, \delta_\pi\right) \geq \frac{\Delta}{4} \cdot \delta_\tau$. For any fixed outer coin sequence c, the strong canonical inner verifier rejects the circuit D_c, input oracle $y\pi[I_c]$ and proof oracle τ_c with probability $\alpha_{\text{in}} \cdot \delta(y\pi[I_c], \texttt{Sat}(D_c))$. The rejection probability of the composite verifier equals the expected rejection probability of the inner verifier (over random c), therefore the composite verifier rejects with probability at least $\mathbb{E}_c\left[\alpha_{\text{in}} \cdot \delta(y\pi[I_c], \texttt{Sat}(D_c))\right]$ which, by the robust strong canonicality of the outer verifier, is at least $\alpha_{\text{out}} \cdot \alpha_{\text{in}} \cdot \max\left(\delta_y, \delta_\pi\right) \geq \alpha_{\text{out}} \cdot \alpha_{\text{in}} \cdot \frac{\Delta}{4} \cdot \max\left(\delta_y, \delta_\pi, \delta_\tau\right)$.

Case 2: $\max\left(\delta_y, \delta_\pi\right) < \frac{\Delta}{4} \cdot \delta_\tau$. As a warm-up, consider the case that $\delta_y = \delta_\pi = 0$. In this case, $y\pi[I_c]$ satisfies D_c for any outer coin sequence c. Recall that the outer verifier has constant residual circuit distance, so all other inputs that satisfy D_c are far from $y\pi[I_c]$, and therefore the inner verifier's strong canonicality implies that the composite verifier rejects with probability $\Omega(\delta(\tau_c, T_c))$. Taking expectation over the coin sequence c, we have that the composite verifier rejects with probability $\Omega(\delta_\tau) = \Omega(\max\left(\delta_y, \delta_\pi, \delta_\tau\right))$.
In the actual analysis, δ_y and δ_π are not necessarily zero, but the general ideas of the warm-up are still applicable: using the smoothness of the outer verifier, we can relate $\mathbb{E}_c\left[\delta(y\pi[I_c], \delta(y'\Pi[I_c]))\right]$ with δ_y and δ_π. Since δ_y and δ_π are assumed to be $O(\delta_\tau)$, then for sufficiently many c's it holds that $y\pi[I_c]$ is close to $y'\Pi[I_c]$. Then, the constant residual circuit distance of the verifier implies that for these c's the inner verifier rejects with probability proportional to $\delta(\tau_c, T_c)$. Details follow.
We say that a fixed coin sequence c is *good* if $y\pi[I_c]$ is $\Delta/2$-close to $y'\Pi[I_c]$. Strong canonicality of the inner verifier means that the probability it rejects circuit D_c, input oracle $y\pi[I_c]$ and proof oracle τ_c is at least

$$\alpha_{\text{in}} \cdot \min_{s \in \texttt{Sat}(D_c)} \left\{\max\left(\delta(y\pi[I_c], s), \delta(\tau_c, T_c(D_c; s))\right)\right\}$$

The outer verifier has residual circuit distance Δ, so for good coins c, $\delta(y\pi[I_c], s) \geq \Delta/2$ for all $s \in \texttt{Sat}(D_c) \setminus \{y'\Pi[I_c]\}$. Therefore, for good c's, $\max\left(\delta(y\pi[I_c], s), \delta(\tau_c, T_c(D_c; s))\right)$ is at least $\delta(\tau_c, T_c)$ when taking $s = y'\Pi[I_c]$, and is at least $\Delta/2$ when s satisfies D_c but differs from $y'\Pi[I_c]$. Hence,

$$\mathbb{P}\left[V_{\text{in}}^{\tau_c}(D_c) = 0 \,|\, c \text{ good}\right] \geq \alpha_{\text{in}} \cdot \min\left\{\Delta/2, \delta(\tau_c, T_c)\right\} \geq \frac{\alpha_{\text{in}} \cdot \Delta}{2} \cdot \delta(\tau_c, T_c) \tag{5}$$

Now, we show that since δ_y and δ_π are $O(\delta_\tau)$, then the expected distance between inner proofs over good c's upper-bounds the distance between all inner proofs δ_τ (up to constants). Smoothness of the outer verifier implies that sampling a random coin

sequence c and location j in I_c is the same as uniformly sampling from $[|y\pi|]$ (as detailed in Appendix B.2). Therefore,

$$\delta(y\pi, y'\Pi) \coloneqq \mathop{\mathbb{P}}_{k}[y\pi[k] \neq y'\Pi[k]] = \mathop{\mathbb{P}}_{j,c}[y\pi[I_c][j] \neq y'\Pi[I_c][j]] = \mathop{\mathbb{E}}_{c}[\delta(y\pi[I_c], y'\Pi[I_c])]$$

Thus,

$$\delta(y\pi, y'\Pi) \geq \frac{\Delta}{2} \cdot \mathop{\mathbb{P}}_{c}\left[\delta(y\pi[I_c], y'\Pi[I_c]) > \frac{\Delta}{2}\right] = \frac{\Delta}{2} \cdot \mathop{\mathbb{P}}_{c}[c \neg\text{good}]$$

Note that $\max(\delta_y, \delta_\pi) \geq \delta(y\pi, y'\Pi)$, so by the assumption that $\max(\delta_y, \delta_\pi) < \Delta \cdot \delta_\tau/4$ we have that $\delta_\tau/2 \geq \mathbb{P}[c \neg\text{good}]$. In addition,

$$\delta_\tau = \mathop{\mathbb{E}}_{c}[\delta(\tau_c, T_c)] \leq \mathop{\mathbb{E}}_{c}[\delta(\tau_c, T_c) \mid c \text{ good}] \cdot \mathop{\mathbb{P}}_{c}[c \text{ good}] + \mathop{\mathbb{P}}_{c}[c \neg\text{good}]$$

Which implies

$$\mathop{\mathbb{E}}_{c}[\delta(\tau_c, T_c) \mid c \text{ good}] \cdot \mathop{\mathbb{P}}_{c}[c \text{ good}] \geq \delta_\tau/2 \tag{6}$$

Using Equations (5) and (6), the probability that the composite verifier rejects the given input and proof is at least

$$\mathop{\mathbb{E}}_{c}[\mathbb{P}[V_{\text{in}}^{\tau_c}(x) = 0] \mid c \text{ good}] \cdot \mathop{\mathbb{P}}_{c}[c \text{ good}] \geq \frac{\alpha_{\text{in}} \cdot \Delta}{2} \cdot \mathop{\mathbb{E}}_{c}[\delta(\tau_c, T_c) \mid c \text{ good}] \cdot \mathop{\mathbb{P}}_{c}[c \text{ good}] \geq \frac{\alpha_{\text{in}} \cdot \Delta}{4} \cdot \delta_\tau$$

If V_{in} is a strong canonical *robust* PCPP, then we note that the lower bounds on the *rejection probability* hold for the *expected distance of the bits read from* $(y\pi[I_c], \tau_c)$ *from satisfying* V_{comp}*'s residual circuit.*[15] As for residual circuit distance: any two strings satisfying the composite residual circuit satisfy the inner residual circuit, so the residual circuit distance of the inner verifier is inherited by the composite verifier. ◀

3.3 Composing the construct of Theorem 1.12

Now that we know *how* to compose PCPPs, let's talk about *which* PCPPs we compose. For now, we postulate two smooth strong canonical (R)PCPPs of certain complexities and reckon that their composition yields a PCPP of logarithmic randomness and constant queries (their actual construction and analysis constitutes the rest of this work).

▶ **Proposition 3.6** (Hadamard-based PCPP)**.** *There exists a smooth strong canonical PCPP for* CircuitVal *of quadratic randomness and constant query complexities.*

▶ **Proposition 3.7** (Reed–Muller-based RPCPP)**.** *There exists a smooth strong canonical robust PCPP for* CircuitVal *of logarithmic randomness complexity, polylogarithmic decision complexity, and constant residual circuit distance.*

As in [2, 6], the Reed–Muller-based RPCPP is composed with itself to obtain a smooth strong canonical RPCPP of logarithmic randomness complexity, poly log log decision complexity and constant residual circuit distance. The resulting RPCPP is then composed (as an outer verifier) with an inner Hadamard-based PCPP to obtain a smooth strong canonical PCPP of logarithmic randomness and constant query complexities, proving Theorem 1.12.

[15] To transform the multi-piece robust PCPP to a single-piece robust PCPP, we note that Lemma 2.2 holds for robust PCPPs as well: for strong canonicality, rather than analyzing the *rejection probability of the verifier*, we can analyze the *expected distance from being accepted by the verifier* (using exactly the same reasoning).

4 The Hadamard-based smooth strong canonical PCPP

We now turn to the actual construction of smooth strong canonical PCPPs, starting with the Hadamard-based PCPP of [2, Section 4] as presented in [20, Chapter 4].

4.1 Algebrization and the Hadamard code

A circuit and its input can be expressed as a system of quadratic equations and an assignment, such that the input satisfies the circuit if and only if the assignment satisfies the equation system. The Hadamard-based PCPP verifier capitalizes on this fact, with the proof for an input consisting of the truth tables of evaluations of all (linear and) quadratic equations on the assignment corresponding to the input. First, let's take a more detailed look at this correspondence, which is sometimes called an *algebrization* of the combinatorial problem of circuit valuation to the algebraic problem of quadratic equation valuation.

▶ **Definition 4.1** (Computational extension). *Let C be a circuit of size n with n_0 input gates and let $y \in \{0,1\}^{n_0}$. The* computational extension of y w.r.t C *is the string corresponding to the values output by each gate of C when computing y, and is denoted $y_C \in \{0,1\}^n$. That is, $y_C[i]$ is the output of the ith gate of C when computing input y. Assuming wlog that the first n_0 gates of C are its input gates, it holds that $y = y_C[1] \cdots y_C[n_0]$.*

▶ **Definition 4.2** (Outer product). *The* outer product *of vectors $u, v \in \{0,1\}^n$, denoted $u \otimes v$, is the n^2 dimensional vector obtained by "flattening" the $n \times n$ matrix whose entry in the ith column and jth row is $u[i] \cdot v[j]$. That is, $u \otimes v[(i-1)n + j] := u[i] \cdot v[j]$ for all $i, j \in [n]$.*

▶ **Proposition 4.3.** *There exists a polynomial-time computable mapping that maps circuit C of size n with m input bits to an $n \times n^2$ matrix A_C and vector $b_C \in \{0,1\}^n$ such that input $y \in \{0,1\}^m$ satisfies C if and only if $y_C \in \{0,1\}^n$ satisfies $A_C(y_C \otimes y_C) = b_C$.*

Proof sketch. Let C be a circuit of size n that has n_0 input gates. The reduction computes $b \in \{0,1\}^n$ and $a_1, \ldots, a_n \in \{0,1\}^{n^2}$ according to Table 1, and lets A_C be the matrix whose ith row is a_i. ◀

Table 1 Mapping of gates to equations. $e_{i,j} \in \{0,1\}^{n^2}$ is all zeroes except for coordinate $(i-1)n + j$.

Gate	First input gate	Second input gate	Gate type	Reduction output
i	j	k	`AND`	$a_i := e_{i,i} + e_{j,k}$ $b_i := 0$
			`OR`	$a_i := e_{i,i} + e_{j,j} + e_{k,k} + e_{j,k}$ $b_i := 0$
			`NOT`	$a_i := e_{i,i} + e_{j,j}$ $b_i := 1$
			`OUTPUT`	$a_i := e_{i,i}$ $b_i := 1$
			`INPUT`	$a_i := \overline{0}$ $b_i := 0$

The *Hadamard encoding* of a string $y \in \{0,1\}^n$ is the evaluation of all linear equations (over $\mathbb{F}_2$) on n variables on the assignment y. Formally,

▶ **Definition 4.4.** *The* Hadamard encoding *of a vector $y \in \{0,1\}^n$ is the function* $\mathrm{Had}_y \colon \{0,1\}^n \to \{0,1\}$ *given by* $\mathrm{Had}_y(z) \coloneqq y \cdot z = \sum_{i=1}^{n} y[i]z[i] \mod 2$.

Recall some useful facts about this encoding:

▶ **Fact 4.5.** *For all $n \in \mathbb{N}$:*

Distance. *A function $f \colon \{0,1\}^n \to \{0,1\}$ is linear if and only if there exists $y \in \{0,1\}^n$ such that $f = \mathrm{Had}_y$. For all $y \neq y'$ it holds that $\delta(\mathrm{Had}_y, \mathrm{Had}_{y'}) = 1/2$, associating Had_y with its 2^n-bit-long truth table. That is to say that the Hadamard code has* relative distance $1/2$.

Strong Testability. *Consider a test that given $f \colon \{0,1\}^n \to \{0,1\}$ uniformly samples $x, x' \in \{0,1\}^n$ and accepts if and only if*

$$f(x) + f(x') = f(x + x')$$

As shown in [9], if f is δ-far from all Hadamard codewords then the test rejects with probability at least $\min(\delta/2, 1/6)$. *Note that the test makes three queries to f and tosses $2n$ random coins.*

Self-correction. *Consider a self-correction procedure that given $f \colon \{0,1\}^n \to \{0,1\}$ and $x \in \{0,1\}^n$, uniformly samples $r \in \{0,1\}^n$ and outputs $f(x+r) - f(r)$. It holds that if f is δ-close to the Hadamard codeword $\widetilde{f}$ then for all x the procedure outputs $\widetilde{f}(x)$ with probability at least $1 - 2\delta$. Note that the procedure makes two queries to f and tosses n random coins.*

4.2 The PCPP and its analysis

▶ **Algorithm 4.6** (The Hadamard-based PCPP [2])**.** The canonical proof-piece strategies of circuit C and satisfying input y are the Hadamard encodings of y_C and $y_C \otimes y_C$ (the computational extension of y, and the outer product of the latter with itself, respectively). That is, $\Pi_1(C; y) \coloneqq \mathrm{Had}_{y_C}$ and $\Pi_2(C; y) \coloneqq \mathrm{Had}_{y_C \otimes y_C}$. The *verifier* takes explicit input C and is given access to input oracle y and alleged proof-piece oracles π_1 and π_2. It performs the following tests, and accepts if and only if all of them passed:

1. *Strong codeword checks.* Perform the strong codeword test of Fact 4.5 on both π_1 and π_2.
2. *Oracle-oracle consistency check:* Check that the assignment allegedly encoded by π_2 is the outer product of the assignment allegedly encoded by π_1; that is, uniformly sample $u, v \in \{0,1\}^n$ and check that $\pi_1(u)\pi_1(v) = \pi_2(u \otimes v)$, using self-correction on π_2.
3. *Satisfaction check.* Compute the quadratic equation system (A_C, b_C) corresponding to circuit C via Proposition 4.3, and check that the assignment allegedly encoded by π_2 satisfies the quadratic equation system (A_C, b_C) by checking that it satisfies a random linear combination of equations; that is, uniformly sample $w \in \{0,1\}^n$ and check that $\pi_2(w^\top A_C) = \mathrm{Had}_{b_C}(w)$, using self-correction on π_2. Note that the verifier computes A_C and Had_{b_C} based only on the circuit C, which is given explicitly.
4. *Oracle-witness consistency check.* Check that π_1 encodes the computational extension of y; that is, uniformly sample $i \in [m]$ and check that $y[i] = \pi_1(e_i)$ using self-correction on π_1, where $e_i \in \{0,1\}^m$ is the unit vector $0^{i-1}10^{n-i}$.

It is known that Algorithm 4.6 is a PCPP for CircuitVal with polynomial randomness complexity and constant query complexity (see [2] or later reformulation in [20, Section 4.1]). It is smooth (as a two-piece PCPP), as only a single uniformly random location of y is queried,

and locations queried in π_1 and π_2 are all uniformly random. We show that it is strong canonical, and can therefore be transformed into a single-piece smooth strong canonical PCPP for CircuitVal using Lemma 2.2, thereby proving Proposition 3.6.

Proof. Fix circuit C, input oracle y and proof-piece oracles π_1 and π_2. We show that the verifier rejects with probability at least

$$\frac{5}{72} \cdot \min_{y' \in \mathtt{Sat}(C)} \{\max\left(\delta(y, y'), \delta(\pi_1, \Pi_1(C; y')), \delta(\pi_2, \Pi_2(C; y'))\right)\} \tag{7}$$

where $\mathtt{Sat}(C)$ denotes the set of satisfying inputs of C.

Let $\widetilde{\pi_1}$ and $\widetilde{\pi_2}$ be the Hadamard codewords closest to π_1 and π_2 respectively. Let $\widetilde{z} \in \{0,1\}^n$ and $\widetilde{Z} \in \{0,1\}^{n \times n}$ be the decodings of $\widetilde{\pi_1}$ and $\widetilde{\pi_2}$ respectively, so that $\widetilde{\pi_1} = \mathrm{Had}_{\widetilde{z}}$ and $\widetilde{\pi_2} = \mathrm{Had}_{\widetilde{Z}}$. Fact 4.5 implies that if π_1 or π_2 are 1/12-far from $\widetilde{\pi_1}$ or $\widetilde{\pi_2}$ (respectively) then one of the linearity tests reject with probability at least 1/24. Hence, we may assume $\max\left(\delta(\pi_1, \widetilde{\pi_1}), \delta(\pi_2, \widetilde{\pi_2})\right) \leq 1/12$. Let $\widetilde{y}$ be the n_0-bit-long prefix of $\widetilde{z}$, which corresponds to the values $\widetilde{z}$ gives to the input gates of C.

We start by reckoning that if $\widetilde{\pi_1}, \widetilde{\pi_2}$ aren't the canonical proof-pieces for $(C; \widetilde{y})$ then rejection occurs with high probability, and then show that if they *were* the canonical proof-pieces then the oracle-witness consistency check (resp. strong codeword check) rejects the input oracle (resp. proof-piece oracles) with probability proportional to $\delta(y, \widetilde{y})$ (resp. $\max\left(\delta(\pi_1, \widetilde{\pi 1}), \delta(\pi_2, \widetilde{\pi_2})\right)$).

Observe that $\widetilde{\pi_1}, \widetilde{\pi_2}$ are the canonical proof-pieces of $(C; \widetilde{y})$ if and only if the following conditions hold:

- $\widetilde{y}$ satisfies the circuit C. (Otherwise, $\widetilde{y}$ has no canonical proof.)
- $\widetilde{\pi_1}$ is the Hadamard encoding of the computational extension of $\widetilde{y}$, denoted $\widetilde{y}_C$.
- $\widetilde{\pi_2}$ is the Hadamard encoding of $\widetilde{y}_C \otimes \widetilde{y}_C$.

By Proposition 4.3, we have that $\widetilde{\pi_1}, \widetilde{\pi_2}$ are the canonical proof-pieces of $(C; \widetilde{y})$ if and only if $\widetilde{Z} = \widetilde{z} \otimes \widetilde{z}$ and $\widetilde{z}$ satisfies the equation system (A_C, b_C). With this observation in mind, we examine the following cases.

Case 1: $\widetilde{\pi_1}, \widetilde{\pi_2}$ are not the canonical proof-pieces of $\widetilde{y}$.

We claim that rejection occurs with high probability. Indeed, by the foregoing observation, one of two sub-cases must hold:

Case 1.1: $\widetilde{Z} \neq \widetilde{z} \otimes \widetilde{z}$.

In this case, oracle-oracle consistency check rejects with high probability: The inequation $\widetilde{\pi_1}(u)\widetilde{v} \neq \widetilde{\pi_2}(u \otimes v)$ holds for at least a quarter of possible $u, v \in \{0,1\}^n$,[16] therefore $\pi_1(u)\pi_1(u) \neq \widetilde{\pi_2}(u \otimes v)$ with probability at least $1/4 - 2 \cdot \delta(\pi_1, \widetilde{\pi_1}) \geq 1/12$. The self-corrected query to π_2 gives the value of $\widetilde{\pi_2}$ with probability $1 - 2 \cdot (\delta(\pi_2, \widetilde{\pi_2})) \geq 5/6$, so we have that $\pi_1(u)\pi(v) \neq \pi_2(u \otimes v)$ occurs with probability at least $(1/12) \cdot (5/6) = 5/72$.

Case 1.2: $\widetilde{Z} = \widetilde{z} \otimes \widetilde{z}$ but $\widetilde{z}$ does not satisfy the quadratic equation system (A_C, b_C).

In this case, it is the satisfaction check that rejects with high probability: By Proposition 4.3 it holds that $A_C \widetilde{Z} \neq b_C$, and notice that $\mathrm{Had}_{\widetilde{Z}}(w^\top A_C)(w) = \mathrm{Had}_{A_C \widetilde{Z}}(w)$ for all $w \in \{0,1\}^n$. Therefore, the inequation $\widetilde{\pi_2}(w^\top A_C)(w) \neq \mathrm{Had}_{b_C}(w)$ holds with probability 1/2 over the choice of a uniformly random w, as the Hadamard code has relative distance 1/2. Accounting for a single self-corrected query, the satisfaction test rejects with probability at least $(1/2) \cdot (1 - 2\delta(\pi_2, \widetilde{\pi_2})) \geq 5/12$.

[16] This follows from the fact that for two distinct $n \times n$ matrices A and B, the inequation $u^\top A v \neq u^\top B v$ for at least a quarter of all possible $u, v \in \{0,1\}^n$.

Case 2: $\widetilde{\pi_1}, \widetilde{\pi_2}$ are the canonical proof-pieces of $\widetilde{y}$.

In particular, $\widetilde{y}$ is a satisfying input for circuit C. By definition of the Hamming distance and accounting for self-correction, the oracle-witness test rejects with probability $\delta(y, \widetilde{y})(1 - 2\delta(\pi_1, \widetilde{\pi_1})) \geq 5\delta(y, \widetilde{y})/6$. Additionally the linearity tests reject with probability greater than both $\delta(\pi_1, \widetilde{\pi_1})/2$ and $\delta(\pi_2, \widetilde{\pi_2})/2$. All in all, in this case the verifier rejects with probability at least

$$\max\left(\frac{5}{6}\cdot\delta(y,\widetilde{y}), \frac{1}{2}\cdot\delta(\pi_1,\widetilde{\pi_1}), \frac{1}{2}\cdot\delta(\pi_2,\widetilde{\pi_2})\right) \geq$$
$$\frac{5}{72}\cdot\min_{y'\in\mathtt{Sat}(C)}\{\max\left(\delta(y,y'), \delta(\pi_1, \Pi_1(C;y')), \delta(\pi_2, \Pi_2 C; y')\right)\}$$

where the inequality is justified by the fact that $\widetilde{y}$ satisfies circuit C (however it does not necessarily minimize the expression on the right hand side). ◀

5 The Reed–Muller-based smooth strong canonical robust PCPP

The construction of the Reed–Muller-based RPCPP is more involved than that of the Hadamard-based PCPP. We follow its presentation in [20, Part 1] (a reorganization of [6]), noting that some components mentioned therein appeared in prior works (for example [3, 2]). We find this presentation appealing as it breaks the construction down to four bite-sized steps. Essentially, we prove that the first step yields a PCPP that is smooth and strong canonical, and that the transition between each step preserves smoothness and strong canonicality. Along the way, we will also keep track of the residual circuit distance (see Definition 3.4) and observe that it is constant.

So how do we go about proving this? First, it's worth noting that the intermediate PCPPs are multi-piece and, more importantly, issue queries to oracles of larger, non-binary alphabets. Don't worry, eventually (Section 5.4) we show how to reduce alphabet size back to binary, and of course our old friend Lemma 2.2 will reduce the number of pieces to one.[17] Anyways, the first step is constructing a smooth and strong canonical RPCPP for the algebraic problem of determining whether a function is a low-degree polynomial that is identically zero on a subcube (Section 5.1). This problem is closely related to circuit satisfiability: fixing a circuit, inputs can be encoded in a way such that the input satisfies the circuit if and only if its encoding is Zero on Subcube. Section 5.2 capitalizes on this to derive an RPCPP for the problem of circuit satisfiability, which is then transformed to an RPCPP for CircuitVal by adding a consistency test (Section 5.3).

PCPPs over larger alphabets

Though the final result of this section is a (Boolean) smooth and strong canonical RPCPP, the PCPPs used along the way are non-Boolean, meaning that their queries are answered not with bits but with elements of an alphabet of larger size. Furthermore, the intermediate multi-piece PCPPs may have different alphabets for different pieces.

So far, we did not explicitly mention which alphabet was used (indeed, it was always binary) so the previously introduced notions and definitions need not be changed, except for replacing *relative Hamming distance* with the *generalized relative Hamming distance*.

[17] As in Section 3, to transform the multi-piece robust PCPP to a single-piece robust PCPP we note that Lemma 2.2 holds for robust PCPPs as well: for strong canonicality, rather than analyzing the *rejection probability of the verifier*, we can analyze the *expected distance from being accepted by the verifier* (using exactly the same reasoning exactly).

▶ **Definition 5.1** (Generalized relative Hamming metric). *Fix sets $\Sigma_1, \ldots, \Sigma_k$ and let $x = (x_1, \ldots, x_k)$ and $y = (y_1, \ldots, y_k)$ such that $x_i, y_i \in \Sigma_i$ for all $i \in [k]$. The* generalized (relative) Hamming distance *between x and y is the fraction of locations in which x and y differ, that is $\delta(x, y) \coloneqq \mathbb{P}_i [x_i \neq y_i]$ for a uniformly random $i \in [k]$.*

Note that some of the sets $\Sigma_1, \ldots, \Sigma_k$ may not be distinct. In particular, if $\Sigma_1 = \cdots = \Sigma_k$, then the generalized relative Hamming distance coincides with the relative Hamming distance as defined way back in Section 1.1.1.

▶ Remark 5.2. A letter in the alphabet Σ_i can be represented by $\log |\Sigma_i|$ bits. However, the size of Σ_i is (deliberately) not accounted for in the generalized Hamming distance, therefore the generalized Hamming distance between x and y is not necessarily the same as the Hamming distance between the binary representation of x and y (in which x_i and y_i are each replaced with $\log |\Sigma_i|$ bits for each $i \in [k]$).

All distances referred to in Section 5 are in the generalized Hamming metric.

5.1 A smooth strong canonical RPCPP for Zero on Subcube

In this section we follow [20, Section 5.3.2] in constructing an RPCPP for the algebraic problem of determining whether a function is (close to) a low-degree polynomial that is identically zero on a subcube (formalized in Definition 5.9). Before we turn to the main construction, we present a generalization of a property tester for low-degree polynomials to sequences (vectors) of functions, which will be used in the RPCPP construction. Specifically, given oracle access to a function $f : \mathbb{F}^m \to \mathbb{F}^k$, we test whether it is close to a sequence (vector) of k low-degree polynomials (see Definition 5.3 quoted below).

▶ **Definition 5.3** (Vector-valued low-degree polynomial). *A function $f \colon \mathbb{F}^m \to \mathbb{F}^k$ is a* vector-valued multivariate polynomial of degree at most d *if for all $i \in [k]$ the projection of f to the ith coordinate is a multivariate polynomial of (total) degree at most d, where the projection of f to the ith coordinate is denoted $f_i : \mathbb{F}^m \to \mathbb{F}$ and defined by $f_i(x) \coloneqq f(x)_i$ for all $x \in \mathbb{F}^m$.*

The distance between such vector-valued functions is measured according to the generalized Hamming metric of Definition 5.1; that is, $f : \mathbb{F}^m \to \mathbb{F}^k$ is δ-far from being a low-degree vector-valued polynomial if $\mathbb{P}_x [f(x) \neq \widetilde{f}(x)] > \delta$ for any vector-valued polynomial $\widetilde{f}$ of degree at most d. We stress that $f(x) \neq \widetilde{f}(x)$ when there *exists* i such that $f_i(x) \neq \widetilde{f}_i(x)$.

Recall that the *line* $\mathcal{L}$ through $\mathbb{F}^m$ with *intercept* $x \in \mathbb{F}^m$ and *slope* $h \in \mathbb{F}^m$ is the set of points $\mathcal{L} \coloneqq \{x + ih : i \in \mathbb{F}\}$. A uniformly random line (through $\mathbb{F}^m$) is obtained by sampling $x, h \in \mathbb{F}^m$ uniformly at random and letting $\mathcal{L}$ be the line with slope h and intercept x. Throughout this section we will use $f[\mathcal{L}]$ to denote the restriction of f to the line $\mathcal{L}$.

▶ **Algorithm 5.4** (PL-VLDT). The *point-line vector-valued low-degree test (PL-VLDT)* is given access to an oracle $f \colon \mathbb{F}^m \to \mathbb{F}^k$ and a "lines" proof oracle g that maps line $\mathcal{L}$ to vector-valued univariate polynomial $g_{\mathcal{L}} : \mathcal{L} \to \mathbb{F}^k$ of degree at most d. It samples a uniformly random line $\mathcal{L}$ through $\mathbb{F}^m$ and uniformly random point $x \in \mathcal{L}$, and accepts if and only if $f(x) = g_{\mathcal{L}}(x)$.

PL-VLDT uses $2m \log |F|$ random coins and makes a single query to each of its two oracles. Its soundness proof is due to Oded Goldreich and Madhu Sudan and can be found in Appendix C. We quote the relevant proposition:

▶ **Proposition 5.5** (Proposition C.4). *Assuming $|\mathbb{F}| > 25k$, if the input oracle $f \colon \mathbb{F}^m \to \mathbb{F}^k$ is δ-far from being a vector-valued polynomial of degree at most d then for any lines oracle g, PL-VLDT rejects f and g with probability at least $\delta/40$.*

We will use a variant of PL-VLDT that has higher query complexity, but does not require an auxiliary "lines" oracle; instead, it queries f on an entire line. As shown in Proposition 5.7, this test has robust soundness, meaning that the expected distance of answers to its queries from agreeing with a univariate low-degree polynomial (and therefore being accepted by the test) is proportional to the distance of the input function from being low-degree.

▶ **Algorithm 5.6** (VLDT). The *vector-valued low-degree test* is given explicit inputs $\mathbb{F}$, m and d and access to an input oracle $f\colon \mathbb{F}^m \to \mathbb{F}^k$. It samples a uniformly random line $\mathcal{L}$ through $\mathbb{F}^m$, queries f on all points in $\mathcal{L}$, and accepts if and only if the obtained values (describing the restriction of f to $\mathcal{L}$) agree with a univariate vector-valued polynomial of degree at most d.

Algorithm 5.6 uses $2m \log |\mathbb{F}|$ random coins and makes $|\mathbb{F}|$ queries to its input. Its robust soundness is asserted in Proposition 5.7.

▶ **Proposition 5.7.** *Assuming $|\mathbb{F}| > 25k$, if the input f is δ-far from being a vector-valued polynomial of degree at most d, then the expected distance of answers to the queries of VLDT from agreeing with a univariate low-degree polynomial (and therefore being accepted by the test) is at least $\beta \cdot \delta$, for $\beta = 1/40$.*

Proof. Given a function f, we define a lines oracle g that assigns each line $\mathcal{L}$ the (univariate, vector-valued) low-degree polynomial $g_{\mathcal{L}}$ closest to $f[\mathcal{L}]$. By definition, when line $\mathcal{L}$ is sampled by VLDT, the distance of answers received from f (namely $f[\mathcal{L}]$) from being accepted is $\delta(f[\mathcal{L}], g_{\mathcal{L}})$. On the other hand, PL-VLDT rejects input oracle f and the lines oracle $g \coloneqq (g_{\mathcal{L}})_{\mathcal{L}}$ with probability $\mathbb{E}_{\mathcal{L}}\left[\mathbb{P}_{x\in\mathcal{L}}\left[f(x) \neq g_{\mathcal{L}}(x)\right]\right] = \mathbb{E}_{\mathcal{L}}\left[\delta(f[\mathcal{L}], g_{\mathcal{L}})\right]$ which, invoking Proposition 5.5, is at least $\delta/40$. Therefore, the *expected* distance of the answers that VLDT receives from being accepted is at least $\delta/40$. ◀

Finally, we recall a basic and important property of low-degree polynomials.

▶ **Fact 5.8** (The Schwartz-Zippel Lemma). *For any finite field $\mathbb{F}$ and integers m and d, if $P\colon \mathbb{F}^m \to \mathbb{F}$ is a nonzero polynomial of degree at most d, then $\mathbb{P}_x\left[P(x) = 0\right] \leq d/|\mathbb{F}|$ for $x \in \mathbb{F}^m$ sampled uniformly at random.*

With these tools in hand, the construction can commence. We analyze the smoothness and strongness of a robust PCPP for Zero on Subcube [20, Section 5.3.2], an algebraic analogue of CircuitVal, starting with a formal definition of this property.

▶ **Definition 5.9** (ZoS). *Fix a finite field $\mathbb{F}$, positive integers m and d and set $H \subseteq \mathbb{F}$. A degree d polynomial $f\colon \mathbb{F}^m \to \mathbb{F}$ is* Zero on the Subcube H^m *(*Zero on Subcube *for short) if its restriction to H^m is identically 0. We denote the set of all Zero on the Subcube polynomials by $\text{ZoS}(\mathbb{F}, m, d, H)$. When the parameters are obvious from context, the shorthand* ZoS *is used instead.*

We quote a useful characterization of Zero on Subcube polynomials that gives rise to a natural robust PCPP for the set ZoS.

▶ **Fact 5.10** ([20, Proposition 5.3.4]). *For any field $\mathbb{F}$, positive integers m and d and set $H \subseteq \mathbb{F}$, a polynomial $f\colon \mathbb{F}^m \to \mathbb{F}$ of degree at most d is zero on the subcube H^m if and only if there exists a sequence of polynomials $P_1, \ldots, P_m : \mathbb{F}^m \to \mathbb{F}$ each of degree at most d, and a sequence of polynomials $Q_1, \ldots, Q_m : \mathbb{F}^m \to \mathbb{F}$ each of degree at most $d - |H|$, such that for all $x_1, \ldots, x_m \in \mathbb{F}$ and $i \in [m]$:*

$$\begin{aligned} P_{i-1}(x_1, \ldots, x_m) &= \eta(x_i) \cdot Q_i(x_1, \ldots, x_m) + P_i(x_1, \ldots, x_m) \\ P_m(x_1, \ldots, x_m) &= 0 \end{aligned} \tag{8}$$

where $P_0 \coloneqq f$, and η is a univariate polynomial of degree $|H|$ that vanishes on H which we define by $\eta(x) \coloneqq \prod_{h\in H}(x-h)$.

The vector-valued polynomials $P = (P_1, \ldots, P_m)$ and $Q = (Q_1, \ldots, Q_m)$ are called the division witnesses *of f.*[18]

▶ **Algorithm 5.11** (ZoS-RPCPP)**.** The canonical proof-pieces for $f \in \text{ZoS}$ are the division witnesses of f, denoted $P(f) \coloneqq (P_1(f), \ldots, P_m(f))$ and $Q(f) \coloneqq (Q_1(f), \ldots, Q_m(f))$, as guaranteed by Fact 5.10. For input field $\mathbb{F}$, dimension m and degree d, given oracle access to input $f\colon \mathbb{F}^m \to \mathbb{F}$ and proof-pieces $p : \mathbb{F}^m \to \mathbb{F}^m$ and $q : \mathbb{F}^m \to \mathbb{F}^m$, the verifier samples a random line $\mathcal{L}$ through $\mathbb{F}^m$, queries f, p and q on each point in $\mathcal{L}$, and performs the following checks:

1. *Low-degree checks.* Check that the restrictions of f and p to the line $\mathcal{L}$, denoted $f[\mathcal{L}]$ and $p[\mathcal{L}]$, are vector-valued univariate polynomials of degree at most d.[19] Check that $q[\mathcal{L}]$ is a vector-valued univariate polynomial of degree at most $d - |H|$.
2. "Unbundle" the answers received from p and q to obtain $p_i(x)$ and $q_i(x)$ for each $i \in [m]$ and $x \in \mathcal{L}$. Check the following:
   a. *Division checks.* For each $i \in [m]$ and $(x_1, \ldots, x_m) \in \mathcal{L}$, check that
   $$p_{i-1}(x_1, \ldots, x_m) = \eta(x_i) \cdot q_i(x_1, \ldots, x_m) + p_i(x_1, \ldots, x_m)$$
   where $p_0 \coloneqq f$, and η is a univariate polynomial of degree $|H|$ that vanishes on H, defined $\eta(x) \coloneqq \prod_{h\in H}(x-h)$.
   b. *Identity check.* For each $(x_1, \ldots, x_m) \in \mathcal{L}$, check that
   $$p_m(x_1, \ldots, x_m) = 0$$

The verifier accepts if and only if all the above checks passed.

ZoS-RPCPP is piecwise-smooth, makes $|\mathbb{F}|$ queries to each of its three oracles and uses $2m \log |\mathbb{F}|$ random coins. Its residual circuit distance (Definition 3.4) is $1/2$ as any satisfying input to its residual circuit is composed of three univariate polynomials of degree at most d, each occupying a third of the input. As such, any two different satisfying inputs agree on at most a $2/3 \cdot d/|\mathbb{F}| \le 1/2$ fraction of locations.

▶ **Lemma 5.12.** *Suppose* ZoS-RPCPP *is given field $\mathbb{F}$ and degree d such that $1 - \beta \ge 4d/|\mathbb{F}|$, and access to oracles f, p and q. Then the expected distance (over a random line) of the answers to* ZoS-RPCPP*'s queries from being accepted is at least*

$$\frac{\beta}{12} \cdot \min_{f' \in \text{ZoS}(\mathbb{F},m,d,H)} \{\max\left(\delta(f,f'), \delta(p, P(f')), \delta(q, Q(f'))\right)\} \tag{9}$$

where β is the constant of Proposition 5.7, and $P(f')$ and $Q(f')$ are the canonical proof-pieces of f' as described in Algorithm 5.11.

Proof. Fix input oracle $p_0 \coloneqq f\colon \mathbb{F}^m \to \mathbb{F}$ and proof oracles $p = (p_1, \ldots, p_m)$ and $q = (q_1, \ldots, q_m)$ with $p_i, q_i : \mathbb{F}^m \to \mathbb{F}$. Let $\widetilde{f}$ and $\widetilde{p}$ be the vector-valued polynomials of degree at most d closest to f and p, and $\widetilde{q}$ be the polynomial of degree at most $d - |H|$ closest to q. Let $\widetilde{p_i}$ and $\widetilde{q_i}$ be the projections of p and q to their ith coordinate. Note that $p_0 \coloneqq f$ and so $\widetilde{p_0} \coloneqq \widetilde{f}$.

We start by showing that unless $\widetilde{f}$ is Zero on Subcube and $\widetilde{p}, \widetilde{q}$ are its canonical proof-pieces, then the expected distance is $\Omega(1)$.

[18] This ad-hoc term alludes to the proof of Fact 5.10: an iterative process that starts with the polynomial $f = P_0$, and in the ith iteration divides P_{i-1} by $\eta(x_i)$ to obtain quotient Q_i and remainder P_i.
[19] Indeed the term "vector-valued" is degenerate for $f\colon \mathbb{F}^m \to \mathbb{F}$ as its range is one-dimensional.

Case 1: Either f, p or q are $1/4$-far from $\widetilde{f}$, $\widetilde{p}$ or $\widetilde{q}$. Assume wlog that p is $1/4$-far from $\widetilde{p}$. By Proposition 5.7, the expected distance of $p[\mathcal{L}]$ from a univariate polynomial of degree at most d is at least $\beta/4$. However the answers of p are just a third of the answers received by all oracles, so the expected distance of $(f[\mathcal{L}], p[\mathcal{L}], q[\mathcal{L}])$ from satisfying the low-degree checks is at least $\frac{1}{3} \cdot \frac{\beta}{4}$.

Case 2: f, p and q are close to low-degree polynomials that do not satisfy Equation (8); that is, f, p and q are $1/4$-close to $\widetilde{f}$, $\widetilde{p}$ and $\widetilde{q}$ but for some i it holds that $\widetilde{p_{i-1}}$, $\widetilde{p_i}$ and $\widetilde{q_i}$ don't satisfy Equation (8). By Fact 5.8, either $\widetilde{p_{i-1}}$, $\widetilde{p_i}$ or $\widetilde{q_i}$ do not satisfy Equation (8) on at least a fraction of $1 - d/|F|$ of points in $\mathbb{F}^m$. Since p_{i-1}, p_i and q_i are $1/4$-close to $\widetilde{p_{i-1}}$, $\widetilde{p_i}$ and $\widetilde{q_i}$ respectively,[20] then p_{i-1}, p_i and q_i do not satisfy Equation (8) on at least a fraction of $1 - d/|\mathbb{F}| - 3/4 = 1/4 - d/|\mathbb{F}|$ points in $\mathbb{F}^m$. Therefore the expected distance of $(f[\mathcal{L}], p[\mathcal{L}], q[\mathcal{L}])$ from satisfying the division checks is at least $1/3 \cdot (1/4 - d/|\mathbb{F}|) \geq \beta/12$.

Case 3: p_m is $1/4$-close to $\widetilde{p_m}$ but $\widetilde{p_m} \not\equiv 0$. By Fact 5.8, $p_m(x) \neq 0$ on at least a fraction of $1/4 - d/|\mathbb{F}|$ of points $x \in \mathbb{F}^m$. As the answers of p are a third of the answers received by all oracles, the expected distance of $(f[\mathcal{L}], p[\mathcal{L}], q[\mathcal{L}])$ from satisfying the identity check is at least $1/3 \cdot (1/4 - d/|\mathbb{F}|) \geq \beta/12$.

If all three previous cases are are false, then the input oracle is close to a Zero on Subcube polynomial, and the given proof-piece oracles are close to the canonical proof-pieces for that polynomial. These proof-pieces are low-degree polynomials, therefore the distance of the given oracles from proving that the input is Zero on Subcube is exactly their distance from being low-degree polynomials (similarly, the distance of the input from being Zero on Subcube is its distance from being low-degree), and VLDT rejects with probability proportional to this distance. Details follow.

As we said, in this case $\widetilde{f}$ is zero on the subcube H^m, and $\widetilde{p}, \widetilde{q}$ are its canonical proof-pieces. Since $\delta(f, \widetilde{f}) < 1/4$ then $\widetilde{f}$ is the only low-degree polynomial $1/4$-close to f, and since $\widetilde{f} \in \text{ZoS}$ it must be that $\widetilde{f}$ is function from $\text{ZoS}(\mathbb{F}, m, d, H)$ closest to f. We conclude the proof by showing that the expected distance of all answers from satisfying is greater than Expression 9: Assume that $\delta(q, \widetilde{q})$ maximizes Expression 9. The expected distance of answers to queries made by VLDT to oracle q from satisfying is at least $\beta \cdot \delta(q, \widetilde{q})$. But as the answers of q account for a third of the all answers received, we have that the expected distance of all answers on a random line from satisfying is at least $\beta \cdot \delta(q, \widetilde{q})/3 = \beta \cdot \delta(q, Q(\widetilde{f}))/3$. If f or p maximized Expression 9 then similar arguing gives the required result. ◀

5.2 A smooth strong canonical RPCP for CircuitSat

To construct a robust PCP for CircuitSat, we first recall an algebrization (i.e. algebraic description) of circuits and their inputs such that an input satisfies the circuit if and only if an encoding of the input is zero on a certain (fixed) subcube.

▶ **Definition 5.13** (Low-degree extension). *Let $z\colon H^m \to \mathbb{F}$. The* low-degree extension of z to $\mathbb{F}^m$ *is the unique polynomial $\widehat{z} : \mathbb{F}^m \to \mathbb{F}$ of degree at most $m|H|$ that agrees with z on H^m.*[21]

[20] Indeed, notice that $\delta(p_i, \widetilde{p_i}) \leq \delta(p, \widetilde{p})$ for all $i \in [m]$ and similarly for q and $\widetilde{q}$.

[21] We will only use the uniqueness and low-degree properties of the extension, but if you insist, know that it is given by

$$\widehat{z}(x) \coloneqq \sum_{h \in H^m} z(h) \cdot \prod_{\substack{i \in [m] \\ h' \in H \setminus \{h_i\}}} \frac{x_i - h'}{h_i - h'}$$

We loosely summarize the algebrization of [20, Section 5.4.1]. A circuit C of size n is associated with a function $C'\colon [n]^3 \times \{0,1\}^3 \to \{0,1\}$ such that $C'(i_1,i_2,i_3,b_1,b_2,b_3) = 1$ if and only if assigning gates i_1, i_2 and i_3 the values $\neg b_1$, $\neg b_2$ and $\neg b_3$ necessarily results in an *invalid* or *unsatisfying* computation of the circuit. The interested reader is referred to [20] for a formal definition of these,[22] and we will describe them by example: if gate 5 is the output gate of C then $C'(5,1,2,1,0,0) = 1$ because (by definition) a satisfying computation does not have the output gate outputting the value 0. As another example, suppose that the 7th gate is an `OR` gate taking input from gates 2 and 4, then $C'(2,4,7,1,1,0) = 1$ because in a valid computation an `OR` gate taking two 0 inputs cannot output 1.

So, C' indicates when an assignment to C's gates results in an invalid or unsatisfying computation. Thus for any alleged computational extension z (see Definition 4.1), the product $F' = C'(i_1,i_2,i_3,b_1,b_2,b_3)\cdot(z[i_1]-b_1)(z[i_2]-b_2)(z[i_3]-b_3)$ will be zero if and only if z is a computational extension of a satisfying input. Another key observation is that replacing C' by its low-degree extension $\widehat{C}$ and z by its low-degree extension $\widehat{z}$ results in a product F that has low-degree. Conversely, F is Zero on Subcube (and in particular low-degree) only if z was a satisfying assignment.

We refer the reader to [20, Section 5.4.1] for a deeper discussion, which is summarized in the following statement.

▶ **Fact 5.14** (Algebrization of CircuitSat)**.** *There exists a polynomial reduction that maps a circuit C of size n to a polynomial $\widehat{C}\colon \mathbb{F}^{3m+3} \to \mathbb{F}$ of degree at most d where $m := \log n/\log\log n$, $H := [n^{1/m}]$, $d = (3m+3)|H|$ and $|\mathbb{F}| = O((d+3m|H|)^3)$, such that $\widehat{C}$ satisfies the following:*

- *For any $e\colon \mathbb{F}^m \to \mathbb{F}$, let $F_{C,e} : \mathbb{F}^{3m+3} \to \mathbb{F}$ be given by*

$$\begin{aligned} F_{C;e}(x_1,\ldots,x_{3m+3}) :=& \widehat{C}(x_1,\ldots,x_{3m+3})\cdot(e(x_1,\ldots,x_m)-x_{3m+1}) \\ &\cdot(e(x_{m+1},\ldots,x_{2m})-x_{3m+2})\cdot(e(x_{2m+1},\ldots,x_{3m})-x_{3m+3}) \end{aligned}$$

  Then, for any polynomial $e\colon \mathbb{F}^m \to \mathbb{F}$ of degree at most $m|H|$, the function $F_{C;e}$ is identically zero on H^{3m+3} if and only if e is the low-degree extension of a computational extension of an assignment that satisfies C.

▶ **Algorithm 5.15** (CircuitSat-RPCP)**.** For any circuit C and satisfying input y the canonical proof consists of four pieces: the low-degree extension of the computational extension of y denoted $\widehat{y_C}$, the polynomial $F_{C;\widehat{y_C}}$ as defined in Fact 5.14, followed by the proof-pieces (for ZoS-RPCPP) that $F_{C;\widehat{y_C}}$ is zero on the subcube H^{3m+3}, namely its division witnesses (of Fact 5.10) denoted $P(C;y)$ and $Q(C;y)$. Given explicit access to circuit C of size n, and oracle access to $e\colon \mathbb{F}^m \to \mathbb{F}$, $f : \mathbb{F}^{m'} \to \mathbb{F}$, $p : \mathbb{F}^{m'} \to \mathbb{F}^{m'}$ and $q : \mathbb{F}^{m'} \to \mathbb{F}^{m'}$ where $|\mathbb{F}| = O((d+1m|H|)^3)$ and $m' := 3m+3$, the verifier samples a uniformly random line $\mathcal{L}'$ through $\mathbb{F}^{m'}$ and performs the following checks:

1. *Zero on Subcube check.* Check that f is a polynomial of degree at most d' that is zero on the subcube $H^{m'}$ using ZoS-RPCPP with the random line $\mathcal{L}'$, p and q as its proof-piece oracles, and with $d' := m'|H|$ and $H = [n^{1/m}]$.
2. *Low-degree check.* Check that e is a polynomial of degree at most d using VLDT where $d = m|H|$, with the projection of $\mathcal{L}'$ onto its first m coordinates used as the random line. That is, check that $\{e(x_1,\ldots,x_m) : (x_1,\ldots,x_m,\ldots,x_{m'}) \in \mathcal{L}'\}$ agrees with a univariate polynomial of degree at most d.

[22] In [20] these notions are both referred to as *invalid configurations*.

3. *Satisfaction check.* For each $x = (x_1, \ldots, x_{m'}) \in \mathcal{L}'$, check that

$$f(x_1, \ldots, x_{3m+3}) \coloneqq \widehat{C}(x_1, \ldots, x_{3m+3}) \cdot (e(x_1, \ldots, x_m) - x_{3m+1}) \cdot (e(x_{m+1}, \ldots, x_{2m}) - x_{3m+2}) \cdot (e(x_{2m+1}, \ldots, x_{3m}) - x_{3m+3}) \tag{10}$$

where $\widehat{C}$ is as guaranteed by the algebrization of CircuitSat (Fact 5.14).
The verifier accepts if and only if all the above checks passed.

CircuitSat-RPCP tosses $m' \log |\mathbb{F}| = O(m \log |\mathbb{F}|)$ random coins. It issues a total of $6|\mathbb{F}|$ queries to all of its oracles. Namely, for each $x \in \mathcal{L}'$ it issues the following queries:

$$f(x), p(x), q(x), e(x_1, \ldots, x_m), e(x_{m+1}, \ldots, x_{2m}), e(x_{2m+1}, \ldots, x_{3m})$$

It has constant residual circuit distance (Definition 3.4), as any answer string that satisfies its decision circuit consists of 6 univariate low-degree polynomials, each composing a sixth of the entire string. Thus, two different strings satisfying its decision circuit agree on at most a $\frac{5}{6} \cdot \frac{d}{|\mathbb{F}|} \leq 1/2$ fraction of locations.

▶ **Lemma 5.16.** *Suppose* CircuitSat-RPCP *is given satisfiable circuit C and access to proof-piece oracles $e\colon \mathbb{F}^m \to \mathbb{F}$, $f : \mathbb{F}^{m'} \to \mathbb{F}$, and $p, q : \mathbb{F}^{m'} \to \mathbb{F}^{m'}$, with $1 - \beta \geq 5d'/|\mathbb{F}|$ where β is the constant of Proposition 5.7. Then the expected distance of the answers to* CircuitSat-RPCP*'s queries from satisfying is at least*

$$\frac{\beta}{30} \cdot \min_{y' \in \mathtt{Sat}(C)} \left\{ \max \left(\delta\left(e, \widehat{y'_C}\right), \delta\left(f, F_{C;\widehat{y'_C}}\right), \delta(p, P(C; y')), \delta(q, Q(C; y')) \right) \right\} \tag{11}$$

where $\mathtt{Sat}(C)$ denotes the set of satisfying inputs of C, and $\widehat{y'_C}$, $F_{C;\widehat{y'_C}}$, $P(C; y')$ and $Q(C; y')$ are the canonical proof-pieces of y' as described in Algorithm 5.15.

Proof. Fix circuit C and oracles e, f, p and q. Let $\widetilde{e}$ be the closest polynomial to e of degree at most d, and let $\widetilde{f}$, $\widetilde{p}$, $\widetilde{q}$, $\widetilde{p_i}$ and $\widetilde{q_i}$ be the closest low-degree polynomials (as in Lemma 5.12). Consider the following cases:

Case 1: Either of the oracles are 1/5-far from their closest low-degree polynomial. By Proposition 5.7, at least $\beta/5$ of the points read on a random line must be changed in order to satisfy VLDT. The answers to queries made by the low-degree checks (either of Step 2 or Step 1) make up for at least a sixth of the all answers received, therefore the expected distance of all received answers from satisfying is at least $\beta/30$.

Case 2: All oracles are 1/5-close to low-degree polynomials, but $\widetilde{p}, \widetilde{q}$ are not the canonical proof-pieces (for ZoS-RPCPP) that $\widetilde{f}$ is Zero on Subcube. Just like in the proof of Lemma 5.12, the answers to queries made in Step 1 are $\beta/5$-far from satisfying. These answers are a half of answers to all queries, therefore the expected distance of all received answers from satisfying is at least $\beta/10$.

Case 3: All oracles are 1/5-close to low-degree polynomials, but $\widetilde{e}$ and $\widetilde{f}$ do not satisfy Equation (10). Then the expected number of points (on a random line) on which e and f violate Equation (10) is at least $1 - d'/\mathbb{F} - 4 \cdot 1/5 = 1/5 - d'/\mathbb{F}$, which is larger than $\beta/5$. Since the answers to queries made in Step 3 make up for at least two-thirds of answers to all queries, the expected distance of all answers from satisfying is at least $2\beta/15$.

If all of the above are false, then the given oracles are close to proof-pieces proving that C is satisfiable. These proof-pieces are low-degree polynomials, therefore the distance of the given oracles from proving that C is satisfiable is exactly their distance from being low-degree, and VLDT rejects with probability proportional to this distance. Details follow.

In this case, $\widetilde{e}$ is a low-degree extension of a computational extension of a satisfying assignment to C, and $\widetilde{f}$, $\widetilde{p}$ and $\widetilde{q}$ are its canonical proof-pieces. Since $\delta(e,\widetilde{e}) < 1/5$ then $\widetilde{e}$ is the only low-degree polynomial $1/5$-close to e, so it must be that $\widetilde{e} = \widehat{y'_C}$ (where y' is the minimizer of Expression 11). Assume that q maximizes Expression 11. The expected distance of answers to queries made by VLDT to q from satisfying is at least $\beta \cdot \delta(q,\widetilde{q})$. But as the answers of q account for a sixth of all answers received, the expected distance of all answers from satisfying is at least $\beta\delta(q,\widetilde{q})/6$. If the maximizers were e, f or p the claim follows using similar reasoning. ◀

5.3 A smooth strong canonical RPCPP for CircuitVal

In this section we transform CircuitSat-RPCP to an RPCP *of proximity* for CircuitVal. The RPCPP is constructed by (additionally) testing consistency of the input oracle (an allegedly satisfying input to the circuit) with the proof-piece oracle corresponding to the alleged low-degree extension of its computational extension (see Definition 4.1). Low degree extension is systematic, in the sense that the extended function is embedded in its extension, so consistency may be checked by testing that the input oracle agrees with the systematic part of its alleged low-degree extension on a uniformly random location. However, the systematic part of the proof-piece oracle is only a tiny fraction of it (as $n = o(|\mathbb{F}^m|)$), so a particularly nasty proof-piece oracle could be close to an arbitrary low-degree polynomial (therefore passing the low-degree check), whose systematic part was changed to match the input oracle (therefore passing the consistency check) – causing the verifier to accept a proof that's extremely far from the canonical one. This is resolved via self-correction: the verifier checks that the proof-piece oracle is a low-degree univariate polynomial on a random (punctured, see below) line through the location queried in the systematic part.

Unlike previous steps in this construction, some care must be taken so that the consistency check does not harm smoothness. If the verifier reads the entire line, locations in the systematic part would be queried more often than others. Indeed, the verifier does not need to read the systematic point on the line and can interpolate it from the others. But if the verifier *always* avoids reading the systematic part, its locations would be queried less often than others. Thus, the verifier chooses the puncture (i.e. point on the line not to be read) to be the location in the systematic part with probability $1 - \Theta(1/|F|^{m-1})$, and to be a different point on the line with the remaining probability.

▶ **Algorithm 5.17** (CircuitVal-RPCPP)**.** The canonical proof of a circuit and its satisfying input consists of the same four proof-pieces as in Algorithm 5.15. Given explicit access to circuit C of size n with n_0 input gates, and oracle access to input $y\colon [n_0] \to \{0,1\}$ and proof-pieces $e : \mathbb{F}^m \to \mathbb{F}$, $f : \mathbb{F}^{m'} \to \mathbb{F}$, $p : \mathbb{F}^{m'} \to \mathbb{F}^{m'}$ and $q : \mathbb{F}^{m'} \to \mathbb{F}^{m'}$ where $m' \coloneqq 3m+3$, the verifier performs the following checks:

1. *Satisfiability check.* Check that C is satisfiable using CircuitSat-RPCP with e, f, p and q as proof-piece oracles.
2. *Consistency check.* Sample a location in the input oracle by uniformly sampling $x \in [n_0] \equiv H^m$. Sample a random line $\mathcal{L}$ with intercept x by sampling $h \in \mathbb{F}^m$ and letting $\mathcal{L} \coloneqq (x + ih : i \in \mathbb{F})$. Let the puncture z be $z \coloneqq x$ with probability $1 - (|\mathbb{F}| - 1)/(|\mathbb{F}|^m + 1)$, and $z \coloneqq x + h$ with the remaining probability.
   Query e on the line $\mathcal{L}$ punctured at z, i.e. $e[\mathcal{L} \setminus \{z\}]$. Query $y(x)$, giving the query a weight of $|\mathcal{L}| = |\mathbb{F}|$.[23]. Do the following:

[23] Reweighting is achieved by repeating the input gate corresponding to $y(x)$ in the verifier's residual circuit.

a. Check that $e[\mathcal{L} \setminus \{x\}]$ agrees with a univariate polynomial of degree at most d. If the check passed, let $e_{\mathcal{L}} : \mathcal{L} \to \mathbb{F}$ be the unique low-degree polynomial that agrees with $e[\mathcal{L} \setminus \{z\}]$.
b. Check that $y(x) = e_{\mathcal{L}}(x)$: If $z = x$, obtain $e_{\mathcal{L}}(x)$ using interpolation (*not* by querying $e(x)$). If $z = x + h$ then $e_{\mathcal{L}}(x) = e(x)$ has already been queried and is known to the verifier.

The verifier accepts if and only if the above checks passed.

CircuitVal-RPCPP tosses $(m + m') \log |\mathbb{F}| = O(m \log |\mathbb{F}|)$ random coins. It issues a total of $8|\mathbb{F}| - 1$ (weighted) queries to its oracles: in addition to the $6|\mathbb{F}|$ queries of CircuitSat-RPCP, it queries e on $\mathcal{L} \setminus \{z\}$ in Step 2a, and issues a query of weight $|\mathbb{F}|$ to w. It has constant residual circuit distance (Definition 3.4), which is shown by an application of Fact 5.8, just as with ZoS-RPCPP and CircuitSat-RPCP.

Smoothness of CircuitVal-RPCPP

We explain how the randomized choice of puncture in Step 2 preserves smoothness. Let α denote the probability that the puncture z is at x (i.e., verifier reads $e[\mathcal{L} \setminus \{x\}]$), and $1 - \alpha$ be the probability that the puncture is at $x + h$. We show that each point in $\mathbb{F}^m$ is queried with equal probability.

- *The non-systematic part:* A point from the non-systematic part, i.e. $\mathbb{F}^m \setminus [n_0]$, is read only if it lies on the line $\mathcal{L}$ and does not equal the puncture. This occurs with probability

$$\frac{|\mathbb{F}| - 2}{|\mathbb{F}|^m} + \frac{1}{|\mathbb{F}|^m} \cdot \alpha \tag{12}$$

- *The systematic part:* Consider a point from the systematic part, i.e. $[n_0]$. If it is chosen to be the intercept x of line $\mathcal{L}$, then it is read only if the puncture is not at x but at $x + h$ – which occurs with probability $1 - \alpha$. If it is not chosen to be the intercept x, then it is queried with probability equal to that of points in the non-systematic part, as analyzed above. Thus, points in the systematic part are queried with probability

$$\frac{1}{n_0} \cdot (1 - \alpha) + \left(1 - \frac{1}{n_0}\right) \cdot \left(\frac{|\mathbb{F}| - 2}{|\mathbb{F}|^m} + \frac{1}{|\mathbb{F}|^m} \cdot \alpha\right) \tag{13}$$

Indeed, choosing $\alpha = 1 - (|F| - 1)/(|\mathbb{F}|^m + 1)$ equalizes Expressions 12 and 13.

Soundness of CircuitVal-RPCPP

Robust strong canonical soundness of CircuitVal-RPCPP follows from the following lemma.

▶ **Lemma 5.18.** *Suppose* CircuitVal-RPCPP *is given a satisfiable circuit C, input oracle $y \colon [n_0] \to \{0,1\}$ and proof-piece oracles $e : \mathbb{F}^m \to \mathbb{F}$, $f : \mathbb{F}^{m'} \to \mathbb{F}$, and $p, q : \mathbb{F}^{m'} \to \mathbb{F}^{m'}$, with $1 - \beta \geq d/|\mathbb{F}|$. Then the expected distance of the answers to* CircuitVal-RPCPP*'s queries from satisfying is at least*

$$\frac{\beta}{64} \cdot \min_{y' \in \mathtt{Sat}(C)} \left\{ \max \left(\delta(y, y'), \delta\left(e, \widehat{y'_C}\right), \delta\left(f, F_{C;\widehat{y'}}\right), \delta(p, P(C; y')), \delta(q, Q(C; y')) \right) \right\} \tag{14}$$

where $\mathtt{Sat}(C)$ *denotes the set of satisfying inputs of C, β is the constant of Proposition 5.7, and $\widehat{y'_C}$, $F_{C;\widehat{y'_C}}$, $P(C; y')$ and $Q(C; y')$ are the canonical proof-pieces of y' as described in Algorithm 5.15.*

Proof. Fix input C and oracles y, e, f, p and q. Let $\widetilde{e}$, $\widetilde{f}$, $\widetilde{p}$, $\widetilde{q}$, $\widetilde{p_i}$ and $\widetilde{q_i}$ as in the proof of Lemma 5.16, and let y' be the minimizer of Expression 14.

Following the analysis of CircuitSat-RPCP, if $\widetilde{f}$, $\widetilde{p}$ and $\widetilde{q}$ are not the canonical proof-pieces of $\widetilde{e}$ proving that it is the low-degree extension of a satisfying assignment to C, then the expected distance of answers to queries issued to e ,f, p and q from satisfying is at least $\beta/30$. Since these answers account for more than $6/8$ of all answers, then the expected distance of answers to queries issued to all oracles is at least $6/8 \cdot \beta/30$.

Otherwise $\widetilde{f}$, $\widetilde{p}$ and $\widetilde{q}$ are the canonical proofs proving that $\widetilde{e}$ encodes a satisfying assignment denoted $\widetilde{y}$. Again using the analysis of CircuitSat-RPCP we have that the expected distance of answers to queries issued to all oracles is at least

$$\frac{6}{8} \cdot \frac{\beta}{30} \cdot \max\left(\delta\left(e, \widehat{y'_C}\right), \delta\left(f, F_{C;\widehat{y'_C}}\right), \delta(p, P(C;y')), \delta(q, Q(C;y'))\right)$$

We now argue that the expected distance is proportional also to $\delta(y, y')$. With probability $\Omega(\delta(y, y'))$, either the query from y must be changed (and it has a constant fraction of the weight of all queries), or e must be changed on a constant fraction of locations on the line $\mathcal{L}$ (due to self-correction). Details follow.

Notice that if e is $1/4$-far from $\widetilde{e}$ then by Proposition 5.7 and a similar weighting argument, the distance of all answers from satisfying is at least $1/4 \cdot \beta \cdot (1/8 - 1/|\mathbb{F}|)$,[24] so what's left is to show that the expected distance is proportional to $\delta(y, \widetilde{y})$. Indeed, if the sampled location x in Step 2 is one on which y and $\widetilde{y}$ differ, i.e. such that $y(x) \neq \widetilde{e}(x)$, then either the value of $y(x)$ must be changed or $e[\mathcal{L} \setminus \{z\}]$ must be changed to agree with a univariate degree d polynomial whose value at x is $y(x)$. Even with x fixed, all points in $\mathcal{L} \setminus \{x\}$ are marginally uniform, so by linearity of expectation,

$$\mathop{\mathbb{E}}_{\text{Random } \mathcal{L} \text{ with intercept } x} [\delta(e[\mathcal{L} \setminus \{x\}], \widetilde{e}[\mathcal{L} \setminus \{x\}])] = \delta(e, \widetilde{e}) \leq 1/4$$

By Markov's inequality, for any $x \in \mathbb{F}^m$ and random line $\mathcal{L}$ with intercept x, with probability at least $1/2$ it holds that $e[\mathcal{L} \setminus \{x\}]$ is $1/2$-close to $\widetilde{e}[\mathcal{L} \setminus \{x\}]$ and therefore $e[\mathcal{L} \setminus \{z\}]$ is $2/3$-close to $\widetilde{e}[\mathcal{L} \setminus \{z\}]$ (this accounts for the case that $z = x + h$ in Step 2. Conditioned on this event, $e[\mathcal{L} \setminus \{z\}]$ must be changed on at least $1 - d/|\mathbb{F}| - 2/3 \geq 1/4$ fraction of locations to make the consistency check of Step 2 pass.

All in all, we have that with probability at least $\delta(y, \widetilde{y})/2$, at least a quarter of the answers received either from y or from e (in Step 2) must be changed so that Step 2 does not reject. As these account for at least a $(1/8 - 1/|\mathbb{F}|)$ of all answers received, we have that the expected distance is at least $\delta(y, \widetilde{y})/64$. ◀

5.4 Alphabet Reduction

So we have an RPCPP for CircuitVal, but we aren't done just yet: CircuitVal-RPCPP is *non-Boolean*, meaning that it's proofs are written using non-binary symbols. It's time to show how to transform it to a Boolean RPCPP. Actually, we will show how to transform *any* non-Boolean RPCPP to a Boolean one of comparable complexities – provided its symbols are not much larger than the number of queries it issues. Most importantly, we prove that this transformation preserves smoothness and strong canonicality.

We show this even for multi-piece RPCPPs that have different-sized alphabets for different pieces. That is, letting the ith proof-piece oracle answers its queries with symbols in the alphabet $\Sigma_i = \{0,1\}^{\sigma_i}$ it could be that not all σ_i's are equal. We call σ_i the *ith proof-piece answer-length complexity* of the RPCPP.

[24] Due to Step 2, but in fact low-degree tests occur in Step 1 as well.

▶ **Lemma 5.19.** *Suppose* CircuitVal *has a smooth and strong canonical multi-piece RPCPP with constant residual circuit distance, ith answer complexity σ_i and query complexity q, and the additional property that the verifier makes the same amount of queries to each of its oracles up to a constant multiplicative factor ρ.*[25] *Then,* CircuitVal *has a Boolean smooth and strong canonical multi-piece RPCPP with query complexity $O(q \cdot \max_{i\in[k]} \sigma_i)$, decision complexity growing by an additive factor of $\widetilde{O}(q \cdot \max_{i\in[k]} \sigma_i)$, and strongness parameter shrinking by a constant factor.*

Following [20, Section 5.4.3], the Boolean PCP is obtained by replacing each letter of the non-Boolean PCP with its encoding in an error correcting code of constant relative distance. Since each proof-piece may have different answer complexity, we choose a different code for each proof-piece. Starting with a good code (i.e., of constant relative distance and rate) for the proof-piece with maximal answer-length complexity, we take the other codes to have equal distance and *block length*, increasing the rate if necessary using code repetition. Equal block length is necessary so that the generalized Hamming distance between two non-Boolean proofs is reflected accurately by their corresponding Boolean proofs (and is not scaled with differences in answer complexities as in Remark 5.2).

The Boolean verifier emulates the non-Boolean one: for each symbol the non-Boolean verifier queries, the Boolean verifier queries all bits in the corresponding block and answers the emulated verifier with their decoding (rejecting if the block was not decodable). We then prove that any Boolean proof is either far from being composed of codewords (therefore far from being accepted by the Boolean verifier in expectation), or its expected distance from satisfying the Boolean verifier is similar to the expected distance of its *decoding* from satisfying the non-Boolean verifier, where its *decoding* refers to the non-Boolean string obtained by decoding each of its blocks. Since the distance of the Boolean proof from the canonical proof (for the Boolean verifier) is proportional to the distance of the decoded proof from the canonical proof (for the non-Boolean verifier), strong canonicality of the non-Boolean verifier implies strong canonicality of the Boolean one. A formal proof follows.

Proof of Lemma 5.19. Let $\widehat{V}$ be the smooth strong canonical RPCPP for CircuitVal with corresponding proof-piece strategies $\Pi_1, \ldots, \Pi_t$, with Π_i answering its queries with symbols in the alphabet $\Sigma_i = \{0,1\}^{\sigma_i}$. For each $i \in [t]$ let ℓ_i denote the length complexity of the ith proof-piece oracle, and let $\mathrm{ECC}_i \colon \Sigma_i \to \{0,1\}^b$ with $b = O(\max_{i\in[t]} \sigma_i)$ be an error correcting code of constant minimal distance $2c > 0$, decodable using circuits of size $\widetilde{O}(b)$. The ith canonical proof-piece of circuit C and input y is $\Gamma_i(C;y)$, where $\Gamma_i(C;y)[j] \coloneqq \mathrm{ECC}_i(\Pi_i(C;y))$ is a binary string of length b.[26]

The Boolean RPCPP verifier, denoted V, expects circuit C, input oracle y and proof-piece oracles $\gamma_1, \ldots, \gamma_t$. It emulates $\widehat{V}$ on circuit C and input oracle y as follows:

- When $\widehat{V}$ queries the jth location of the ith proof-piece oracle, issue b queries to γ_i to receive a binary string $\gamma_i[j]$ of length b. Decode $\gamma_i[j]$ using the decoding algorithm of ECC_i. If decoding failed then reject, and otherwise answer $\widehat{V}$'s query with the decoded string.
- When $\widehat{V}$ queries its input oracle, answer it according to the input oracle y, giving the query weight b.

[25] That is, for any two oracles, the ratio between the number of queries that the verifier makes to each oracle is at least ρ.

[26] To be clear, $\Pi_i(C;y)$ is a string of length ℓ_i over alphabet $\Sigma_i = \{0,1\}^{\sigma_i}$, and $\Gamma_i(C;y)$ is a string of length $\ell_i \cdot b$ over the binary alphabet.

Smoothness of V follows immediately from the smoothness of $\widehat{V}$: for each proof-piece, a bit is queried if and only if the emulated verifier queried the location which it allegedly encodes, and the the probability that the latter is queried is the same for all locations.

We show that V is a strong canonical RPCPP for CircuitVal. Fix circuit C, input oracle y and alleged proof-piece oracles $\gamma_1, \ldots, \gamma_t$. For each $i \in [t]$ and $j \in [\ell_i]$, let $\widehat{\pi}_i[j] \in \Sigma_i$ be the "best decoding" of $\gamma_i[j]$, namely, the string $x \in \Sigma_i$ that minimizes the expression $\delta(\mathrm{ECC}_i(x), \gamma_i[j])$. Let $\widehat{\gamma}_i[j]$ be the encoding of $\widehat{\pi}_i[j]$, that is $\widehat{\gamma}_i[j] \coloneqq \mathrm{ECC}_i(\widehat{\pi}_i[j])$.

▷ Claim 5.19.1. For all $i \in [t]$, the expected distance of answers to V's queries from being accepted is greater than

$$\frac{\rho}{t+1} \cdot \delta(\gamma_i, \widehat{\gamma}_i)$$

Proof of Claim 5.19.1. Fix $i \in [t]$ and denote by J the locations that V queries in γ_i. To satisfy V, the answers of γ_i to V's queries (denoted $\gamma_i[J]$) must be codewords, therefore the expected distance of $\gamma_i[J]$ from satisfying V (over J generated according to V's random coins) is at least the expected distance of $\gamma_i[J]$ from being codewords; that is, at least

$$\mathop{\mathbb{E}}_{J}\left[\delta(\gamma_i[J], \widehat{\gamma}_i[J])\right] = \mathop{\mathbb{E}}_{J,\, j \in J}\left[\delta(\gamma_i[j], \widehat{\gamma}_i[j])\right] \tag{15}$$

Smoothness of V implies that a uniformly random element of $j \in J$ is distributed uniformly at random in $[\ell_i]$ (as detailed in Appendix B.2). Thus,

$$\textit{Equation (15)} = \mathop{\mathbb{E}}_{k \in [\ell_i]}\left[\gamma_i[k], \widehat{\gamma}_i[k]\right] = \delta(\gamma_i, \widehat{\gamma}_i)$$

The distance shrinks by $\rho/(t+1)$ because $\gamma_i[J]$ make up for only (at least) a $\rho/(t+1)$ fraction of all bits read by V. ◁

▷ Claim 5.19.2. Fix a random coin sequence and suppose that the Boolean verifier V received answers $s = s_1 \cdots s_q$ from the oracles $y, \gamma_1, \ldots, \gamma_t$. Let $\widehat{s} = \widehat{s}_1 \cdots \widehat{s}_1$ denote the answers that the non-Boolean verifier $\widehat{V}$ received from oracles $y, \widehat{\pi}_1, \ldots, \widehat{\pi}_t$ when it samples the same coin sequence; in other words, letting $s_k = \gamma_i[j]$ for some i and j, we denote $\widehat{s}_k = \widehat{\pi}_i[j]$.

Let D and $\widehat{D}$ be the residual circuits generated by V and $\widehat{V}$ upon sampling the fixed random coin sequence. It holds that $\delta(s, \mathtt{Sat}(D)) \geq c \cdot \delta(\widehat{s}, \mathtt{Sat}(\widehat{D}))$, where $\mathtt{Sat}(\cdot)$ denotes the set of satisfying inputs of a circuit, and $2c$ is the distance of the error correcting codes ECC_i.

Proof of Claim 5.19.2. The proof follows from the Markov bound and a triangle inequality. Suppose there is $s' = s'_1 \cdots s'_q$ that satisfies D such that $\delta(s, s') = \Delta$. For at least a $1 - \Delta/c$ fraction of $k \in [q]$ it holds that s_k is c-close to s'_k. For these k's, s'_k is the closest codeword to s_k, because distinct codewords are $2c$-far apart, and s'_k is a codeword for it satisfies D. Therefore, for a $1 - \Delta/c$ fraction of k's it holds that s'_k encodes $\widehat{s}_k$, and it follows that $\widehat{s}$ is Δ/c-close to satisfying $\widehat{V}$. ◁

Concluding the proof of Lemma 5.19

Fix $y' \in \mathtt{Sat}(C)$ such that $\max_{i \in [t]}\{\delta(y, y'), \delta(\widehat{\pi}_i, \Pi_i(C; y'))\}$ is minimal, and let $\Pi'_i \coloneqq \Pi_i(C; y')$ and $\Gamma'_i \coloneqq \Gamma_i(C; y')$. Suppose we give $\widehat{V}$ oracle access to $y, \widehat{\pi}_1, \ldots, \widehat{\pi}_t$. Robust strong canonicality means that the expected distance of locations it reads from satisfying its residual circuit is at least

$$\alpha \cdot \max_{i \in [t]}\left(\delta(y, y'), \delta(\widehat{\pi}_i, \Pi'_i)\right)$$

By Claim 5.19.2, this implies that when the Boolean verifier V is given $y, \gamma_1, \ldots, \gamma_t$, the expected distance of locations it reads from satisfying its residual circuit is at least

$$\alpha \cdot c \cdot \max_{i \in [t]} \left(\delta(y, y'), \delta(\widehat{\pi}_i, \Pi'_i)\right)$$

Noticing that $\delta(\widehat{\pi}_i, \Pi'_i) = 2c \cdot \delta(\widehat{\gamma}_i, \Gamma'_i)$, this is at least

$$\alpha c \cdot 2c \cdot \max_{i \in [t]} \left(\delta(y, y'), \delta(\widehat{\gamma}_i, \Gamma'_i)\right) \tag{16}$$

Claim 5.19.1 means that the expected distance of answers to V from satisfying is lower bounded not merely by Expression 16, but by

$$2\alpha c^2 \cdot \max_{i \in [t]} \left(\delta(y, y'), \delta(\widehat{\gamma}_i, \Gamma'_i), \frac{\rho}{t+1} \cdot \delta(\gamma_i, \widehat{\gamma}_i)\right) \tag{17}$$

For each $i \in [t]$, if $\delta(\gamma_i, \widehat{\gamma}_i) \leq \delta(\widehat{\gamma}_i, \Gamma'_i)/2$ then by the triangle inequality $\delta(\widehat{\gamma}_i, \Gamma'_i) \geq \delta(\gamma_i, \Gamma'_i)/2$, and otherwise $\delta(\gamma_i, \widehat{\gamma}_i) > \delta(\widehat{\gamma}_i, \Gamma'_i)/2$, so Expression 17 is at least

$$2\alpha c^2 \cdot \frac{\rho}{2(t+1)} \cdot \max_{i \in [t]} \left(\delta(y, y'), \delta(\gamma_i, \Gamma'_i)\right) \geq \frac{\alpha \rho c^2}{(t+1)} \cdot \min_{y'' \in \mathtt{Sat}(C)} \left\{ \max_{i \in [t]} \left(\delta(y, y''), \delta(\gamma_i, \Gamma_i(x; y''))\right) \right\}$$

where the inequality is because y' is a satisfying input for C, but not necessarily a minimizer of the above quantity. Thus, V is robust strongly canonical.

If $\widehat{V}$ has residual circuit distance c' (Definition 3.4), then V has residual circuit distance $c \cdot c'$, because an answer string that satisfies the decision circuit of V must be formed of encodings of an answer string that satisfies $\widehat{V}$. ◀

5.5 Putting it all together

Because CircuitVal-RPCPP is the Reed–Muller-based RPCPP of [20, Section 5.4], it enjoys the (standard) soundness analysis presented in that work. Paired with the strong canonical soundness of Lemma 5.18, we have a non-Boolean smooth strong canonical RPCPP for CircuitVal of logarithmic randomness complexity, polylogarithmic query complexity and constant residual circuit distance. Decision complexity is also polylogarithmic (though we did not keep track of it explicitly). Using Lemma 5.19 we get a Boolean RPCPP for CircuitVal of similar properties, and a final application of Lemma 2.2 reduces the number of oracles to one – proving Proposition 3.7.

References

1 Josh Alman and Lijie Chen. Efficient Construction of Rigid Matrices Using an NP Oracle. In David Zuckerman, editor, *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, Baltimore, MD, USA, November 9-12, 2019*, pages 1034–1055. IEEE Computer Society, 2019. doi:10.1109/FOCS.2019.00067.

2 Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof Verification and the Hardness of Approximation Problems. *J. ACM*, 45(3):501–555, 1998. doi:10.1145/278298.278306.

3 Sanjeev Arora and Shmuel Safra. Probabilistic Checking of Proofs: A New Characterization of NP. *J. ACM*, 45(1):70–122, 1998. doi:10.1145/273865.273901.

4 Pranjal Awasthi, Avrim Blum, and Or Sheffet. Stability Yields a PTAS for k-Median and k-Means Clustering. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*, pages 309–318. IEEE Computer Society, 2010. doi:10.1109/FOCS.2010.36.

5 Pranjal Awasthi, Moses Charikar, Ravishankar Krishnaswamy, and Ali Kemal Sinop. The Hardness of Approximation of Euclidean k-Means. In Lars Arge and János Pach, editors, *31st International Symposium on Computational Geometry, SoCG 2015, June 22-25, 2015, Eindhoven, The Netherlands*, volume 34 of *LIPIcs*, pages 754–767. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015. doi:10.4230/LIPIcs.SOCG.2015.754.

6 Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil P. Vadhan. Robust PCPs of Proximity, Shorter PCPs, and Applications to Coding. *SIAM J. Comput.*, 36(4):889–974, 2006. doi:10.1137/S0097539705446810.

7 Yonatan Bilu and Nathan Linial. Lifts, Discrepancy and Nearly Optimal Spectral Gap*. *Combinatorica*, 26(5):495–519, 2006. doi:10.1007/s00493-006-0029-7.

8 Yonatan Bilu and Nathan Linial. Are Stable Instances Easy? *Combinatorics, Probability & Computing*, 21(5):643–660, 2012. doi:10.1017/S0963548312000193.

9 Manuel Blum, Michael Luby, and Ronitt Rubinfeld. Self-Testing/Correcting with Applications to Numerical Problems. *J. Comput. Syst. Sci.*, 47(3):549–595, 1993. doi:10.1016/0022-0000(93)90044-W.

10 Irit Dinur, Oded Goldreich, and Tom Gur. Every Set in P Is Strongly Testable Under a Suitable Encoding. In Avrim Blum, editor, *10th Innovations in Theoretical Computer Science Conference, ITCS 2019, January 10-12, 2019, San Diego, California, USA*, volume 124 of *LIPIcs*, pages 30:1–30:17. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2019. doi:10.4230/LIPIcs.ITCS.2019.30.

11 Irit Dinur and Omer Reingold. Assignment Testers: Towards a Combinatorial Proof of the PCP Theorem. *SIAM J. Comput.*, 36(4):975–1024, 2006. doi:10.1137/S0097539705446962.

12 Zachary Friggstad, Kamyar Khodamoradi, and Mohammad R. Salavatipour. Exact Algorithms and Lower Bounds for Stable Instances of Euclidean k-MEANS. In Timothy M. Chan, editor, *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 2958–2972. SIAM, 2019. doi:10.1137/1.9781611975482.183.

13 Anna Gál and Andrew Mills. Three-Query Locally Decodable Codes with Higher Correctness Require Exponential Length. *TOCT*, 3(2):5:1–5:34, 2012. doi:10.1145/2077336.2077338.

14 Oded Goldreich. *Introduction to Property Testing*. Cambridge University Press, 2017. doi:10.1017/9781108135252.

15 Oded Goldreich, Tom Gur, and Ilan Komargodski. Strong Locally Testable Codes with Relaxed Local Decoders. *TOCT*, 11(3):17:1–17:38, 2019. doi:10.1145/3319907.

16 Oded Goldreich, Howard J. Karloff, Leonard J. Schulman, and Luca Trevisan. Lower bounds for linear locally decodable codes and private information retrieval. *Computational Complexity*, 15(3):263–296, 2006. doi:10.1007/s00037-006-0216-3.

17 Oded Goldreich and Madhu Sudan. Locally testable codes and PCPs of almost-linear length. *J. ACM*, 53(4):558–655, 2006. doi:10.1145/1162349.1162351.

18 Tom Gur, Govind Ramnarayan, and Ron D. Rothblum. Relaxed Locally Correctable Codes. In Anna R. Karlin, editor, *9th Innovations in Theoretical Computer Science Conference, ITCS 2018, January 11-14, 2018, Cambridge, MA, USA*, volume 94 of *LIPIcs*, pages 27:1–27:11. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018. doi:10.4230/LIPIcs.ITCS.2018.27.

19 Tom Gur and Ron D. Rothblum. Non-interactive proofs of proximity. *Computational Complexity*, 27(1):99–207, 2018. doi:10.1007/s00037-016-0136-9.

20 Prahladh Harsha. *Robust PCPs of Proximity and Shorter PCPs*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 2004.

21 Jonathan Katz and Luca Trevisan. On the efficiency of local decoding procedures for error-correcting codes. In F. Frances Yao and Eugene M. Luks, editors, *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing, May 21-23, 2000, Portland, OR, USA*, pages 80–86. ACM, 2000. doi:10.1145/335305.335315.

22 Subhash Khot. Hardness Results for Coloring 3 -Colorable 3 -Uniform Hypergraphs. In *43rd Symposium on Foundations of Computer Science (FOCS 2002), 16-19 November 2002, Vancouver, BC, Canada, Proceedings*, pages 23–32. IEEE Computer Society, 2002. doi:10.1109/SFCS.2002.1181879.

23 Christos H. Papadimitriou and Mihalis Yannakakis. Optimization, Approximation, and Complexity Classes. *Journal of Computer and System Sciences*, 43(3):425–440, 1991. doi:10.1016/0022-0000(91)90023-X.

24 Sergey Yekhanin. Locally Decodable Codes. *Foundations and Trends in Theoretical Computer Science*, 6(3):139–255, 2012. doi:10.1561/0400000030.

A Hardness of approximating bounded-occurrence stable3SAT

To see the connection between our PCP result (i.e. Theorem 1.5) and the hardness of approximating bounded-occurrence stable3SAT (i.e. Corollary 1.7), first recall the connection between standard PCPs and constraint satisfaction problems (CSPs).[27] At its heart stands a correspondence between the probability that a nonadaptive PCP verifier V for a set S accepts an input x and proof π, and the fraction of simultaneously satisfiable constraints in a CSP $\Phi_{V,x}$ over $|\pi|$ variables. Consider the CSP $\Phi_{V,x}$ of 2^r constraints that express the verifier's decision after tossing r random coins and querying its proof oracle in q locations. These are constraints over $|\pi|$ variables, with each constraint depending on the q variables corresponding to locations that the verifier queries (i.e., $\Phi_{V,x}$ is a qCSP). A proof π is an assignment to these variables; completeness means that if $x \in S$ then there is an assignment that satisfies all constraints in $\Phi_{V,x}$, and soundness means that if $x \notin S$ then no assignment satisfies more than half of the constraints in $\Phi_{V,x}$ simultaneously.

Now, what can be said about the CSP $\Phi_{V,x}$ if the PCP verifier V was also strong and smooth?

- A strong PCP verifier with strongness parameter α yields a *stable* CSP $\Phi_{V,x}$, in the sense that an assignment that is δ-far from all satisfying assignments violates an $\alpha \cdot \delta$ fraction of constraints. Like strong soundness, this property holds both when $\Phi_{V,x}$ is satisfiable (i.e. when inputs $x \in S$) and when $\Phi_{V,x}$ is unsatisfiable (i.e. when $x \notin S$). We note that in the latter case, $\Phi_{V,x}$ cannot have more than an α fraction of constraints satisfied simultaneously.
- A smooth PCP verifier yields a CSP such that each variable occurs in the same number of constraints.

Thus, Theorem 1.5 implies that for any set $S \in \mathcal{NP}$ there is an efficient parsimonious reduction from S to α-stable bounded-arity qCSPs with equal variable occurrence. To prove Corollary 1.7, we show a polynomial-time computable parsimonious reduction of α-stable qCSPs with *equal variable occurrence* to $\Omega(\alpha)$-stable qCNFs with *bounded variable occurrence* (Proposition A.1), and then reduce the latter to $\Omega(\alpha)$-stable 3CNFs with bounded variable occurrence (Proposition A.2).

Recall that a formula has *b-bounded-occurrence* if any variable appears in at most b clauses, and that the promise problem (α, b)-stable$q$SAT is distinguishing b-bounded-occurrence qCNF formulas that are α-stable and satisfiable from ones in which any assignment violates at least an α fraction of the clauses.

[27] A CSP Φ of size m over ℓ variables is a set of m functions (called *constraints*) $\Phi = \{C_1, \ldots, C_m\}$ with $C_i : \{0,1\}^\ell \to \{0,1\}$. An assignment $A : [\ell] \to \{0,1\}$ *satisfies* constraints C_i if $C_i(A(1), \ldots, A(\ell)) = 1$. We say that Φ has *arity* q (is a qCSP) if each C_i depends on at most q variables. If constraint C_i depends on variable v then we say that v *occurs* in C_i.

▶ **Proposition A.1.** *For any $q \in \mathbb{N}$ there is $b \in \mathbb{N}$ such that there is a polynomial-time computable parsimonious reduction from α-stable qCSPs with equal variable occurrence to $(\alpha/qb, b)$-stable$q$SAT.*

Proof. We will show a reduction to α/qb'-stable qCSPs with b'-bounded variable occurrence for some constant $b' \in \mathbb{N}$ (independent of q, actually). Then, q-arity of the resulting CSP implies that each of its constraints can be expressed as the conjunction of at most 2^q disjunctive clauses, and conjuncting all these clauses gives a CNF formula that is $\alpha/qb'2^q$-stable and $b'2^q$-bounded variable occurrence. Setting $b \coloneqq b'2^q$ gives the required result.

The reduction, taken from [23], replaces the occurrence of each variable in a constraint with a copy of that variable, and adds consistency (i.e. equality) constraints between copies based on an expander graph of constant degree. We stress that the stability of the obtained CNF crucially relies on the equal variable occurrence of the reduced CSP so that each variable is replaced with the same number of consistency checks, and that the new instance is not necessarily stable without this property (see [12, Appendix D]).

Fix an α-stable CSP Φ consisting of m constraints over ℓ variables, such that each constraint depends on exactly q variables and each variable occurs in exactly mq/ℓ constraints.

Fix a d-regular, explicit[28] graph G over mq/ℓ vertices, with an expansion property guaranteeing that for any set of vertices T that consists of at most a half of the vertices, there are $2|T|$ edges crossing from T to its complement (for example, the expanders of [7]). The variables of Φ' are $\{v_C\}$ for each variable v of Φ and each constraint $C \in \Phi$ in which v occurs. There are two types of constraints in Φ':

- For each constraint $C \in \Phi$, a *primal constraint* C' is added, which is simply C but with variables replaced by their corresponding (alleged) copies. For example, if $C = (x \vee \overline{y} \vee z \vee w) \wedge (\overline{w} \vee x \vee u)$, then $C' = (x_C \vee \overline{y_C} \vee z_C \vee w_C) \wedge (\overline{w_C} \vee x_C \vee u_C)$.
- For each variable v of Φ, $\frac{mq}{\ell} \cdot \frac{d}{2}$ *consistency constraints* are added to Φ': letting $\{C_1, \ldots, C_{mq/\ell}\}$ be the set of constraints in which v occurs, for each edge $\{i, j\}$ in G we add a constraint that is satisfied if and only if v_{C_i} equals v_{C_j}.

All in all, Φ' has $\ell \cdot mq/\ell = mq$ variables and $m + \ell \cdot \frac{mq}{\ell} \cdot \frac{d}{2} = (1 + qd/2)m$ constraints.

Each variable of Φ' occurs in $d+1$ constraints, and the reduction is indeed parsimonious since any assignment that satisfies Φ' must be consistent and therefore gives a satisfying assignment to Φ (and vice-versa). We show that Φ' is $\frac{\alpha}{qd}$-stable. The case that Φ is unsatisfiable follows from [23], so we focus on the case that Φ is satisfiable (and then so is Φ'). Let A' be an assignment to Φ' that violates a γ' fraction of its constraints. Let A_{Maj} be an assignment to Φ that gives each variable a value according to the majority value that A' gives to its (alleged) copies; that is, for each variable v, $A_{\mathrm{Maj}}(v) \coloneqq \mathrm{Maj}_C(A'(v_C))$. Let A'_{Maj} be the extension of A_{Maj} back to the variables of Φ'; that is, $A'_{\mathrm{Maj}}(v_C) \coloneqq A_{\mathrm{Maj}}(v)$ for each variable v and constraint C that depends on v. Let γ_{Maj} denote the fraction of constraints in Φ unsatisfied by A_{Maj}.

▷ Claim A.1.1.

$$\gamma' \geq \frac{\gamma_{\mathrm{Maj}} + \delta(A', A'_{\mathrm{Maj}})}{qd}$$

[28] I.e. there is an algorithm that constructs G in polynomial time when given mq/ℓ as input.

Proof of Claim A.1.1. To prove the claim, we show that whatever primal constraints may be satisfied by A' but not by A'_{Maj} (which works against the lower bound we need) are "made up for" in consistency clauses unsatisfied by A'. It will be nicer to deal with whole numbers rather than fractions, so we let a' (resp. a_{Maj}) denote the *number* of constraints of Φ' (resp. Φ) unsatisfied by A' (resp. A_{Maj}), and notice that the claim follows from showing that

$$a' \geq a_{\text{Maj}} + \left|\left\{v_C : A'(v_C) \neq A'_{\text{Maj}}(v_C)\right\}\right|$$

Fix a variable v, and let k be the number of constraints C such that A' disagrees with A'_{Maj} on the value assigned to v_C. Then v occurs in at most k constraints of Φ that are unsatisfied by A_{Maj} but whose corresponding primal constraint is satisfied by A'. But by the expansion property of the graph G, each of the k inconsistencies result in $2k$ consistency constraints unsatisfied by A'. ◁

Now, let B be the closest satisfying assignment to A_{Maj}. Note that the extension of B to the variables of Φ', denoted B', is a satisfying assignment to Φ', and that $\delta(A_{\text{Maj}}, B) = \delta(A'_{\text{Maj}}, B')$. Using Claim A.1.1 and stability of Φ, we have

$$\gamma' \geq \frac{\gamma_{\text{Maj}} + \delta(A', A'_{\text{Maj}})}{qd} \geq \frac{\alpha\delta(A_{\text{Maj}}, B) + \delta(A'_{\text{Maj}}, B')}{qd} \geq \frac{\alpha}{qd} \cdot \delta(A', B') \qquad \blacktriangleleft$$

▶ **Proposition A.2.** *For any $\alpha \in (0, 1]$ and $b, q \in \mathbb{N}$ there is $\alpha' \in (0, 1]$ such that there is a polynomial-time parsimonious reduction from (α, b)-`stable`q`SAT` to (α', b)-`stable3SAT`.*

Friggstad et al. show a similar reduction in [12][Section 4]. Furthermore, their reduction yields CNFs that have *exactly* three literals per clause.

Proof. We use the standard parsimonious reduction of qSAT to 3SAT to transform a formula φ to a formula φ'. The reduction operates clause-by-clause according to the following recurrence relation: for each clause $C \coloneqq (x_1 \vee \cdots \vee x_k)$ in φ (where $k \leq q$),

- If $k < 4$, add the clause C to φ' and halt.
- Otherwise, introduce a "fresh" *auxiliary variable* z, add the clauses $(x_1 \vee x_2 \vee z)$, $(\overline{x_1} \vee \overline{z})$ and $(\overline{x_2} \vee \overline{z})$ to φ', and then recurse on $C' \coloneqq (\overline{z} \vee x_3 \vee \cdots \vee x_k)$.

If φ is not satisfiable then no assignment satisfies more than an $\alpha/3q$ fraction of the clauses of φ' (by the standard soundness of this reduction), so we focus on the case that φ (and therefore φ') is satisfiable. Let A be some assignment for φ', given as an assignment X to the original variables (i.e. to φ) and an assignment Z to the auxiliary variables. Let X^* be a satisfying assignment to φ closest to X. By parsimony of the reduction, there is a unique assignment Z^* to the auxiliary variables such that $A^* \coloneqq (X^*, Z^*)$ is a satisfying assignment.

Again, it would be more convenient for us to deal with absolute quantities rather than relative quantities, so let V_A denote the number of clauses violated by A, and $\overline{\Delta}(A, A^*)$ denote the number of variables on which A and A^* disagree. Similarly define V_X, $\overline{\Delta}(X, X^*)$ and $\overline{\Delta}(Z, Z^*)$. Since φ' has b-bounded variable occurrence then the ratio between its number of variables and number of clauses is at least $1/b$, therefore it suffices to show that $V_A = \Omega(\alpha \cdot \Delta(A, A^*))$.

Now, on the one hand, α-stability of φ means that $V_X \geq \frac{\alpha}{b}\overline{\Delta}(X, X^*)$ (using the assumption that φ has b-bounded variable occurrence to transition from fractional to absolute quantities, just like in the previous paragraph). Additionally, $V_A \geq V_X/3q$ because if X doesn't satisfy a clause in φ then A doesn't satisfy at least one of the corresponding clauses in φ'. Therefore,

$$V_A \geq \frac{\alpha}{3qb} \cdot \overline{\Delta}(X, X^*) \tag{18}$$

On the other hand, the parsimony of the reduction implies that, for any clause C of φ, if X and X^* agree on all q variables occurring in C, and A satisfies all clauses introduced by C (in φ'), then it must be that Z and Z^* agree on all of the auxiliary variables introduced by C. That is, for any clause C, if Z and Z^* disagree on any of the auxiliary variables introduced by C, then either X and X^* disagree on some of the q variables occurring in C, or A violates some of the clauses introduced by C. Therefore,

$$q \cdot \overline{\Delta}(X, X^*) + V_A \geq \overline{\Delta}(Z, Z^*) \tag{19}$$

Combining Equations (18) and (19), we have that $V_A \geq \frac{\alpha}{4qb} \cdot \max\left(\overline{\Delta}(X, X^*), \overline{\Delta}(Z, Z^*)\right)$, and since $\overline{\Delta}(A, A^*) = \overline{\Delta}(X, X^*) + \overline{\Delta}(Z, Z^*)$ we have $V_A \geq \frac{\alpha}{8qb}\overline{\Delta}(A, A^*)$. ◀

B Tedious notes regarding smoothness

B.1 On uniform sampling

Recall that probabilistic Turing machines (and PCP verifiers in particular) obtain their randomness by *tossing random coins*. It is clear how this ability enables uniform sampling from sets of size that is a power of 2, but throughout this work we allow verifiers to sample uniformly random elements from sets of arbitrary (finite) size. Since the technical implementation of such sampling may affect the smoothness of the verifier, a clarification on this matter is due.

To implement a PCP verifier that uses its randomness only to sample a uniformly random element from a set $[N]$, *rejection sampling* may be employed while losing a constant factor in soundness: the verifier samples a uniformly random element $i \in [2^{\lceil \log N \rceil}]$; if $i > N$ the verifier *immediately accepts*, and otherwise it proceeds as intended. Soundness is halved because the sampled i is in $[N]$ with probability $N/2^{\lceil \log N \rceil} \geq 1/2$.

PCP verifiers in this work use their randomness *only* to sample uniformly random elements from a constant number of sets, and are implemented using rejection sampling on the product of sets from which they sample. Namely, a verifier that uniformly samples from sets $\Omega_1, \ldots, \Omega_k$ is implemented using rejection sampling on the set $\Omega_1 \times \cdots \times \Omega_k$. Still, this (only) halves soundness, and adds at most k random coin tosses (as the randomness complexity $r \coloneqq \sum_{i=1}^{k} \log |\Omega_i|$ grows to $\sum_{i=1}^{k} \lceil \log |\Omega_i| \rceil \leq r + k$).

B.2 Smoothness and uniformity

Without loss of generality, we may require that PCP verifiers query each bit in their proof with some positive probability and never query the same location more than once. Any verifier that doesn't satisfy these simplifying assumptions can be transformed into one that does: we add a single uniformly random query, and then modify the verifier so that whenever it attempts to query the same location twice, it uniformly samples from the remaining unqueried locations in the proof instead.

Now here's a thought: Suppose we have a smooth verifier that issues q queries to a proof oracle of length ℓ. We claim that if we look at a set of query locations $I \subseteq [\ell]$ generated by this verifier (based on its random coins), and subsequently choose a uniformly random element $i \in I$ from this set, then i is uniformly distributed in $[\ell]$. Why? Well, smoothness means that the probability that the verifier queries a certain location in the proof oracle in any of its q queries is equal for any certain location (it's equal to q/ℓ, if you must know). By

the previous paragraph, we can assume all queries of the verifier are distinct, so if a certain location is queried by the verifier it's queried exactly once. Therefore, a uniformly random element from the set of query locations is distributed uniformly in the proof.

Smoothness and marginal uniformity

It may seem natural to define smooth PCPs to be those whose queries are marginally uniform.[29] That is, that the first query of the verifier is distributed uniformly in the proof, and so is the second, third, and so on. We claim that there's almost no difference between this notion and ours (Definition 1.3): by the observation made in the previous paragraph, any nonadaptive PCP satisfying Definition 1.3 can be transformed into one that satisfies marginal uniformity by randomly permuting the order of its queries. Actually, even a random cyclic shift would suffice, and only incurs a $\log q$ additive overhead in randomness complexity – exponentially smaller than the original randomness complexity in the constructions used throughout this work.

C The vector-valued low-degree polynomial test

This section was kindly contributed by Oded Goldreich and Madhu Sudan. It sees the generalization of the *Point-line* low-degree polynomial test ([2, Section 7.2]) to the vector-valued setting.

▶ **Definition C.1** (Definition 5.3 restated). *A function $f\colon \mathbb{F}^m \to \mathbb{F}^k$ is a* vector-valued multivariate polynomial of degree at most d *if for all $i \in [k]$ the projection of f to the ith coordinate is a multivariate polynomial of (total) degree at most d, where the projection of f to the ith coordinate, denoted $f_i : \mathbb{F}^m \to \mathbb{F}$, is defined $f_i(x) \coloneqq f(x)_i$ for all $x \in \mathbb{F}^m$.*

▶ **Algorithm C.2** (Algorithm 5.4 restated). The *point-line vector-valued low-degree test (PL-VLDT)* is given access to an oracle $f\colon \mathbb{F}^m \to \mathbb{F}^k$ and a "lines" proof oracle g that maps line $\mathcal{L}$ to vector-valued univariate polynomial $g_{\mathcal{L}} : \mathcal{L} \to \mathbb{F}^k$ of degree at most d. It samples a uniformly random line $\mathcal{L}$ through $\mathbb{F}^m$ and uniformly random point $x \in \mathcal{L}$, and accepts if and only if $f(x) = g_{\mathcal{L}}(x)$.

▶ **Fact C.3.** *For $k = 1$, PL-VLDT is the low degree test of [2, Theorem 65], therefore if input oracle $f\colon \mathbb{F}^m \to \mathbb{F}$ is δ-far from being a polynomial of degree at most d then PL-VLDT rejects with probability at least $\delta/2$ for any lines oracle g.*

▶ **Proposition C.4** (Proposition 5.5 restated). *Assuming $|\mathbb{F}| > 25k$, if input oracle $f\colon \mathbb{F}^m \to \mathbb{F}^k$ is δ-far from being a vector-valued polynomial of degree at most d then for any lines oracle g, PL-VLDT rejects f and g with probability at least $\delta/40$.*

The proof follows three main cases. If a projection of the input is far from being a (scalar-valued) low-degree polynomial then we are done due to Fact C.3. If $\delta = \Omega(1/|\mathbb{F}|)$ then the distance of the restriction $f[\mathcal{L}]$ from being a univariate low-degree univariate polynomial is proportional to the distance of f from being a low-degree polynomial (i.e. δ) with high probability, and assuming $g_{\mathcal{L}}$ is the closest low-degree polynomial to $f[\mathcal{L}]$, rejection occurs with this very probability. The third case is when $\delta = O(1/|\mathbb{F}|)$, and then we capitalize on δ being very small to show that with probability $\Theta(|\mathbb{F}|\delta)$ the restriction $f[\mathcal{L}]$ is low-degree except for exactly one point, which is sampled with probability $1/|\mathbb{F}|$. A formal proof follows.

[29] For example, as used in [1].

Proof. Throughout this proof, *low-degree* means *of degree at most d*. For each $i \in [k]$ let $f_i : \mathbb{F}^m \to \mathbb{F}$ be the projection of f onto its ith coordinate, and let g_i be a mapping of lines through $\mathbb{F}^m$ to low-degree univariate polynomials given by $g_{\mathcal{L},i}(x) \coloneqq g_{\mathcal{L}}(x)_i \in \mathbb{F}$ for each line $\mathcal{L}$ and point $x \in \mathcal{L}$. Let $\widetilde{f}_i \colon \mathbb{F}^m \to \mathbb{F}$ be a low-degree polynomial closest to f_i, and notice that $\widetilde{f} \coloneqq (\widetilde{f}_1, \ldots, \widetilde{f}_k)$ is a vector-valued low-degree polynomial closest to f. Let δ and δ_i be the distances of f from $\widetilde{f}$ and f_i from $\widetilde{f}_i$ respectively.

First, note that if $f_i(x) \neq g_{\mathcal{L},i}(x)$ for the sampled $\mathcal{L}$ and x then the test rejects, therefore the probability that PL-VLDT rejects f and g is greater than the probability that it rejects input oracle f_i and lines oracle $(g_{\mathcal{L},i})_{\mathcal{L}}$. Hence, if there exists i such that $\delta_i \geq 1/5$ then by Fact C.3 rejection occurs with probability at least $1/10$. Therefore, we may assume that $\delta_i < 1/5$ for all $i \in [k]$. We proceed with a partial analysis that assumes that $\delta \leq 2/5$, and later show how the remaining possibilities follow.

Case 1: $\delta > 5/|\mathbb{F}|$. Points on a randomly sampled line $\mathcal{L}$ are pairwise independent and marginally uniform, so the Chebyshev inequality implies that the relative distance between $f[\mathcal{L}]$ and $\widetilde{f}[\mathcal{L}]$ is at least $\delta/2$ and at most $3\delta/2$ with probability greater than $1 - \frac{\delta(1-\delta)}{(\delta/2)^2|\mathbb{F}|} \geq 1/5$. Conditioned on this event, one of two cases must hold:

- **Case 1.1:** There is i such that $g_{\mathcal{L},i} \neq \widetilde{f}_i[\mathcal{L}]$. Note that $g_{\mathcal{L},i}$ and $\widetilde{f}_i[\mathcal{L}]$ are distinct (univariate) polynomials of degree at most d so they agree on at most d points in $\mathcal{L}$. Since $f_i[\mathcal{L}]$ disagrees with $\widetilde{f}_i[\mathcal{L}]$ on at most $3\delta|\mathbb{F}|/2 \leq 3|\mathbb{F}|/5$ points, it holds that $f[\mathcal{L}]$ and $g_{\mathcal{L}}$ agree on at most $d + 3|\mathbb{F}|/5 \leq 4|\mathbb{F}|/5$ points. Therefore rejection occurs with probability at least $4/5 \geq \delta/2$.
- **Case 1.2:** For all i it holds that $g_{\mathcal{L},i} = \widetilde{f}_i[\mathcal{L}]$. Then rejection occurs if and only if $f[\mathcal{L}]$ and $\widetilde{f}[\mathcal{L}]$ disagree on the sampled $x \in \mathcal{L}$, which occurs with probability at least $\delta/2$.

All in all, rejection occurs with probability at least $\frac{1}{5} \cdot \frac{\delta}{2} = \frac{\delta}{10}$.

Case 2: $\delta < 1/2|\mathbb{F}|$. We claim that $f[\mathcal{L}]$ and $\widetilde{f}[\mathcal{L}]$ agree on exactly 1 point with probability at least $|\mathbb{F}|\delta/2$. Again we use pairwise independence and marginal uniformity of points on a random line, this time served with a side of inclusion-exclusion. Let each line $\mathcal{L}$ have a fixed ordering $\mathcal{L} = \{x_1, \ldots, x_{|\mathbb{F}|}\}$. Then,

$$\begin{aligned}
&\mathbb{P}_{\mathcal{L}}\Big[\exists_! \; x \in \mathcal{L} \; f(x) \neq \widetilde{f}(x)\Big] \\
&\quad \geq \sum_{i \in [|\mathbb{F}|]} \mathbb{P}_{x_i}\Big[f(x_i) \neq \widetilde{f}(x_i)\Big] - \sum_{j \in [|\mathbb{F}| \setminus \{i\}]} \mathbb{P}_{x_i, x_j}\Big[f(x_i) \neq \widetilde{f}(x_i), f(x_j) \neq \widetilde{f}(x_j)\Big] \\
&\quad \geq |\mathbb{F}|\big(\delta - |\mathbb{F}|\delta^2\big) = |\mathbb{F}|\delta(1 - |\mathbb{F}|\delta) \geq |\mathbb{F}|\delta/2
\end{aligned}$$

As in Case 1, conditioned on this event rejection occurs with probability at least $4/5 \geq 1/|\mathbb{F}|$ or $1/|\mathbb{F}|$, depending on whether g and $\widetilde{f}$ agree on the random line $\mathcal{L}$. All in all, rejection occurs with probability at least $\frac{|\mathbb{F}|\delta}{2} \cdot \frac{1}{|\mathbb{F}|} = \frac{\delta}{2}$.

Now, if $\delta > 2/5$ we show that there exists $k' < k$ such that the distance of $f_{[k']} \coloneqq (f_1, \ldots, f_{k'})$ from being a vector-valued low-degree polynomial, denoted $\delta_{[k']}$, is greater than $1/5 \geq 5/|\mathbb{F}|$ and less than $2/5$, and since the rejection probability of f and g is greater than that of $f_{[k']}$ and $g_{[k']} \coloneqq (g_1, \ldots, g_{k'})$ we may then apply Case 1 to $f_{[k']}$. Indeed, $\delta_{[k']} - \delta_{[k'-1]} \leq \delta_{k'} \leq 1/5$ and $\delta_{[k]} = \delta \geq 2/5$, so by a greedy argument there must exist $k' \leq k$ such that $\delta_{[k']} \in [1/5, 2/5]$.

Finally, we tend to the case that $\delta \in [1/2|\mathbb{F}|, 5/|\mathbb{F}|]$. If there exists i such that $\delta_i \geq 1/4|\mathbb{F}|$ then by Fact C.3 rejection occurs with probability at least $1/8|\mathbb{F}| \geq \delta/40$. Otherwise, by a greedy argument there exists $k' \leq k$ such that $\delta_{[k']} \in [1/4|\mathbb{F}|, 1/2|\mathbb{F}|]$. Applying Case 2 to $f_{[k']}$, we have that rejection occurs with probability at least $\delta_{[k']}/2 \geq 1/8|\mathbb{F}| \geq \delta/40$. ◀

Approximately Strategyproof Tournament Rules: On Large Manipulating Sets and Cover-Consistence

Ariel Schvartzman
Department of Computer Science, Princeton University, NJ, USA
acohenca@cs.princeton.edu

S. Matthew Weinberg
Department of Computer Science, Princeton University, NJ, USA
smweinberg@princeton.edu

Eitan Zlatin
Department of Computer Science, Princeton University, NJ, USA
ezlatin@princeton.edu

Albert Zuo
Computer Science Department, Stanford University, CA, USA
azuo@stanford.edu

Abstract

We consider the manipulability of tournament rules, in which n teams play a round robin tournament and a winner is (possibly randomly) selected based on the outcome of all $\binom{n}{2}$ matches. Prior work defines a tournament rule to be k-SNM-α if no set of $\leq k$ teams can fix the $\leq \binom{k}{2}$ matches among them to increase their probability of winning by $> \alpha$ and asks: for each k, what is the minimum $\alpha(k)$ such that a Condorcet-consistent (i.e. always selects a Condorcet winner when one exists) k-SNM-$\alpha(k)$ tournament rule exists?

A simple example witnesses that $\alpha(k) \geq \frac{k-1}{2k-1}$ for all k, and [22] conjectures that this is tight (and prove it is tight for $k = 2$). Our first result refutes this conjecture: there exists a sufficiently large k such that no Condorcet-consistent tournament rule is k-SNM-1/2. Our second result leverages similar machinery to design a new tournament rule which is k-SNM-2/3 for all k (and this is the first tournament rule which is k-SNM-(< 1) for all k).

Our final result extends prior work, which proves that single-elimination bracket with random seeding is 2-SNM-1/3 [22], in a different direction by seeking a stronger notion of fairness than Condorcet-consistence. We design a new tournament rule, which we call Randomized-King-of-the-Hill, which is 2-SNM-1/3 and *cover-consistent* (the winner is an uncovered team with probability 1).

2012 ACM Subject Classification Theory of computation → Algorithmic mechanism design

Keywords and phrases Tournament design, Non-manipulability, Cover-consistence, Strategyproofness

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.3

Related Version A full version of the paper is available at `https://arxiv.org/abs/1906.03324`.

Funding *S. Matthew Weinberg*: This work is supported by the National Science Foundation, under grant CNS-0435060, grant CCR-0325197 and grant EN-CS-0329609.

Acknowledgements The authors are extremely grateful to Mikhail Khodak and Jon Schneider, who contributed both with many helpful discussions as well as code to help test the non-manipulability of tournament rules. The authors would also like to thank the anonymous reviewers for their feedback on extensions, clarifications and relevant references unknown to the authors.

1 Introduction

Consider n teams vying for a single championship via pairwise matches. A *Tournament Rule* maps (possibly randomly) the outcome of all $\binom{n}{2}$ matches to a single winner. A successful tournament rule should on one hand be fair, in that it selects a team that could reasonably be considered the best. For example, a tournament rule is Condorcet-consistent if whenever an undefeated team exists, that team is selected as the winner with probability 1. On the other hand, a successful tournament rule should also incentivize teams to win their matches. For example in a monotone tournament rule, no team can unilaterally increase their probability of winning by throwing a match.

While numerous rules satisfy the two specific properties mentioned above, these hardly suffice to call a tournament both fair and incentive compatible. Consider for example a single-elimination tournament, which is Condorcet-consistent and monotone. One might reasonably argue that single-elimination is unfair in the sense that a *covered* team may win. That is, some team x, who is beaten by y, and for which all z who beat y also beat x, could be crowned the champion (Observation 19), even though y is in some sense clearly a superior team. When multiple teams come from the same organization (e.g. in the Olympics where multiple teams from the same country participate, or in eSports where the same organization sponsors multiple teams), one could also argue that single-elimination is not incentive compatible: two teams from the same organization may wish to fix the match between them so that the team with the best chance of winning gold advances.

Prior work establishes, however, that this stronger notion of incentive compatibility (termed 2-Strongly Nonmanipulable by [1], and previously Pairwise Nonmanipulable in [2]) is incompatible even with the basic notion of Condorcet-consistency: no 2-SNM tournament rule is Condorcet-consistent (recapped in Lemma 17). This motivated [1] to seek instead tournaments that were 2-SNM and *approximately* Condorcet-consistent (i.e. guaranteed to pick an undefeated team with probability at least $\alpha > 0$, whenever one exists), and later [22] to seek tournaments which were Condorcet-consistent and *approximately* 2-SNM (i.e. the maximum probability with which two teams can improve their joint probability of winning by fixing a match is $\alpha < 1$, termed 2-SNM-α).

Like [22], we find it more reasonable to seek a tournament which is only approximately strategyproof rather than one which is only approximately Condorcet-consistent: it is hard to imagine a successful sporting event which sends an undefeated team home empty-handed. The main result of [22] proves that a Single-Elimination Bracket with Random seeding (RSEB, formally defined in Section 2) is both Condorcet-consistent and 2-SNM-1/3. This is tight, as no Condorcet-consistent tournament is 2-SNM-α for any $\alpha < 1/3$. They also define a tournament to be k-SNM-α if no set of $\leq k$ teams can fix the $\leq \binom{k}{2}$ matches between them and improve their joint probability of winning by α, establish that no Condorcet-consistent tournament rule is k-SNM-α for $\alpha < \frac{k-1}{2k-1}$ (recapped in Lemma 17), and conjecture that this is tight. The main open problem posed in their work is to prove this conjecture (recapped in Question 18). The main results of this paper extend [22] in three different directions:

- First, we resolve the main open problem posed in [22] by *refuting* their conjecture (including two weaker forms): There exists a sufficiently large k such that no Condorcet-consistent tournament rule is k-SNM-1/2 (and therefore not k-SNM-$\frac{k-1}{2k-1}$ either).
- Second, we develop a new Condorcet-consistent tournament rule which is k-SNM-2/3 for all k. All tournament rules are trivially k-SNM-1 for all k, and this is the first tournament rule known to be k-SNM-α for all k, for any $\alpha < 1$.
- Finally, we develop a new *cover-consistent* tournament rule which is 2-SNM-1/3, which we call Randomized King-of-the-Hill (RKotH).

1.1 Theorem Statements, Roadmap, and Technical Highlights

After overviewing related work in Section 1.3, and establishing preliminaries in Section 2, we develop machinery related to our first two results in Section 3. Specifically, consider the following: there are two kinds of manipulations which may cause a set S of teams to improve their joint probability of winning under a Condorcet-consistent tournament rule $r(\cdot)$. First, perhaps S contains i and every team which beats i (for some i). Then S can make i a Condorcet winner, and improve their joint probability of winning to 1 (the maximum possible). Or, perhaps every team in S loses to at least one team outside S, but S can increase their joint probability of winning anyway.

If, for a particular tournament graph T, we wish to ensure that no set S can improve their joint probability of winning by more than α under $r(\cdot)$ by creating a Condorcet winner in S, this is a fairly simple linear constraint: we must only ensure that for each team i, the joint probability of winning (under $r(T)$) of i and all teams that beat i in T is at least $1-\alpha$. For each tournament graph T, this reasoning gives a feasibility linear program (with n variables corresponding to the probability that each team wins under $r(T)$, and n linear constraints which depend on T) that every winning-probability vector $r(T)$ must satisfy in order for $r(\cdot)$ to possibly be k-SNM-α. Note that these constraints are by no means *sufficient* to guarantee k-SNM-α, however, as we have completely ignored the second type of constraints.

Inspired by this feasibility LP, we study a similar LP in Section 3, which we call the *Special Linear Program* (SLP). In particular, we show that SLP has a unique solution, and therefore well-defines a tournament rule (if the outcome of the matches is T, solve the SLP parameterized by T and select according to these probabilities). Surprisingly, this LP and its unique optimal solution have been studied decades ago [16, 10] (see Section 1.3 for further discussion of this – the LP describes the unique Nash equilibrium of a generalized rock-paper-scissor game). In Section 4, we explain why the SLP Tournament Rule is special: **if *any* tournament rule is k-SNM-1/2 for all k, then the SLP Tournament Rule is k-SNM-1/2 for all k.** The remainder of Section 4 is then just a simple six team example witnessing that the SLP Tournament Rule is *not* 3-SNM-1/2 (and therefore not k-SNM-1/2 for all k), yielding our first main result:

▶ **Theorem 1.** *There exists $k < \infty$ such that no Condorcet-consistent tournament rule is k-SNM-1/2.*

Note that Theorem 1 implies that (a) no Condorcet-consistent tournament rule is simultaneously k-SNM-$\frac{k-1}{2k-1}$ for all k, (b) there exists a k for which no Condorcet-consistent tournament rule is k-SNM-$\frac{k-1}{2k-1}$, and (c) no Condorcet-consistent tournament rule is k-SNM-1/2 for all k, thereby refuting the main conjecture of [22] along with two weaker conjectures.

We also wish to emphasize the following: in principle, if one wishes to determine whether a particular tournament rule is k-SNM-α for tournaments of n teams, one could do an exhaustive search over all $2^{\binom{n}{2}}$ tournament graphs (with some savings due to isomorphism), and all $\binom{n}{k}$ possible manipulating sets. This is a feasible search for small values of n, k. If one wishes to determine whether there *exists* a rule that is k-SNM-α for tournaments of n teams, one could still imagine an exhaustive search. But observe that the space of tournament *rules* for n teams lies in $n2^{\binom{n}{2}}$-dimensional space, and it is hard to imagine a successful exhaustive search beyond $n = 10$ (and even that is likely impossible).

Our proof establishes that no 939-SNM-1/2 tournament rule exists for $n = 1878$ teams, and there is no hope of discovering this via exhaustive search. Indeed, our own exhaustive searches found numerous candidate rules which were k-SNM-1/2 on n teams for $k, n \leq 7$

(indicating that large parameters are provably necessary before the conjecture is false). Yet, the SLP Tournament Rule is already not 3-SNM-1/2 even for 6 teams (which we found via exhaustive search), and the machinery of Sections 3 and 4 allows us to use this tiny example to conclude that no 939-SNM-1/2 tournament rules exist for $n = 1878$ teams.

In Section 5, we show how to use the machinery developed in Section 3 to propose new tournament rules. The main idea is the following: the SLP Tournament Rule, by design, does a great job discouraging manipulations that produce a Condorcet winner. In fact, no such manipulation benefits the manipulators by more than 1/2. Unfortunately, the rule itself may still be only k-SNM-1 (due to manipulations which don't produce a Condorcet winner). However, we show that a convex combination of the SLP Tournament Rule with the simple rule which selects a Condorcet winner when one exists, or a uniformly random winner otherwise can leverage the SLP properties to be less manipulable:

▶ **Theorem 2.** *There exists a Condorcet-consistent tournament rule that is k-SNM-2/3 for all k.*

▶ **Observation 3.** *The rule referred to in Theorem 2 is* not *monotone. We provide an example of its non-monotonicity in Appendix B for completeness.*

Finally, in Section 6, we shift gears and extend the results of [22] in a different direction. Specifically, we design a new tournament rule called Randomized King-of-the-Hill (RKotH) which is cover-consistent and 2-SNM-1/3. Each round, the rule checks if there is a team that is a Condorcet winner. If there is one, it declares that team the winner and terminates. Otherwise, it selects a uniformly random "prince" among the remaining teams, and then removes it and every team which the prince beats. When there is only one team left, that team is crowned champion. It is not hard to see that RKotH is cover-consistent, so the main result of this section is that RKotH is 2-SNM-1/3. The main idea in the proof is that the joint winning probability of $\{u, v\}$ when v beats u is *only* higher than when u beats v if u is selected as a prince while v still remains in contention, and at least one team which beats either u or v remains in contention as well. We are then able to show that the probability of this event is at most 1/3, and therefore the rule is 2-SNM-1/3.

▶ **Theorem 4.** *Randomized King-of-the-Hill is cover-consistent and* 2-SNM-1/3.

In fact, RKotH satisfies a property stronger than cover-consistency. It will provably pick teams that belong to the *Banks set* of the tournament, which may be strictly smaller than the set of uncovered teams. We defer the definition of the Banks and related proofs set to Section 6 but point out that the support of RKotH is *exactly* the Banks set of a tournament.

▶ **Lemma 5.** *A team v is in the Banks set of a tournament T if, and only if,* RKotH *declares v as the winner with non-zero probability.*

1.2 Extensions and Brief Discussion

While the main appeal of our results is clearly theoretical (it is hard to imagine a 939-team coalition manipulating a real tournament), the events motivating a deep study of fair and incentive compatible tournament rules are not purely hypothetical. In the popular "group stage" format, strategic manipulations have occurred on the grandest stage, including Badminton at the 2012 Olympics and the "disgrace of Gijón" in the 1982 World Cup. But narrowing one's focus exclusively to incentives (and, e.g., running a single-elimination bracket) may have negative consequences for the quality of winner selected. For example in

the 2010 World Cup, eventual winners and second seed Spain lost their opening match to Switzerland, who didn't advance out of the group stage (the implication being that Spain could be considered a "high quality winner" who would have been immediately eliminated in a single-elimination bracket). So while our particular results are valuable mostly for their theoretical contributions, the surrounding literature provides valuable insight on the tradeoff between incentive compatibility of the winner selection process and quality of the winner selected.

We also wish to briefly note that while we formally define SNM only for deterministic tournaments (i.e. the teams try to manipulate from a fixed tournament graph T to another fixed tournament graph T'), [22] establishes that all results extend to (arbitrarily correlated) distributions over tournament graphs as well (where the "real" outcomes may be a distribution $\mathcal{T}$ over tournament graphs, and the coalition may try to manipulate to a different distribution $\mathcal{T}'$). We refer the reader to [22] for a formal statement, but the main idea is that any lower bounds immediately carry over, while for every rule the largest possible manipulation occurs on a deterministic instance anyway (so positive results carry over as well).

1.3 Related work

The most related works have already been discussed above: [2] first introduces the terminology used for these problems (and establishes that no deterministic tournament rule is 2-SNM), [1] first considers randomized tournament rules and designs tournament rules which are 2-SNM but only approximately Condorcet-consistent. [22] is the most closely related, which also considers rules which are Condorcet-consistent and approximately incentive compatible. Our work can most appropriately be viewed as extending [22] in multiple directions (including resolving their main open problem) as detailed in Section 1.

Also related are some recent works which rigorously analyze the manipulability of specific tournament formats (most notably, the World Cup and related qualifying procedures) [20, 7].

Incentive compatibility of *voting rules* has an enormous history, dating back at least to seminal works of [3, 11, 21, 12]. While there are obvious conceptual connections between voting rules and tournament rules (e.g. any tournament rule can be used as a voting rule: call the "match" between alternatives x and y won by x if more voters prefer x to y), the notions of manipulability are quite different. For one, voters have preferences over alternatives whereas teams in tournaments only care about their collusion's joint probability of winning. Moreover, in a voting rule, a *voter* has a tiny role to play in every single "match," whereas in a tournament, the *teams* themselves can manipulate only matches that involve them. So there is little technical (and even conceptual) similarity between works which study incentives in voting rules versus tournament rules.

The linear program we introduce in Section 3 has been studied in a related context by two independent works [16, 10]. They consider the following two player zero-sum game. Given a tournament T the players must pick a team to represent them and reveal it simultaneously. If they pick the same team, no one wins. Otherwise, the winner is determined by the edge that they jointly query from T. The two works showed that, for any given tournament T, there is a unique mixed strategy Nash equilibrium which can be computed in polynomial time through linear programming (and this LP is exactly our $\mathsf{SLP}(T)$). Some proof techniques are similar, but our interest in $\mathsf{SLP}(T)$ is a means to drastically different end. For more on the history and properties of the solution to these LPs, known as *maximal lotteries*, see [8, 5].

The notion of uncovered teams is also extremely well-studied in computational social choice theory (see, e.g., [6, 17], the latter attributes the concept's introduction to [9] and [19] independently). Additionally, an uncovered team is equivalent to the notion of a "king" [23]

(a team x such that for all teams y, either x beats y or there exists a z who beats y such that x beats z – not to be confused with the kings of our hill) in works which study how a single-elimination bracket designer can rig the seeding to make a particular team win [24, 15, 25] or sufficient conditions under which a covered team can be crowned winner of a single-elimination bracket [14]. The volume of these works certainly helps establish that cover-consistence is a valuable endeavor beyond Condorcet-consistence, but otherwise bear no technical similarity to our work. To the best of our knowledge, this is the first paper to consider cover-consistence jointly with a notion of incentive compatibility.

Another related notion is that of the Banks set of a tournament [4], which is stricter than the set of uncovered teams. While it is NP-Complete to decide if a given team is in the Banks set of a tournament [26], there exist algorithms that can efficiently output an element from the Banks set [13]. It is worth pointing out that the algorithm we propose to sample teams from the Banks set, Algorithm 1, has a different implementation from that of [13] even if they will output the same set of teams.

2 Preliminaries

In this section we introduce notation, and develop some concepts that will be relevant throughout the paper, consistent with prior work [22, 1].

▶ **Definition 6** (Tournament). *A (round robin) tournament T on n teams is a complete, directed graph on n vertices whose edges denote the outcome of a match between two teams. Team i beats team j if the edge between them points from i to j. T_n denotes the set of all n-team tournaments.*

▶ **Definition 7** (Tournament Rule). *A tournament rule r is a function $r : T_n \to \Delta([n])$ that maps n-team tournaments $T \in T_n$ to a distribution over teams, where $r_i(T) = \Pr(r(T) = i)$ denotes the probability with which team i is declared the winner of tournament T under rule r. We will often abuse notation and refer to r as a collection of tournament rules $\{r^1(\cdot), \ldots, r^n(\cdot), \ldots\}$, of which exactly one operates on T_n (for all n).*

Like prior work, we will be interested in tournaments which satisfy natural properties. For instance, [22] concerned tournaments which always select a Condorcet winner, when one exists. Below are the main properties we consider in this paper.

▶ **Definition 8** (Condorcet-consistency). *Team i is a Condorcet winner of a tournament T if i beats every other team according to T. A tournament rule r is Condorcet-consistent if for every tournament T with a Condorcet winner i, $r_i(T) = 1$ (i.e. the tournament rule always declares the Condorcet winner as the winner of T).*

▶ **Definition 9** (cover-consistency). *Team i covers team j under T if (a) i beats j and (b) every $k \notin \{i, j\}$ which beats i also beats j. A team is covered if it is covered by at least one team. A tournament rule r is cover-consistent if for all T, $r_j(T) = 0$ when j is covered.*

▶ **Observation 10.** *Every cover-consistent rule is Condorcet-consistent.*

Proof. If T has no Condorcet winner, then a Condorcet-consistent rule can be aribtrary on T. If T has a Condorcet winner i, then i is the only uncovered team. Therefore, any cover-consistent rule will have $r_i(T) = 1$, and is Condorcet-consistent as well. ◀

Intuitively, one should think of i covering j to mean that any reasonable evaluation should declare team i better than team j. Cover-consistence proposes that no team should win if they are inferior to another by any reasonable evaluation, but does not always propose who

the winner should be. Condorcet-consistence can therefore be interpreted as a relaxation of cover-consistence, which only binds when cover-consistence would propose a unique winner. The following lemma establishes this formally.

▶ **Lemma 11** (Restated Theorem 4 from [18]). *Whenever tournament T has a unique uncovered team i, i is a Condorcet winner in T.*

Proof. We first claim that the covering relation is transitive: if i covers j and j covers k, then i covers k. Indeed, let $S(k)$ denote the teams that k defeats, $S(j)$ for j, and $S(i)$ for i. Then as j covers k and i covers j, we have $S(k) \cup \{k\} \subseteq S(j) \subseteq S(i)$, meaning that i covers k.

Therefore, if we draw the directed graph $G(T)$ with an edge from j to k iff j covers k, the graph must be acyclic. If not, then the work above establishes that a path from j to k implies that j covers k, while a path from k to j implies that k covers j, a contradiction (as we cannot have both that j beats k and k beats j). Uncovered teams are exactly those with indegree 0 in $G(T)$. If there is a unique such team i, then there must be a path from i to every other team j (follow edges backwards starting from j. This process must terminate, and can only terminate at a team with indegree 0, which must be i). Therefore, i covers every other team j, which in particular implies that i beats every other team j and is therefore a Condorcet winner. ◀

The above conditions concern natural properties of the winner selected, and essentially say that good tournament rules should never select obviously inferior teams as their winner. We are also concerned with properties regarding the procedure by which the winner is selected, and in particular how manipulable this procedure is. We formalize these properties below (which are proposed in [1, 22]).

▶ **Definition 12** (S-Adjacent Tournaments). *Two tournaments $T, T' \in T_n$ are S-adjacent when the outcomes of all matches in T, T' are identical, except for the matches between two teams in $S \subseteq [n]$. Formally, for all $i, j \in [n]$, if $|\{i, j\} \cap S| < 2$, then the edge between i and j in T is identical to the one in T'. Less formally, T and T' are S-adjacent if teams in S can manipulate the outcomes of matches* only *between pairs of teams in S and cause the tournament results to change from T to T'.*

▶ **Definition 13** (k-SNM-α). *A tournament rule r is k-strongly non-manipulable at probability α (henceforth k-SNM-α) if for all subsets $S \subseteq [n]$ with $|S| \leq k$, and all pairs T, T' of S-adjacent tournaments, $\sum_{i \in S}(r_i(T) - r_i(T')) \leq \alpha$. Informally, if a set S of $\leq k$ teams decide to manipulate their pairwise matches, they cannot improve the probability that the winner is in S by more than α. We abuse notation and use ∞-SNM-α to refer to a tournament rule which is k-SNM-α for all k.*

2.1 Technical Recap of Prior Work

Finally, let's recap [22], which serves as the starting point for our work. Their main result establishes that the Random Single-Elimination Bracket (formally defined below) is 2-SNM-1/3.

▶ **Definition 14** (Random Single-Elimination Bracket). *The Random Single-Elimination Bracket (*RSEB*) rule places n teams uniformly at random among $2^{\lceil \log_2 n \rceil}$ seeds [1] (and fills the remaining seeds with byes)[2]. Then, a single-elimination tournament is played with these seeds to determine the winner. That is, whenever i meets j in the bracket, T determines which team advances to the next round, and the other team is eliminated. Note that the only randomness in the rule is in the seeding.*

▶ **Theorem 15** ([22]). RSEB *is* 2-SNM-1/3 *and Condorcet-consistent.*

The following explicit tournament was also used in [22] for lower bounds:

▶ **Definition 16** (Balanced Tournament). *The k-balanced tournament is the tournament $T^{Bal} \in T_{2k-1}$ where team i beats exactly the $k-1$ teams in $\{i+1, i+2, ..., i+k-1 \mod (2k-1)\}$.*

▶ **Lemma 17** ([22]). *No Condorcet-consistent Tournament rule is k-SNM-α for any $\alpha < \frac{k-1}{2k-1}$.*

Proof Sketch. Consider $r(T^{\text{Bal}})$. There exists some adjacent set of teams $S = \{i-k+1 \pmod{2k-1}, \ldots, i\}$ of size k which together win with probability at most $\frac{k}{2k-1}$ in $r(T^{\text{Bal}})$. These teams can make i into a Condorcet winner, which necessarily wins with probability 1. Therefore, for any $r(\cdot)$, some set of size k can gain at least $\frac{k-1}{2k-1}$ by manipulating when the original tournament is T^{Bal}. ◀

Inspired by the tightness of Theorem 15 with the simple balanced tournament T^{Bal}, [22] conjectured that same simple tournament would be the worst-case tournament for larger k:

▶ **Open Question 18** ([22]). *Does there exist a tournament rule that is Condorcet-consistent and k-SNM-$\frac{k}{2k-1}$ for all k? What about a family of rules $\mathcal{F}$ such that for all k, F_k is k-SNM-$\frac{k-1}{2k-1}$? What about a rule that is k-SNM-1/2 for all k?*

The first results of this paper refute all three conjectures from [22] and resolve Question 18. The following results concern the difference between Condorcet-consistence and cover-consistence, as the following observation shows that RSEB is not cover-consistent.

▶ **Observation 19.** RSEB *is not cover-consistent.*

Proof. Consider a tournament with eight teams A, B, C, D, E, F, G, H, where A beats exactly $\{B, C, E\}$, and H beats exactly $\{A, B, C, E\}$. C beats D, E beats F, E beats G. Any matches not explicitly stated can be arbitrarily decided. Consider the seeded bracket shown in Figure 1. This bracket shows A can win with non-zero probability. But H covers A. Therefore, RSEB is not cover-consistent. ◀

2.2 Linear Algebra Preliminaries

Some of our proofs require linear algebra. Below are facts that we use, both proofs are in Appendix A.

▶ **Definition 20** (Unit Skew Symmetric Matrix). *An $n \times n$ matrix $A \in \mathbb{R}^{n \times n}$ is* unit skew symmetric *if $|A_{ij}| = 1 \ \forall i \neq j$, and $A_{ij} = -A_{ji} \ \forall i, j$.*

---

[1] A *seed* is a position in the tournament bracket associated to a specific number.
[2] A *bye* is a dummy team that loses to all non-bye teams.

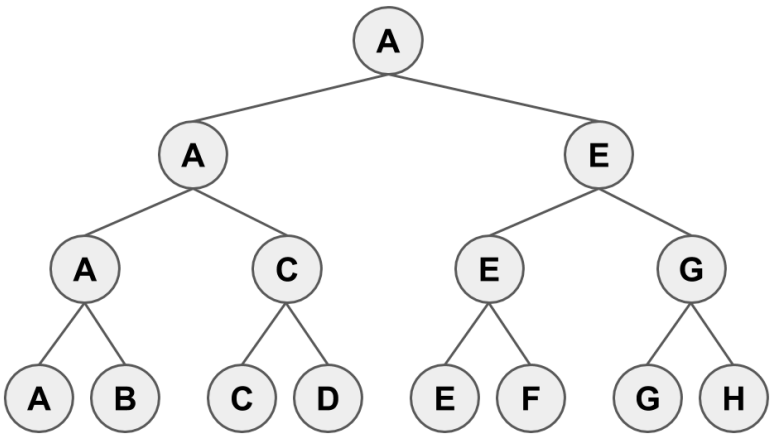


Figure 1 In this example there exists a bracket where a covered team A (covered by H) may still be declared the winner.

▶ **Proposition 21.** *If A is a unit skew symmetric matrix and n is even, rank$(A) = n$. If n is odd rank$(A) = n - 1$.*

▶ **Proposition 22.** *Let $\varepsilon \in \mathbb{R}_{\geq 0}$, $A \in \mathbb{R}^{m \times n}$, and $\vec{b} \in \mathbb{R}^m$ and denote by $P_\varepsilon := \{\vec{x} \in \mathbb{R}^n, A \cdot \vec{x} \geq \vec{b} - \varepsilon\vec{1}\} \cap [0,1]^n$. Then for all $\delta > 0$, there exists a sufficiently small $\varepsilon > 0$ such that:*

$$\max_{\vec{y} \in P_\varepsilon} \{d_{\ell_1}(P_0, \vec{y})\} \leq \delta,$$

where $d_{\ell_1}(S, \vec{x}) = \min_{\vec{y} \in S}\{|\vec{x} - \vec{y}|_1\}$.

3 A Special Linear Program $SLP(T)$

In this section we present a linear program $SLP(T)$ and characterize its optimal solutions. The analysis of $SLP(T)$ is the main tool which allows us to conclude both the non-existence of rules which are ∞-SNM-1/2 (Section 4) and the existence of a rule which is ∞-SNM-2/3 (Section 5). The main result of this section is Proposition 26, which states that $SLP(T)$ has a unique solution, and therefore yields a well-defined tournament rule. In Section 4 we show that a tournament rule that is ∞-SNM-1/2 exists *if and only if* the $SLP(T)$ rule is k-SNM-1/2 (and subsequently show that this rule is not k-SNM-1/2 via Proposition 26). Let T be a tournament graph and let $\delta_T^-(v)$ denote the set of teams that beat v in T (and $\delta_T^+(v)$ the set that v beats). Then $SLP(T)$ is the following:

***SLP(T)*:**

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{n} p_i & & \\ \text{subject to} \quad & \sum_{j \in \delta_T^-(i)} p_j + \frac{1}{2} p_i \geq \frac{1}{2} & & \forall i \in [n] \\ & p_i \geq 0 & & \forall i \in [n] \end{aligned}$$

Before further proceeding, let's get some (informal) intuition for why $SLP(T)$ is possibly related to Question 18. Starting from a tournament rule r, if we define $p_i := r_i(T)$, then $\sum_i p_i = 1$, $p_i \geq 0$ for all i. Moreover, if r is ∞-SNM-1/2, it must be that for all i, $\sum_{j \in \delta_T^-(i)} p_j + p_i \geq 1/2$. If not, then i together with $\delta_T^-(i)$ could collude to make i a Condorcet winner, and i would win with probability 1. So the initial probability of winning for i together with $\delta_T^-(i)$ must have been at least 1/2.

Of course, the afore-described constraints seem very weak in comparison to all of the constraints imposed by k-SNM-1/2. In particular, they only guarantee that no coalition can gain by making one of their members into a Condorcet winner (but do not guarantee that no coalition can otherwise gain by manipulating their matches). Notice now that the constraints in $SLP(T)$ are slightly stronger than this (because they have a multiplier of 1/2 instead of 1 in front of p_i in the constraint for i). In particular, the constraints in $SLP(T)$ imply Condorcet-consistence (while the afore-mentioned do not): if i is a Condorcet winner, then $\delta_T^-(i) = \emptyset$ and the constraint reads $p_i/2 \geq 1/2$ as desired. Of course, we've yet to establish a formal relationship, but at this point the reader may have some intuition for a connection between a profile of solutions to $SLP(T)$ with $\sum_i p_i \leq 1$ (for all T) and Condorcet-consistent tournament rules which are ∞-SNM-1/2.

We postpone a formal discussion of this connection (as this connection is the entire focus of Section 4), but note here that it is not particularly direct. For example, a profile of solutions to $SLP(T)$ for all $T \in T_n$ does not imply a tournament rule for n teams which is ∞-SNM-1/2. Similarly, an ∞-SNM-1/2 tournament rule for n teams does not imply a profile of solutions to $SLP(T)$ for all $T \in T_n$. However, we show that ∞-SNM-1/2 rules exist for all n *if and only if* for all n, the rule defined via profiles of solutions to $SLP(T)$ is ∞-SNM-1/2 (i.e. we will relate this LP on n teams to tournament rules for $\gg n$ teams).

We now begin our analysis of $SLP(T)$ by taking the dual, and refer to it as $D_{SLP}(T)$. Below, we use r_i as the dual variable for the constraint corresponding to team i. On the left-hand side, we've taken the dual directly. On the right hand side, we did a change of variables and redefined $q_i := r_i/2$ (so the two programs below are identical).

$D_{SLP}(T)$:

$$\begin{aligned} \text{maximize} \quad & \sum_{i=1}^{n} r_i/2 \\ \text{subject to} \quad & \sum_{j \in \delta_T^+(i)} r_j + \frac{1}{2} r_i \leq 1 \quad \forall i \in [n] \\ & r_i \geq 0 \quad \forall i \in [n] \end{aligned} \qquad \begin{aligned} \text{maximize} \quad & \sum_{i=1}^{n} q_i \\ \text{subject to} \quad & \sum_{j \in \delta_T^+(i)} q_j + \frac{1}{2} q_i \leq \frac{1}{2} \quad \forall i \in [n] \\ & q_i \geq 0 \quad \forall i \in [n] \end{aligned}$$

We now prove that the optimal value of $SLP(T)$ is always 1. This is stated in Corollary 24, which uses Lemma 23 as a building block.

▶ **Lemma 23.** *Suppose there exists a feasible solution $\vec{p}$ to $SLP(T)$ with $\sum_{i \in [n]} p_i = c$. Then $\vec{q}$ with $q_i := p_i \cdot \frac{1}{2c-1}$ is a feasible solution to $D_{SLP}(T)$ with value $\frac{c}{2c-1}$. Likewise, if there exists a feasible solution $\vec{q}$ to $D_{SLP}(T)$ with $\sum_{i \in [n]} q_i = c$, then $\vec{p}$ with $p_i := q_i \cdot \frac{1}{2c-1}$ is a feasible solution to $SLP(T)$ with value $\frac{c}{2c-1}$.*

Proof. Consider any solution $\vec{p}$ with $\sum_i p_i = c$. First, we observe that we must have $c > 1/2$. If not, there certainly exists some i with $\sum_{j \in \delta_T^-(i)} p_j + p_i/2 < 1/2$, and a constraint is violated (to see this, observe that maybe $c = 0$, in which case all the constraints are violated. Or $0 < c \leq 1/2$, in which case we can take i to be any i with $p_i > 0$). Then because

$\sum_{j\in\delta_T^-(i)} p_j + p_i/2 \geq 1/2$, we must have $\sum_{j\in\delta_T^+(i)} p_j + p_i/2 \leq c - 1/2$. As $q_i := p_i/(2c-1)$, we immediately conclude that $\sum_{j\in\delta_T^+(i)} q_j + q_i/2 \leq \frac{c-1/2}{2c-1} = 1/2$. Also, as $c > 1/2$, each $q_i \geq 0$ (and is well-defined). Therefore, $\vec{q}$ is feasible for $D_{SLP}(T)$, and it's clear that $\sum_i q_i = \frac{c}{2c-1}$. The other direction follows from identical calculations. ◀

▶ **Corollary 24.** *$SLP(T)$ always has an optimal solution with value 1.*

Proof. It is clear that $SLP(T)$ is feasible for all T, since setting $p_i = 1$ for all i is a feasible solution. Suppose we had a primal solution $\vec{p}$ with value $c < 1$. Applying Lemma 23, we can conclude $\vec{q}$ would be a dual solution with value $\frac{c}{2c-1} > c$. By weak LP duality, the existence of such a dual would verify that there are no primal solutions with value c, a contradiction.

Similarly, suppose we had an optimal primal solution $\vec{p}$ with value $c > 1$. This implies there is an optimal dual solution $\vec{q}$ with value $c > 1$. Applying the opposite direction of Lemma 23 we can conclude there is a primal solution with value $\frac{c}{2c-1} < c$, a contradiction. ◀

Now that we know the optimal value of $SLP(T)$, we wish to understand its optimal solution. We now begin taking steps towards characterizing the solution (and in particular, that it is unique).

▶ **Corollary 25.** *Let $\vec{p}$ be an optimal solution to $SLP(T)$. Then for all i s.t $p_i > 0$ and for all optimal solutions $\vec{w}$ to $SLP(T)$, $\sum_{j\in\delta_T^-(i)} w_j + \frac{1}{2}w_i = \frac{1}{2}$.*

Proof. If we apply Lemma 23 to $\vec{p}$ (which has $|\vec{p}|_1 = 1$ by Corollary 24), we conclude $\vec{p}$ is also a feasible dual solution. Hence, $\vec{p}$ is an optimal dual solution (as it has equal value in both the primal and the dual). Now consider applying the complementary slackness conditions for the alternative optimal primal $\vec{w}$, and optimal dual solution $\vec{p}$. If $p_i > 0$, we know that the corresponding primal constraint in $\vec{w}$ must be tight. This exactly states that $\sum_{j\in\delta_T^-(i)} w_j + \frac{1}{2}w_i = \frac{1}{2}$ whenever $p_i > 0$. ◀

▶ **Proposition 26.** *The optimal solution $\vec{p}$ to $SLP(T)$ is unique.*

Proof. By Corollary 24, we know that all solutions to $SLP(T)$ have value 1. Assume toward contradiction there exist two distinct solutions, $\vec{p}$ and $\vec{q}$ to $SLP(T)$ such that $|\vec{p}|_1 = 1$ and $|\vec{q}|_1 = 1$. Let $P = \{i : p_i > 0\}$ and let $Q = \{i : q_i > 0\}$. Let A be the unit skew symmetric matrix where $A_{ij} = 1$ if i beats j in T and -1 otherwise, with $A_{ii} = 0$. By Corollary 25, we know for all i in $P \cup Q$ that

$$\sum_{j\in\delta_T^-(i)} q_j + \frac{1}{2}q_i = 1/2$$

which implies for those same $i \in Q \cup P$ that (because $\sum_j q_j = 1$)

$$(A \cdot q)_i = \sum_{j\in\delta_T^+(i)} q_j - \sum_{j\in\delta_T^-(i)} q_j = 0.$$

Now, let A' denote the submatrix A restricted only to rows and columns in $P \cup Q$. Let $\vec{p}'$ and $\vec{q}'$ be the vectors $\vec{p}$ and $\vec{q}$ (respectively) restricted also to the entries in $P \cup Q$. Observe now that $(A' \cdot \vec{p}')_i = (A \cdot \vec{p})_i = 0$ for all i, and also that $(A' \cdot \vec{q}')_i = (A \cdot \vec{q})_i = 0$ (both of these follow because we have simply deleted all non-zero entries of $A \cdot \vec{p}$ and $A \cdot \vec{q}$ by restricting to $P \cup Q$).

Now we are ready to derive our contradiction. The above paragraph concludes that both $\vec{p}'$ and $\vec{q}'$ are in the null space of A', which is a unit skew symmetric matrix. But also $|\vec{p}|_1 = |\vec{q}|_1$, meaning that the null space of A' must have dimension at least 2. But this contradicts Proposition 21, which claims that the dimension can be at most 1. We therefore conclude that no such distinct $\vec{p}, \vec{q}$ can exist. ◀

Now that we know the solution to $SLP(T)$ is unique, and has $\sum_i p_i = 1$, it yields a well-defined tournament rule, which is the main takeaway from this section:

▶ **Definition 27** (SLP Tournament Rule)**.** *Let $\vec{p}(T)$ denote the (unique, by Proposition 26) solution to $SLP(T)$. Define the* SLP Tournament Rule *to select i as the winner with probability $p_i(T)$ on input T.*

4 No Condorcet-consistent ∞-SNM-$1/2$ Rule Exists

In this section we leverage our analysis of SLP to prove Theorem 1. First, we make the connection between rules that are k-SNM-α and SLP, by introducing a series of linear programs and relaxations.

Recall that for a Condorcet-consistent tournament rule r to be k-SNM-α, it must be that no coalition of size k can gain more than α probability of winning by manipulating the pairwise matches between them. In particular, if $|\delta_T^-(v)| < k$, it must be that $r_v(T) + \sum_{j \in \delta_T^-(v)} r_j(T) \geq 1 - \alpha$. Otherwise the set $\delta_T^-(v) \cup \{v\}$ can collude to make v a Condorcet winner. Formally, any k-SNM-α rule must satisfy the following feasibility $LP_0(T, \alpha, k)$ for all tournaments T.

$LP_1(T, \alpha, z)$:

$$
\begin{aligned}
p_i + \sum_{j \in \delta_T^-(i)} p_j &\geq 1 - \alpha && \forall i \in [n] \text{ such that } |\delta_T^-(i)| \leq k-1 \\
\sum_{\forall v} p_i &= 1 \\
p_i &\geq 0 && \forall i \in [n]
\end{aligned}
$$

Note that any k-SNM-α rule certainly satisfies $LP_0(T, \alpha, k)$ for all T, but that a profile of solutions to LP_0 for all $T \in T_n$ does not necessarily imply a rule which is k-SNM-α (as the LP only considers deviations which produce a Condorcet winner). Note also that the k-balanced tournament witnesses that no rule satisfies $LP_0(T, \alpha, k)$ for any $\alpha < \frac{k-1}{2k-1}$. We will also consider the case where $k \to \infty$ (and therefore, all $i \in [n]$ have $|\delta_T^-(i)| \leq k-1$, and refer to this LP simply as $LP_0(T, \alpha) := LP_0(T, \alpha, \infty)$).

Our first step will be switching from $LP_0(T, \alpha)$ to $LP_1(T, \alpha, z)$ for $z \geq 1$. Below, observe that we have made two changes. The first is insignificant: we've phrased $LP_1(T, \alpha, z)$ as a minimization LP instead of a feasibility LP. The second is a strengthening: we've changed the multiplier of p_i in the constraint corresponding to i from 1 to $\frac{z}{2z-1} \leq 1$ (so the space of feasible solutions is smaller).

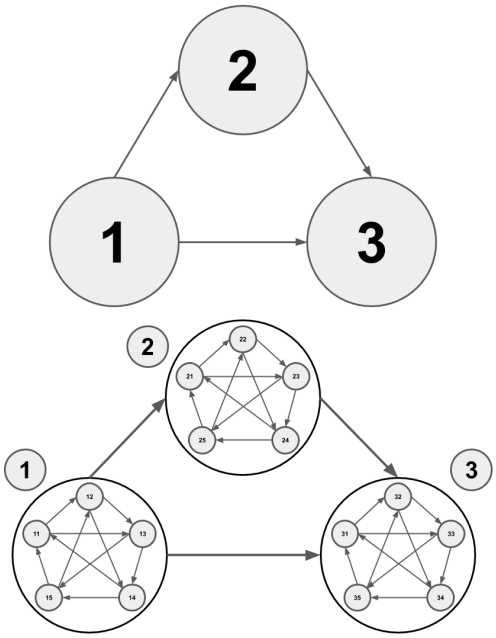


Figure 2 A tournament T (top) and its associated construction T' (bottom), as described in the proof of Lemma 28 for $z = 3$. The outcomes of the games between teams in different components of T' mimic the outcomes of the games between the representative nodes in T.

$LP_1(T, \alpha, z)$:

$$\text{minimize} \sum_{j=1}^{n} p_j$$

$$\text{subject to} \sum_{j \in \delta_T^-(i)} p_j + \frac{z}{2z-1} p_i \geq 1 - \alpha \qquad \forall i \in [n]$$

$$p_i \geq 0 \qquad \forall i \in [n]$$

Observe that $LP_1(T, 1/2, z)$ is a relaxation of $SLP(T)$ (the only difference is a multiplier of $\frac{z}{2z-1} > 1/2$ in front of p_i in the constraint corresponding to i). The main step in this section is Lemma 28 below, which formally connects $LP_1(T, 1/2, z)$ to k-SNM-α rules.

▶ **Lemma 28.** *If for all n there exists a tournament rule $r(\cdot)$ which is ∞-SNM-α, then for all $z \in \mathbb{N}_+$ and all n, there exists a tournament rule $w(\cdot)$ which is ∞-SNM-α and for which $w(T)$ is a feasible solution to $LP_1(T, \alpha, z)$ with $\sum_i p_i = 1$ for all T.*

Similarly, if for all n there exists a tournament rule $r(\cdot)$ which is k-SNM-α, then for all $z \in \mathbb{N}_+$ and all n, there exists a tournament rule $w(\cdot)$ which is $\frac{k}{2z-1}$-SNM-α [3] and for which $w(T)$ is a feasible solution to $LP_1(T, \alpha, z)$ with $\sum_i p_i = 1$ for all T.

Proof. Consider an arbitrary tournament T with n teams, and consider a related tournament T' with $n(2z-1)$ teams, labeled v_{ij}, for $i \in [n]$ and $j \in [2z-1]$. Conceptually, think that we have split each original team into a group of $2z-1$ copies. For $i \neq j$, and $x, y \in [2z-1]$, have v_{ix} beat v_{jy} in T' if and only if v_i beat v_j in T (that is, match results in T are preserved

[3] We abuse notation throughout this section to define $\frac{k}{2z-1}$-SNM-α as $\lfloor \frac{k}{2z-1} \rfloor$-SNM-$\alpha$ when the first term may not be an integer.

between different groups in T'). Within each group, have v_{ix} beat v_{iy} in T' iff $x < y < x + z \pmod{2z-1}$ (so each group is isomorphic to the z-balanced tournament, recall Definition 16 and see Figure 2 for a small example). Let $G(i)$ denote i's group, $G(i) := \{v_{ij}, j \in [2z-1]\}$.

Now, we wish to claim that if $r(\cdot)$ is a rule that is ∞-SNM-α (respectively, k-SNM-α) for $n(2z-1)$ teams, and we define $w(T)$ so that $w_i(T) := \sum_{j=1}^{2z-1} r_{ij}(T')$, then $w(\cdot)$ is a rule that is ∞-SNM-α (respectively, $\frac{k}{2z-1}$-SNM-α) for n teams, *and $w(T)$ is a feasible solution to $LP_1(T, \alpha, z)$ for all T.*

Let's first confirm that $w(T)$ is a feasible solution to $LP_1(T, \alpha, z)$ with $\sum_i p_i = 1$. The latter statement is clear: as $r(\cdot)$ is a tournament rule, we have $\sum_i p_i = \sum_i w_i(T) = \sum_{ij} r_{i,j}(T) = 1$. Next, it is also clear that $p_i \geq 0$ for all i, so we just need to check that $\sum_{j \in \delta_T^-(i)} p_j + \frac{z}{2z-1} p_i \geq 1 - \alpha$.

To this end, we know that there exists *some* adjacent set of z teams in $G(i)$ such that the total probability that these teams win is at most $\frac{z}{2z-1} \cdot p_i$. Call this set S_x and let v_{ix} denote the team in this set which loses to the others. Then the set of teams $\cup_{j \in \delta_T^-(i)} G(j) \cup S_x$ together can create a Condorcet winner (v_{ix}) in T'. Therefore, we get that this set of teams must have won with probability at least $1 - \alpha$ under $r(\cdot)$, and by definition of $w(\cdot)$ (and the choice of S_x above), we immediately get that $\sum_{j \in \delta_T^-(i)} w_j(T) + \frac{z}{2z-1} w_i(T) \geq 1 - \alpha$, as desired.

So now we know that $w(\cdot)$ satisfies $LP_1(T, \alpha, z)$ with $\sum_i p_i = 1$ for all T. We now need to confirm that it is also ∞-SNM-α (respectively, $\frac{k}{2z-1}$-SNM-α). But suppose for contradiction that $w(\cdot)$ was not k-SNM-α for some k (respectively, $\frac{k}{2z-1}$-SNM-α). This would imply the existence of tournaments T_1 and T_2 that are S-adjacent for some set $S \subseteq [n]$ (respectively, $S \subseteq [n]$, with $|S| \leq \frac{k}{2z-1}$) where $\sum_{i \in S} w_i(T_1) - \sum_{i \in S} w_i(T_2) > \alpha$. If we let T_1' and T_2' represent the corresponding tournaments that determined the values of T_1 and T_2 from $r(\cdot)$ respectively, and let $S' = \cup_{i \in S} G(i)$, we can conclude $\sum_{i \in S'} r_i(T_1') - \sum_{i \in S'} r_i(T_2') > \alpha$, contradicting the fact that r is ∞-SNM-α (respectively, that r is k-SNM-α, as $|S'| = |S| \cdot (2z-1)$ and $|S| \leq \frac{k}{2z-1}$). ◀

With Lemma 28 in hand, we're very close to our goal. In particular, we've now shown that ∞-SNM-α rules exist for all n if and only if ∞-SNM-α rules exist for all n *which additionally satisfy the constraints in $LP_1(T, \alpha, z)$ for all $z \in \mathbb{N}_+$*. Note that as $z \to \infty$, the constraints of $LP_1(T, 1/2, z)$ approach those of $SLP(T)$. So one might reasonably expect that $SLP(T)$ can be used in place of $LP_1(T, 1/2, z)$ above, specifically when $\alpha = 1/2$. Indeed, this is the case (and the only place where we use Proposition 22).

▶ **Theorem 29.** *There exists an ∞-SNM-1/2 tournament rule for all n if and only if the* SLP *Tournament Rule is ∞-SNM-1/2 for all n.*

Moreover, if the SLP *Tournament Rule is* not *∞-SNM-1/2 for all n, there exists a pair of integers $k, n < \infty$ such that no k-SNM-1/2 Tournament Rule exists on n teams.*

Proof. The proof follows from a proper application of Lemma 28 and Proposition 22. Suppose towards contradiction that the SLP Tournament Rule is not k-SNM-1/2 for some k, n. This implies that there must be some tournaments $T, T' \in T_n$ and manipulating set S which verify this fact by gaining probability $c > \frac{1}{2}$. Call $A(T), b(T)$ be the constraint matrix and vector of $SLP(T)$, respectively, when written in standard form (i.e. $A(T)$ has n rows, corresponding to the n non-trivial constraints in $SLP(T)$. $b(T)$ is just the n-dimensional vector of all 1/2s).

Now apply Proposition 22 with $A := A(T)$ and $b := b(T)$, with $\delta = \frac{c - \frac{1}{2}}{4}$, and let $\varepsilon(T)$ be the promised ε. Do the same for T', and set $\varepsilon = \min\{\varepsilon(T), \varepsilon(T')\}$. Pick now a sufficiently large z such that $\frac{z}{2z-1} - \frac{1}{2} \leq \varepsilon$ (such a z exists as $\varepsilon > 0$).

Now, observe that any feasible solution $\vec{x}$ for $LP_1(T, 1/2, z)$ satisfies $A(T) \cdot \vec{x} \geq \vec{b} - \varepsilon\vec{1}$ (and any feasible solution $\vec{y}$ for $LP_1(T', 1/2, z)$ satisfies $A(T') \cdot \vec{y} \geq \vec{b} - \varepsilon\vec{1}$). If there is an ∞-SNM-$1/2$ tournament rule (respectively, k-SNM-$1/2$ tournament rule, for k to be chosen later), Lemma 28 tells us that there exists an ∞-SNM-$1/2$ (respectively, $\frac{k}{2z-1}$-SNM-$1/2$) tournament rule y such that $y(T)$ is feasible for $LP_1(T, 1/2, z)$ and $y(T')$ is feasible for $LP_1(T', 1/2, z)$. So we know $A(T) \cdot y(T) \geq \vec{b} - \varepsilon \cdot \vec{1}$, and also that $A(T') \cdot y(T') \geq \vec{b} - \varepsilon \cdot \vec{1}$. Proposition 22 then allows us to conclude that $|y(T) - w(T)|_1 \leq \delta$, and also that $|y(T') - w(T')|_1 \leq \delta$. But now we are ready to derive a contradiction and claim that in fact $y(\cdot)$ is not ∞-SNM-$1/2$. Indeed, we know that

$$\sum_{v \in S} w_v(T) - w_v(T') \geq c,$$

by definition of S, T, T'. But from the triangle inequality, we get that:

$$\sum_{v \in S} w_v(T) - y_v(T) \leq \sum_{v \in S} |w_v(T) - y_v(T)| \leq |w(T) - y(T)|_1 \leq \delta,$$

$$\sum_{v \in S} y_v(T') - w_v(T') \leq \sum_{v \in S} |w_v(T') - y_v(T')| \leq |w(T') - y(T')|_1 \leq \delta.$$

Summing these three equations then yields:

$$\sum_{v \in S} y(T)_v - y(T')_v + 2\delta \geq c \quad \Rightarrow \quad \sum_{v \in S} y(T)_v - y(T')_v \geq \frac{c + \frac{1}{2}}{2} > \frac{1}{2}.$$

This contradicts that $y(\cdot)$ was ∞-SNM-$1/2$ (and contradicts that $y(\cdot)$ is $\frac{k}{2z-1}$-SNM-$1/2$, as long as $|S| \leq \frac{k}{2z-1}$, or $k \geq |S|(2z-1)$. Note that k can indeed be defined after z and S), as now S can manipulate from T' to T and gain $> 1/2$. ◀

To briefly recap the entire proof of Theorem 29: we first showed that the existence of ∞-SNM-α rules imply the existence of specific kinds of ∞-SNM-α rules (those which satisfy $LP_1(T, \alpha, z)$ for all $z \in \mathbb{N}_+$). Note that we relied on the existence of ∞-SNM-α rules for $n' \gg n$ in order to show the existence of our specialized ∞-SNM-α rules for n. Then, we showed that for $\alpha = 1/2$, the existence of specialized rules implies that a particular rule (the SLP Tournament Rule) is ∞-SNM-$1/2$ (and the fact that the SLP Tournament Rule is well-defined is the focus of Section 4).

Now, we make use of Theorem 29 by proving that the SLP Tournament Rule is not ∞-SNM-$1/2$.

▶ **Lemma 30.** *The* SLP *Tournament Rule is not* ∞-SNM-$1/2$.

Proof. See Figure 3 where two $\{B, C, E\}$-adjacent tournaments are evaluated under the SLP Tournament Rule. The three teams $\{B, C, E\}$ together have probability $4/9$ under T, but 1 under T', and therefore gain $5/9 > 1/2$ by manipulating. So the rule is not ∞-SNM-$1/2$.

Note that in order to verify that we have computed the SLP Tournament Rule correctly on T and T', the reader need only verify (in each graph) that the probabilities sum to 1, and the SLP constraints: for all i, $\sum_{j \in \delta_T^-(i)} p_j + p_i/2 \geq 1/2$. By Proposition 26, any such solution is the unique optimum, and therefore output by the SLP Tournament Rule. ◀

Proof of Theorem 1. The proof of Theorem 1 now follows immediately from Theorem 29 and Lemma 30. ◀

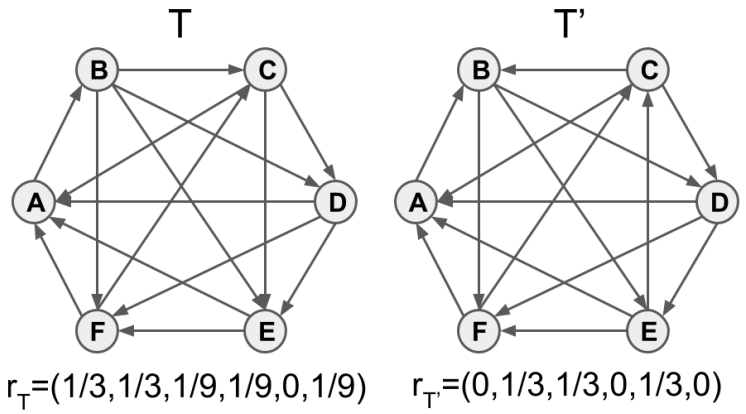


Figure 3 The tournaments T, T' are $\{B, C, E\}$ adjacent. The distribution of winners as prescribed by SLP is provided below each tournament, where the teams are presented alphabetically. Note that under T, the collusion wins with chance $4/9$ but in T' they win with probability 1, gaining $5/9$.

4.1 Concluding Thoughts on Lower Bounds

We emphasize again that our lower bounds reduce the problem of determining existence of Condorcet-consistent ∞-SNM-$1/2$ tournament rules for large $n' \gg n$ to the problem of determining whether the specific SLP Tournament Rule is ∞-SNM-$1/2$ for small n. In particular, now that we have a specific n, k for which the SLP Tournament Rule is not k-SNM-$1/2$ on n teams, we can backtrack through Theorem 29 and recover a specific k', n' for which no k'-SNM-$1/2$ (and therefore no k'-SNM-$\frac{k'-1}{2k'-1}$) Tournament Rule exists on n' teams:

In our example, there are $n = 6$ teams, and the SLP Tournament Rule is not better than 3-SNM-$5/9$. So we may take $c = 5/9$ in the proof of Theorem 29, which results in $\delta = 1/72$. Note that ε is now a function of δ via Proposition 22, and is ≈ 0.0016, so we'd set $z := \lceil \frac{1/2+\varepsilon}{2\varepsilon} \rceil = 157$. This therefore rules out the possibility of a tournament rule that is 939-SNM-$1/2$ for 1878 teams. While of course portions of the proof of Theorem 29 could be optimized to yield a smaller k', n', the point we are trying to make is that there could very well be k-SNM-$1/2$ tournament rules for n teams for quite large values of k, n, and there is virtually no hope of uncovering the non-existence for extremely large k via exhaustive search – recall that the space of tournament rules is all functions from the $2^{\binom{n}{2}}$ different complete directed graphs on n teams to the n-dimensional simplex (indeed, the authors had no luck via exhaustive search, and numerous rules appeared k-SNM-$1/2$ for small n in simulations). However, the machinery developed in this section allows us to brute-force search for manipulations of a *single* tournament, which happened to resolve for $k = 3, n = 6$ and conclude our desired claim, which would have required a significantly larger exhaust for *significantly* larger k', n'.

Finally, we note that it may be tempting to use our machinery, almost as is, to rule out Condorcet-consistent ∞-SNM-α rules for $\alpha > 1/2$. In particular, it is tempting to conclude that because Figure 3 exhibits that the SLP Tournament Rule is no better than 3-SNM-$5/9$, that there should not be an ∞-SNM-$5/9$ tournament rule for all n. Note, however, that the SLP is really special for $\alpha = 1/2$. Indeed, if we were to replace $1/2$ with $4/9$ in the SLP, we would (for instance) no longer have a unique solution. Therefore, we'd lose the well-defined SLP Tournament Rule, and still have to do a broad exhaustive search, and significantly new ideas would be needed to get leverage out of this. Still, while our current tools only

preclude rules which are ∞-SNM-1/2 (and ever so slightly more: ∞-SNM-.50016 via similar reasoning to the previous paragraph), it is reasonable to expect that our general approach (e.g. Lemma 28) may help rule out the existence of SNM-α tournament rules for $\alpha > 1/2$.

5 Less Manipulable Tournament Rules via SLP

In this section we prove Theorem 2: an ∞-SNM-2/3 tournament rule exists (for all n). Fortunately, a lot of the work has been done in Sections 3 and 4 in the form of our understanding of how feasible solutions to $SLP(T)$ and $LP_0(T, \beta, k)$ relate. We first show how any k-SNM-α rule that is a valid solution to $LP_0(T, \beta, k)$ (for some β) can be transformed into a k-SNM-$\frac{\alpha}{\alpha-\beta+1}$ rule. This transformation yields a stronger tournament rule if $\alpha \geq \beta$. Our proof then exploits the fact that an optimal solution to $SLP(T)$ satisfies $LP_0(T, 1/2, k)$ by design and, trivially, is ∞-SNM-1. The naive upper bound of $\alpha \leq 1$ suffices to yield a ∞-SNM-2/3 rule (previously no ∞-SNM-< 1 rule is known). Moreover, this reduction now allows any improved bounds (even if $\gg 2/3$) on the manipulability of the SLP Tournament Rule to imply tournament rules which are ∞-SNM-α for $\alpha < 2/3$. Below is the main lemma of this section.

▶ **Lemma 31** (Augmentation Lemma). *Let tournament rule $r(\cdot)$ be such that $r(T)$ satisfies $LP_0(T, \beta, k)$ for all tournaments T, and be k-SNM-α. Then a tournament rule $w(\cdot)$ exists which is k-SNM-$\frac{\alpha}{\alpha-\beta+1}$.*

Proof. Consider the following rule: pick a $c \in [0, 1]$ (to be chosen later). Set $w_i(T) = r_i(T) \cdot c + \frac{1-c}{n}$, if T does not have a Condorcet winner. If T has a Condorcet winner, allocate probability 1 to the Condorcet winner.

To evaluate the manipulability of this rule, first consider a manipulating set which creates a Condorcet winner. The total probability gained is at most $\beta \cdot c + (1 - c)$. To see this, observe that because the set can create a Condorcet winner, they must have total probability at least β under $r(\cdot)$ (and therefore at least $\beta \cdot c$ after scaling down by c).

Now consider a manipulating set that does not create a Condorcet winner. Then there is certainly no Condorcet winner in T', and so the extra $(1 - c)$ probability mass is still allocated uniformly (and the set gains nothing here). So the set can only gain what they would by manipulating under $r(\cdot)$ (scaled down by c), which is at most $\alpha \cdot c$.

To minimize $\max\{\alpha c, \beta c + (1 - c)\}$, set $c := \frac{1}{\alpha-\beta+1}$. This results in $w(\cdot)$ being k-SNM-$\frac{\alpha}{\alpha-\beta+1}$. ◀

Proof of Theorem 2. By construction the SLP Tournament Rule is Condorcet-consistent, and is feasible for $LP_0(T, \frac{1}{2}, k)$ for all k. Thus the SLP Tournament Rule satisfies the requirements of the augmentation lemma for $\beta = \frac{1}{2}$ and $\alpha = 1$ (as all rules are k-SNM-1 for all k) for all k, so Lemma 31 results in an ∞-SNM-2/3 rule. ◀

At this point the experienced reader may wonder about other useful properties of the SLP Tournament Rule. In Appendix B we show that the SLP Tournament Rule is *not* monotone.

6 Cover-Consistent Tournament Rules

In this section we shift gears and return to 2-SNM-α tournaments. We extend the results of [22] not in the direction of larger k or smaller α, but towards a more stringent requirement than Condorcet-consistence (cover-consistence). The main result of this section is Theorem 4, which develops a new tournament rule which is cover-consistent and 2-SNM-1/3 (the smallest α possible, by Lemma 17). We call our rule Randomized-King-of the Hill and define it below.

Algorithm 1 Pseudocode for the Randomized-King-of-the-Hill Tournament Rule.

Input: A tournament graph $T = (V, E)$ on n teams.
Output: A winning team $i \in V$.
repeat
  Choose a team $j \in V$ uniformly at random ;
  if *(j is a Condorcet winner);*
  then
    return j;
  end
  $V \leftarrow V \setminus \{j \cup \delta_T^+(j)\}$;
until;

▶ **Definition 32** (Randomized-King-of-the-Hill)**.** *The* Randomized-King-of-the-Hill *Tournament Rule (RKotH) starts every step by first checking whether there is a Condorcet winner among the remaining teams. If so, that team is declared the winner. If not, it picks a uniformly random remaining team i (which we'll call the* prince*) and removes team i and all teams which lose to i. Algorithm 1 provides pseudocode.*

The main distinction we'll emphasize between RSEB and RKotH is that RKotH is cover-consistent (Lemma 33 below), while RSEB is not (Observation 19). We later show that RKotH is also 2-SNM-1/3, just like RSEB.

▶ **Lemma 33.** *RKotH is cover-consistent.*

Proof. Consider any two teams u, v where u covers v. If RKotH is to possibly output v, the team u must be removed at some round where team x is selected. If at the start of this round, v has already been removed, then v will clearly not be declared the winner. If at the start of this round, v has not already been removed, then v is removed this round because v loses to x (as x beats u and u covers v). Therefore, v can never be declared the winner by RKotH. ◀

In fact, RKotH satisfies an even stronger property than cover-consistency. Before stating this we need to introduce some definitions.

▶ **Definition 34** (Sub tournament)**.** *A* sub tournament *of a tournament T with respect to a set of teams S is the tournament induced by the games between teams in S.*

▶ **Definition 35** (Transitive tournaments)**.** *A tournament T is* transitive *if there are no directed cycles.*

▶ **Definition 36** (Banks set)**.** *Team v is a Banks winner of tournament T if there exists a maximal (with respect to inclusion) transitive sub tournament T' of T where v is the Condorcet-winner. The* Banks set *of a tournament T is the set of Banks winners of the tournament.*

▷ Claim 37. The Banks set of a tournament is a subset of the set of uncovered teams of the tournament.

Proof. We will show the contrapositive. Consider a team v that is covered by some other team u. No maximal transitive sub tournament of T can have v as its Condorcet-winner because u beats v and everyone v beats and hence can always be added on top of v, contradicting the maximality of the sub tournament. ◁

We can now prove Lemma 5, which states that the Banks set of a tournament T is *exactly* the set of teams RKotH can declare as winner.

Proof of Lemma 5. First, we argue that any Banks winner can be output by RKotH with non-zero probability. Indeed, consider any Banks winner v, and let T' denote the maximal transitive subtournament in which v is the Condorcet winner. Name the teams in $T' \setminus \{v\}$ as $u_1, \ldots, u_k$, where u_i beats u_j for all $j < i$ (and v beats u_i for all i). Now, because T' is a maximal transitive subtournament, there does not exist any w which beats all teams in T' (otherwise we could add w to the subtournament, witnessing non-maximality). Now, consider an execution of RKotH which first selects princes in order of u_i (u_1, then u_2, etc.) and then finally v. First, observe that each u_i has not yet been eliminated by the time we hope they are selected (by definition of u_i). Second, observe that every team $w \notin T'$ must be eliminated by the end, because they lose to some team in T'. Therefore, after this execution, v is the only remaining team, and crowned champion.

It remains to show that any team in the support of RKotH is always a Banks winner. Out of all the executions of RKotH where v wins, consider one that goes through the most princes before v is picked. We claim this will be a maximal transitive sub tournament. Let $P = \{p_1, ..., p_k\}$ be the set of princes used by the algorithm in that order. Since v wins under those princes, it must beat every team in P. Moreover, since the algorithm first picks p_1, then p_2, and so on, it must be the case that p_i beats p_j for $i > j$. Otherwise p_i would be eliminated on the step where p_j was selected as the prince. Therefore, the sub tournament $v \cup P$ is transitive, and has v as its Condorcet-winner.

Consider any team $x \notin v \cup P$. If x beat teams $p_1, ..., p_i$ and lost to teams $p_{i+1}, ..., v$ for some $i \in \{0, ..., k+1\}$, then an execution that places x between p_i, p_{i+1} would be feasible. It would still output v but run for one more step than P, contradicting the maximality of P. Therefore, $v \cup P$ is a maximal transitive sub tournament where v is a Condorcet-winner, implying v is in the Banks set. ◀

We will also use the fact that RKotH is monotone in our remaining proof. Below (and for the remainder of this section), we'll refer to a *prince* as the most recently selected remaining team, and we'll refer to an *execution* of RKotH as simply an ordering over potential princes (to be selected if they haven't yet been eliminated when their turn comes).

▶ **Lemma 38.** *RKotH is monotone. That is, if T, T' are $\{u, v\}$-adjacent and u beats v in T, then $r_u(T) \geq r_u(T')$.*

Proof. Consider any execution of RKotH and consider the first time that either u or v is prince (observe that prior to this, the edge between u and v is never queried, so the execution on T and T' is identical). If either u or v is already eliminated, then it doesn't matter whether u beats v or vice versa, and the outcome is the same. Otherwise, if u is the prince and u beats v, then there is a chance that u wins. If u loses to v, then u is eliminated immediately. If v is the prince and u beats v, then there is a chance that u wins. If u loses to v, then u is eliminated immediately. Therefore, for every execution, if u wins in T', u also wins in T, and the lemma holds. ◀

The rest of this section is devoted to proving that RKotH is 2-SNM-1/3. The key approach of our proof is the following: consider any round in which both u and v still remain. The next team selected as prince might beat both u and v (in which case the outcome between u and v is never queried), lose to both u and v (in which case the outcome has not yet been queried), or beat exactly one of $\{u, v\}$ (in which case again the outcome between u and v is

never queried). The only event in which we ever query the outcome of the match is when one of $\{u, v\}$ is selected as prince while the other remains (and even then, the outcome only matters if some teams remain which beat exactly one of $\{u, v\}$). So the key approach in the proof is a coupling argument between different possible executions of RKotH (some of which make the match between u and v irrelevant, and some of which cause $\{u, v\}$ to prefer the match turn one way or the other).

Proof of Theorem 4. In order to show the rule is 2-SNM-1/3, consider two teams u, v who are trying to collude in a given tournament T. Suppose wlog that u beats v in T and let T' be the $\{u, v\}$-adjacent tournament to T where v beats u.

To begin the analysis, we first introduce some notation. Let S be the subset of teams which either beat at least one of u or v, or are u or v. For a given execution of RKotH let x denote the first prince in S on tournament T. Observe first that there must be a prince in S at some point (otherwise neither u nor v is ever eliminated), and also that x is the first prince in S on tournament T' as well (for the same execution). Let also X denote the set of princes *strictly before* x was chosen, and $Y(X)$ denote the set of un-eliminated teams after the set X of princes. We first observe that, conditioned on X, the next prince is a uniformly random element of $Y(X) \cap S$.

▶ **Lemma 39.** *For all $X \subset [n] \setminus S$, conditioned on the set X being princes so far, and the next prince being an element of S, the next prince is a uniformly random element of $Y(X) \cap S$.*

Proof. For all $X \subseteq [n] \setminus S$, conditioned on the set X being princes so far, the next prince is a uniformly random element of $Y(X)$, so each element of $Y(X) \cap S$ is selected with equal probability. ◀

The main step in the proof is the following lemma, which claims that after conditioning on X, the difference between T and T' in terms of whether one of $\{u, v\}$ wins under RKotH is small.

▶ **Lemma 40.** *For all X, let $r_{u,v}(T, X)$ denote the probability that RKotH selects a winner in $\{u, v\}$, conditioned on X being exactly the set of princes before the first prince in S. Then for all X, $|r_{u,v}(T', X) - r_{u,v}(T, X)| \leq 1/3$.*

Proof. We consider a few possible cases, conditioned on the structure of $Y(X) \cap S$, and which team is the next prince. To aid in formality, we'll use the notation $r_{u,v}(T, X|E)$ to denote the probability that RKotH selects a winner in $\{u, v\}$ on tournament T conditioned on exactly the set X of princes before the first prince in S and event E.

Case One: $Y(X) \cap S = \{u, v\}$. In this case, certainly u or v will win in tournament T and T'. This is because all remaining teams lose to both u and v, so whoever wins the match between u and v is a Condorcet winner among the remaining teams and will therefore win. So if E_1 denotes the event that $Y(X) \cap S = \{u, v\}$, we have that $r_{u,v}(T, X|E_1) = r_{u,v}(T', X|E_1)$.

Case Two: $|Y(X) \cap S| > 2$, next prince $\notin \{u, v\}$. In this case, at least one of $\{u, v\}$ are eliminated immediately, and the match result is never queried. Therefore, the result is the same under T and T'. So if E_2 denotes the event that $|Y(X) \cap S| > 2$ and the next prince is not in $\{u, v\}$, we have that $r_{u,v}(T, X|E_2) = r_{u,v}(T', X|E_2)$.

Case Three: $|Y(X) \cap S| > 2$, next prince is u. In this case, we claim it is *always better* for $\{u, v\}$ to be in T' (v beats u) versus T. To see this, first consider that maybe some remaining element of $Y(X) \cap S$ beats u. Then u certainly will not win. If u beats v, then v also will certainly not win. But if v beats u, then maybe v can win. If no remaining element of $Y(X) \cap S$ beats u, then either u will win in T, or v will win in T' (because all teams aside from u and v are eliminated). So if E_3 denotes the event that $|Y(X) \cap S| > 2$ and the next prince is u, we have that $r_{u,v}(T', X|E_3) \geq r_{u,v}(T, X|E_3)$.

Case Four: $|Y(X) \cap S| > 2$, next prince is v. This case is symmetric to the above, and it is always better for $\{u, v\}$ to be in T versus T'. So if E_4 denotes the event that $|Y(X) \cap S| > 2$ and the next prince is v, we have that: $r_{u,v}(T, X|E_4) \geq r_{u,v}(T', X|E_4)$.

So to conclude, we've seen that if $|Y(X) \cap S| = 2$, then the outcome is the same under T and T', so the lemma statement clearly holds for any X with $|Y(X) \cap S| = 2$. If $|Y(X) \cap S| > 2$, then there is *exactly* one choice for the next team which *may* cause $\{u, v\}$ to prefer T to T' (and vice versa). By Lemma 39, the next prince is drawn uniformly at random from $|Y(X) \cap S|$, so this element is selected with probability at most $1/3$. Formally, for any X with $|Y(X) \cap S| > 2$ we have:

$$\begin{aligned} r_{u,v}(T, X) - r_{u,v}(T', X) = &\Pr[E_2] \cdot (r_{u,v}(T, X|E_2) - r_{u,v}(T', X|E_2)) \\ &+ \Pr[E_3] \cdot (r_{u,v}(T, X|E_3) - r_{u,v}(T', X|E_3)) \\ &+ \Pr[E_4] \cdot (r_{u,v}(T, X|E_4) - r_{u,v}(T', X|E_4)) \\ \leq &\Pr[E_2] \cdot 0 + \Pr[E_3] \cdot 0 + \Pr[E_4] \cdot 1 \leq \frac{1}{|Y(X) \cap S|} \leq 1/3. \end{aligned}$$

Similar inequalities hold for $r_{u,v}(T', X) - r_{u,v}(T, X)$ but with the role of E_3 and E_4 swapped, allowing us to conclude that indeed $|r_{u,v}(T, X) - r_{u,v}(T', X)| \leq 1/3$. ◀

The rest of the proof now follows easily. Below, if $r_{u,v}(T)$ denotes the probability that either u or v wins under RKotH for tournament T, and $p(X)$ denotes the probability that X is exactly the set of princes before the first prince in S is selected (under tournament T), we have:

$$|r_{u,v}(T) - r_{u,v}(T')| \leq \sum_X p(X) \cdot |r_{u,v}(T, X) - r_{u,v}(T', X)| \leq 1/3.$$ ◀

While RKotH and RSEB are optimal tournament rules against collusions of size 2, their effectiveness disappears as the collusions become larger. In particular, in Appendix B we show a specific tournament against both rules for which large collusions can increase their odds of winning up to close to 1.

7 Conclusion

We extend work of [22] in three different directions: First, we refute their main conjecture (Theorem 1, Sections 3 and 4). Next, we design the first Condorcet-consistent tournament rule which is ∞-SNM-(< 1) (Theorem 2, Sections 3 and 5). Finally, we design a new tournament rule (RKotH) which is 2-SNM-1/3 (just like RSEB), but which is also cover-consistent (Theorem 4, Section 6).

Reiterating from Section 1, the main appeal of our results is clearly theoretical, and some of this appeal comes from the process itself. For example, Theorem 29 reduces the search for a Condorcet-consistent ∞-SNM-1/2 rule to determining whether or not the SLP Tournament Rule is ∞-SNM-1/2. Additionally, the same tools developed in Section 3 proved useful both for proving lower bounds and designing new tournament rules, suggesting that these tools should be useful in future works as well.

One clear direction for future work, now that the main conjecture of [22] is refuted, is to understand what the minimum α is such that an ∞-SNM-α tournament rule exists. It is also interesting to understand how large k needs to be in order for the [22] conjecture to be false. Our work does not rule out the existence of a 3-SNM-2/5 tournament rule, yet we also do not know of any 3-SNM-2/5 rule (nor even a 3-SNM-1/2 rule). More generally, our work contributes to the broad agenda of understanding the tradeoffs between incentive compatibility and quality of winner selected in tournament rules, and there are many interesting problems in this direction.

References

1 Alon Altman and Robert Kleinberg. Nonmanipulable Randomized Tournament Selections. In *AAAI Conference on Artificial Intelligence*, 2010. URL: https://www.aaai.org/ocs/index.php/AAAI/AAAI10/paper/view/1703.

2 Alon Altman, Ariel D. Procaccia, and Moshe Tennenholtz. Nonmanipulable Selections from a Tournament. In *Proceedings of the 21st International Joint Conference on Artifical Intelligence*, IJCAI'09, pages 27–32, San Francisco, CA, USA, 2009. Morgan Kaufmann Publishers Inc. URL: http://dl.acm.org/citation.cfm?id=1661445.1661451.

3 Kenneth J. Arrow. A Difficulty in the Concept of Social Welfare. *Journal of Political Economy*, 58(4):328–346, 1950. URL: http://www.jstor.org/stable/1828886.

4 J. S. Banks. Sophisticated voting outcomes and agenda control. *Social Choice and Welfare*, 1(4):295–306, December 1985. doi:10.1007/BF00649265.

5 Florian Brandl, Felix Brandt, and Hans Georg Seedig. Consistent Probabilistic Social Choice. *CoRR*, abs/1503.00694, 2015. arXiv:1503.00694.

6 Felix Brandt, Vincent Conitzer, Ulle Endriss, Jérôme Lang, and Ariel D Procaccia. *Handbook of Computational Social Choice*. Cambridge University Press, 2016.

7 Laszlo Csato. European qualifiers 2018 FIFA World Cup qualification can be manipulated, 2017. URL: https://mpra.ub.uni-muenchen.de/id/eprint/83437.

8 P. C. Fishburn. Probabilistic Social Choice Based on Simple Voting Comparisons. *The Review of Economic Studies*, 51(4):683–692, October 1984. doi:10.2307/2297786.

9 Peter C. Fishburn. Condorcet Social Choice Functions. *SIAM Journal on Applied Mathematics*, 33(3):469–489, 1977. doi:10.1137/0133030.

10 David C. Fisher and Jennifer Ryan. Optimal Strategies for a Generalized "Scissors, Paper, and Stone" Game. *The American Mathematical Monthly*, 99(10):935–942, 1992. doi:10.1080/00029890.1992.11995957.

11 Allan Gibbard. Manipulation of voting schemes: a general result. *Econometrica*, 41(4):587–601, 1973.

12 Allan Gibbard. Manipulation of Schemes that Mix Voting with Chance. *Econometrica*, 45(3):665–681, 1977. URL: http://www.jstor.org/stable/1911681.

13 Olivier Hudry. A note on "Banks winners in tournaments are difficult to recognize" by G. J. Woeginger. *Social Choice and Welfare*, 23(1):113–114, August 2004. doi:10.1007/s00355-003-0241-y.

14 Michael P. Kim, Warut Suksompong, and Virginia Vassilevska Williams. Who Can Win a Single-Elimination Tournament? In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, pages 516–522, 2016. URL: http://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/12194.

15 Michael P. Kim and Virginia Vassilevska Williams. Fixing Tournaments for Kings, Chokers, and More. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 561–567, 2015. URL: http://ijcai.org/Abstract/15/085.

16 G. Laffond, J.F. Laslier, and M. Le Breton. The Bipartisan Set of a Tournament Game. *Games and Economic Behavior*, 5(1):182–201, 1993. doi:10.1006/game.1993.1010.

17 J.F. Laslier. *Tournament Solutions and Majority Voting*. Studies in Economic Theory (Berlin, Germany), 7. Springer, 1997. URL: https://books.google.com/books?id=vYbGAAAAIAAJ.

18 Stephen B. Maurer. The King Chicken Theorems. *Mathematics Magazine*, 53(2):67–80, 1980. URL: http://www.jstor.org/stable/2689952.

19 Nicholas R. Miller. A New Solution Set for Tournaments and Majority Voting: Further Graph-Theoretical Approaches to the Theory of Voting. *American Journal of Political Science*, 24(1):68–96, 1980. URL: http://www.jstor.org/stable/2110925.

20 Marc Pauly. Can strategizing in round-robin subtournaments be avoided? *Social Choice and Welfare*, 43(1):29–46, 2014. URL: http://EconPapers.repec.org/RePEc:spr:sochwe:v:43:y:2014:i:1:p:29-46.

21 Mark Allen Satterthwaite. Strategy-proofness and Arrow's conditions: Existence and Correspondence theorems for voting procedures and social welfare functions. *Journal of Economic Theory*, 10(2):187–217, 1975.

22 Jon Schneider, Ariel Schvartzman, and S. Matthew Weinberg. Condorcet-Consistent and Approximately Strategyproof Tournament Rules. In *8th Innovations in Theoretical Computer Science Conference, ITCS 2017, January 9-11, 2017, Berkeley, CA, USA*, pages 35:1–35:20, 2017. doi:10.4230/LIPIcs.ITCS.2017.35.

23 Kenneth A. Shepsle and Barry R. Weingast. Uncovered Sets and Sophisticated Voting Outcomes with Implications for Agenda Institutions. *American Journal of Political Science*, 28(1):49–74, 1984. URL: http://www.jstor.org/stable/2110787.

24 Isabelle Stanton and Virginia Vassilevska Williams. Rigging Tournament Brackets for Weaker Players. In *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*, pages 357–364, 2011. doi:10.5591/978-1-57735-516-8/IJCAI11-069.

25 Virginia Vassilevska Williams. Fixing a Tournament. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA, July 11-15, 2010*, 2010. URL: http://www.aaai.org/ocs/index.php/AAAI/AAAI10/paper/view/1726.

26 Gerhard J. Woeginger. Banks winners in tournaments are difficult to recognize. *Social Choice and Welfare*, 20(3):523–528, June 2003. doi:10.1007/s003550200197.

A Missing proofs from Section 2.2

To prove Proposition 21, we first need some additional machinery.

▶ **Definition 41** (Pfaffian). *The* Pfaffian *of an $n \times n$ skew-symmetric matrix A (when $n = 2k$ is even) is defined as follows. First, let Π be the set of all partitions of $[2k]$ into pairs without regard to order. If we write an element $\alpha \in \Pi$ as $\{(i_1, j_1), (i_2, j_2), \cdots (i_k, j_k)\}$ with $i_\ell < j_\ell$ for all ℓ and $i_1 < i_2 < \ldots < i_k$, then let π_α denote the permutation with $\pi_\alpha(2\ell - 1) = i_\ell$ and $\pi_\alpha(2\ell) = j_\ell$ (i.e. π_α sorts elements in the order $i_1, j_1, i_2, j_2, \ldots$). Let $A_\alpha := sgn(\pi_\alpha) \prod_{\ell=1}^{k} A_{i_\ell, j_\ell}$.[4] Then the Pfaffian of A, denoted by $Pf(A)$, is equal to $\sum_{\alpha \in \Pi} A_\alpha$.*

▶ **Theorem 42.** *When n is even, any $n \times n$ skew-symmetric matrix A satisfies $Det(A) = (Pf(A))^2$.*

Proof of Proposition 21. We will first prove the case when n is even, using Theorem 42. We first observe that as A is unit skew-symmetric, $|A_{ij}| = 1$ for all $i \neq j$. Therefore, for all $\alpha \in \Pi$, $|A_\alpha| = 1$. So the Pfaffian of A is the sum of $|\Pi|$ terms, each of which are ± 1. Therefore, if we can show that $|\Pi|$ is odd, the Pfaffian must be non-zero, Theorem 42 implies that $Det(A) \neq 0$ as well (meaning that A has full rank).

[4] Recall that the sign of a permutation π, denoted here by $sgn(\pi)$ is equal to the number of inversions in π (that is, the number of pairs $i < j$ with $\pi(i) > \pi(j)$).

It is straight forward to count the number of elements in Π: first, there are $n-1$ choices for the partner of 1. Then, there are $n-3$ choices for the partner of the smallest unpartnered element. After choosing the first ℓ pairs, there are $n-2\ell-1$ choices for the partner of the smallest unpartnered element. So there are $(n-1)\cdot(n-3)\ldots 5\cdot 3$ elements of Π, which is an odd number.

Now we consider the case where n is odd. Observe first that the submatrix of A created by removing the last row and last column is a unit skew symmetric matrix with even dimension, and therefore has rank $n-1$. Therefore, the rank of A is also at least $n-1$. By Jacobi's theorem, all skew symmetric matrices of odd dimension have determinant zero (and are therefore not of full rank). Therefore, the rank must be exactly $n-1$. ◀

Proof of Proposition 22. For a given $\delta > 0$, let Y be the set of all vectors in the unit hypercube with ℓ_1 distance strictly less than δ from some element of P_0 (i.e. there exists an element in P_0 within ℓ_1 distance strictly less than δ). It is easy to see that Y is an open set. Now let $S = [0,1]^n \setminus Y$. Note that S is closed and bounded, and therefore compact.

Definte now the function $f(\vec{y}) = \max\{0, \max_{i\in[n]} \vec{b}_i - (A\cdot\vec{y})_i\}$. First observe that $f(\cdot)$ is continuous, as it is a composition (via maximums, subtractions, etc.) of affine functions. Observe also that $\vec{y} \in P_\varepsilon$ for all $\varepsilon \geq f(\vec{y})$, and that $\vec{y} \notin P_\varepsilon$ for all $\varepsilon < f(\vec{y})$. We now consider two possible cases for $\inf_{\vec{y}\in S}\{f(\vec{y})\}$, and find our desired ε.

- Case 1: $\inf_{\vec{y}\in S}\{f(\vec{y})\} = 0$. As S is compact, f achieves its infimum over S. Therefore, there exists a $\vec{y} \in S$ with $f(\vec{y}) = 0$. Let's parse what this means. First, as $f(\vec{y}) = 0$, we know that $\vec{y} \in P_0$. But also, $P_0 \subseteq Y$, and $S \cap Y = \emptyset$. So any $\vec{y} \in S$ cannot also be in P_0, which means $f(\vec{y}) > 0$ for all $\vec{y} \in S$, meaning that we can't have $\inf_{\vec{y}\in S}\{f(\vec{y})\} = 0$ after all.
- Case 2: $\inf_{\vec{y}\in S}\{f(\vec{y})\} = c > 0$. Then let $\varepsilon = c/2$. Now, observe that all elements of S are *not* in P_ε, as they all have $f(\vec{y}) > \varepsilon$.

So to wrap up, we must have $\inf_{\vec{y}\in S}\{f(\vec{y})\} = c > 0$, and if we set $\varepsilon = c/2$, then $S \cap P_\varepsilon = \emptyset$. Therefore, as $S \cap Y = [0,1]^n$, it must be the case that all of P_ε is contained in Y. But this is exactly the desired statement: all elements of P_ε have some point in P_0 within distance δ. So take this to be our desired ε. ◀

B Further properties of the proposed tournament rules

In this section we analyze further properties of the two main tournament rules discussed on this paper. First we show that the SLP tournament rule fails to satisfy monotonicity, a very natural and desirable property from the point of view of a tournament designer.

▷ Claim 43. SLP is not a monotone tournament rule.

Proof. Consider the tournament T and it's $\{3,5\}$-adjacent tournament T', both depicted in Figure 4. In T, 5 beats 3 originally and the SLP tournament rule awarded team 5 a .2 chance of winning. If instead 5 purposely throws its game to 3, the SLP tournament rule *rewards* team 5 by increasing its chance of winning to 1/3. Therefore the SLP tournament rule is not monotone. ◁

Next we show that the optimality of RSEB, RKotH against collusions of size 2 fails to translate to larger collusions. In particular, we show that the manipulability of both rules tends to 1 as the size of the collusion increases, ruling both of them out as candidates for ∞-SNM-α for constant $\alpha < 1$. First we define a family of tournaments that will be useful in showing lower bounds for both rules. These tournaments were introduced in [22].

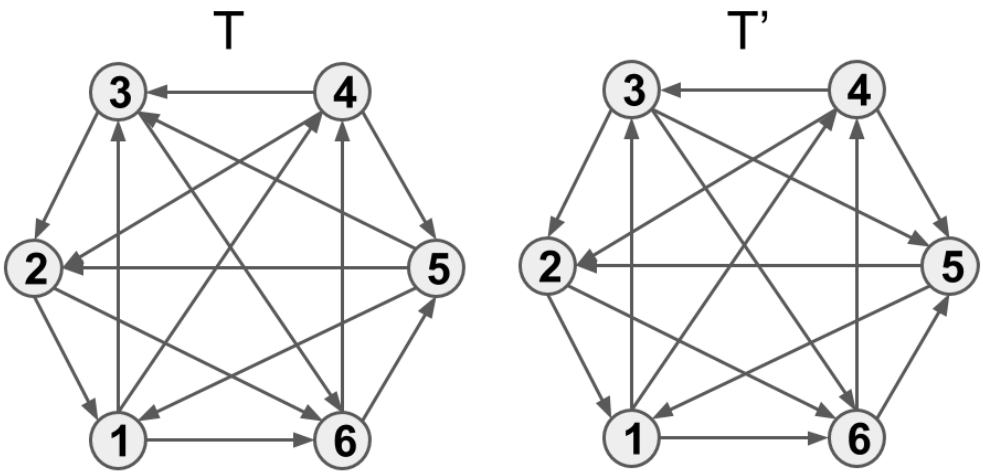


Figure 4 The tournament T can be unilaterally manipulated by team 5 to become the tournament T'.

▶ **Definition 44** (Kryptonite Tournament [22]). *A kryptonite tournament T on n teams has a* superman *team (wlog label it 1) who beats everyone except a* kryptonite *team (wlog label it n). Moreover, team n loses to every other team except 1. The outcomes of the matches between the remaining teams may be arbitrary.*

We now proceed with the main claim of this section.

▷ Claim 45. RKotH is no better than k-SNM-$(k-1)/(k+1)$ for any k. RSEB is no better than k-SNM-$(k-1)/k$ when $k+1$ is a power of 2.

Proof. Consider k and let $n = k+1$. Let T be any kryptonite tournament on n teams, with team 1 as the superman team and team n as the kryptonite team. In order for team 1 to win under RKotH, the rule must avoid selecting teams 1 and n. Under any other first choice of prince, RKotH will declare team 1 the winner. Therefore it declares team 1 the winner with probability $(k-1)/(k+1)$. However, if all teams but the superman collude they can make the kryptonite a Condorcet-winner. The collusion's combined odds of winning before were exactly $2/(k+1)$, so RKotH is at least k-SNM-$(k-1)/(k+1)$.

Now let $k+1$ be a power of two and consider the same T as before. In order for the superman team to win under RSEB it must avoid the kryptonite team in the first round. Therefore RSEB crowns the superman team winner with probability $(k-1)/k$. However, all other teams can form a collusion and turn the kryptonite team into a Condorcet-winner, increasing their winning mass by $(k-1)/k$. Therefore, RSEB is at least k-SNM-$(k-1)/k$ when $k+1$ is a power of 2. ◁

Span Programs and Quantum Space Complexity

Stacey Jeffery
CWI, Amsterdam, The Netherlands
QuSoft, Amsterdam, The Netherlands
https://homepages.cwi.nl/~jeffery/
jeffery@cwi.nl

Abstract

While quantum computers hold the promise of significant computational speedups, the limited size of early quantum machines motivates the study of space-bounded quantum computation. We relate the quantum space complexity of computing a function f with *one-sided error* to the logarithm of its *span program size*, a classical quantity that is well-studied in attempts to prove formula size lower bounds.

In the more natural *bounded error* model, we show that the amount of space needed for a unitary quantum algorithm to compute f with bounded (two-sided) error is lower bounded by the logarithm of its *approximate span program size*. Approximate span programs were introduced in the field of quantum algorithms but not studied classically. However, the approximate span program size of a function is a natural generalization of its span program size.

While no non-trivial lower bound is known on the span program size (or approximate span program size) of any concrete function, a number of lower bounds are known on the *monotone span program size*. We show that the approximate monotone span program size of f is a lower bound on the space needed by quantum algorithms of a particular form, called *monotone phase estimation algorithms*, to compute f. We then give the first non-trivial lower bound on the approximate span program size of an explicit function.

2012 ACM Subject Classification Theory of computation → Quantum computation theory

Keywords and phrases Quantum space complexity, span programs

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.4

Related Version A full version of the paper is available at https://arxiv.org/abs/1908.04232.

Funding Supported by an NWO WISE Fellowship, an NWO Veni Innovational Research Grant under project number 639.021.752, and QuantERA project QuantAlgo 680-91-03. SJ is a CIFAR Fellow in the Quantum Information Science Program.

Acknowledgements I am grateful to Tsuyoshi Ito for discussions that led to the construction of approximate span programs from two-sided error quantum algorithms presented in Section 3.2, and to Alex B. Grilo and Mario Szegedy for insightful comments. I thank Robin Kothari for pointing out the improved separation between certificate complexity and approximate degree in [5], which led to an improvement in from $(\log n)^{7/6}$ (using [1]) to $(\log n)^{2-o(1)}$ in Theorem 32.

1 Introduction

While quantum computers hold the promise of significant speedups for a number of problems, building them is a serious technological challenge, and it is expected that early quantum computers will have quantum memories of very limited size. This motivates the theoretical question: what problems could we solve faster on a quantum computer with limited space? Or similarly, what is the minimum number of qubits needed to solve a given problem (and hopefully still get a speedup).

We take a modest step towards answering such questions, by relating the space complexity of a function f to its *span program size*, which is a measure that has received significant attention in theoretical computer science over the past few decades. Span programs are a

model of computation introduced by Karchmer and Wigderson [10] in an entirely classical setting. They defined a span program for a Boolean function $f : \{0,1\}^n \to \{0,1\}$ as a matrix A with each of its columns labelled by an index $i \in [n]$ and a bit $b \in \{0,1\}$, and some fixed target vector in the columnspace of A. The span program *decides* f if for all x such that $f(x) = 1$, the target vector is in the span of the vectors labelled by (i, x_i) for $i \in [n]$. The *size* of the span program is the sum over i of the dimension of the span of the columns labelled by $(i, 0)$ or $(i, 1)$ (see also Definition 12). The span program size of f is then the minimum size of any span program deciding f, and was originally defined to lower bound the size of *counting branching programs*.

Several decades after the introduction of span programs, Reichardt and Špalek [18] related them to quantum algorithms, and introduced the new measure of *span program complexity* (see Definition 13). The importance of span programs in quantum algorithms stems from the ability to compile any span program for a function f into a bounded error quantum algorithm for f [17]. In particular, there is a tight correspondence between the span program *complexity* of f, and its quantum query complexity – a rather surprising and beautiful connection for a model originally introduced outside the realm of quantum computing. In contrast, the classical notion of span program *size* had received no attention in the quantum computing literature before now.

Ref. [8] defined the notion of an approximate span program for a function f. Loosely speaking, a span program *approximates* f if for every x such that $f(x) = 1$, the target is *close to* the span of the columns labelled by $\{(i, x_i)\}_{i\in[n]}$, and otherwise, the target is far from this span. They showed that even an approximate span program for f can be compiled into a bounded error quantum algorithm for f. In this work, we further relax the definition of an approximate span program for f, making analysis of such algorithms significantly easier (see Definition 15).

Let $\mathsf{S}_U(f)$ denote the *bounded error unitary space complexity of* f, or the minimum space needed by a unitary quantum algorithm[1] that computes f with bounded error (see Definition 7). For a function $f : \{0,1\}^n \to \{0,1\}$, we can assume that the input is accessed by queries, so that we do not need to store the full n-bit input in working memory, but we need at least $\log n$ bits of memory to store an index into the input. Thus, a lower bound of $\omega(\log n)$ on $\mathsf{S}_U(f)$ for some f would be non-trivial.

Letting $\mathsf{SP}(f)$ denote the minimum size of a span program deciding f, and $\widetilde{\mathsf{SP}}(f)$ the minimum size of a span program *approximating* f (see Definition 16), we have the following (see Theorem 24):

▶ **Theorem 1** (Informal). *For any Boolean function f, if $\mathsf{S}_U(f)$ denotes its bounded error unitary space complexity, and $\widetilde{\mathsf{SP}}(f)$ its approximate span program size, then*

$$\mathsf{S}_U(f) \geq \log \widetilde{\mathsf{SP}}(f).$$

Similarly, if $\mathsf{S}^1_U(f)$ denotes its one-sided error unitary space complexity, and $\mathsf{SP}(f)$ its span program size, then

$$\mathsf{S}^1_U(f) \geq \log \mathsf{SP}(f).$$

[1] A unitary quantum algorithm is a quantum algorithm in which all measurements are delayed until the end. In contrast to time complexity, the space complexity of an algorithm may be significantly smaller if we allow intermediate measurements. See [6] for a discussion of the distinction between unitary and non-unitary quantum space.

The relationship between span program size and unitary quantum space complexity is rather natural, as the span program size of f is known to lower bound the minimum size of a symmetric branching program for f, and the logarithm of the branching program size of a function f characterizes its classical deterministic space complexity.

The inequality $\mathsf{S}_U^1(f) \geq \log \mathsf{SP}(f)$, although not observed previously, follows straightforwardly from a construction of [17] for converting a one-sided error quantum algorithm for f into a span program for f – one need only observe that the size of the resulting span program is closely related to the space complexity of the algorithm. We adapt this construction to show how to convert a bounded (two-sided) error quantum algorithm for f with query complexity T and space complexity $S \geq \log T$ into an approximate span program for f with complexity $\Theta(T)$ and size $2^{\Theta(S)}$, proving $\mathsf{S}_U(f) \geq \Omega(\log \widetilde{\mathsf{SP}}(f))$. The connection between $\mathsf{S}_U(f)$ and $\log \widetilde{\mathsf{SP}}(f)$ is tight up to an additive term of the logarithm of the minimum complexity of any span program for f with optimal size. This follows from the fact that an approximate span program can be compiled into a quantum algorithm in a way that similarly preserves the correspondence between space complexity and (logarithm of) span program size, as well as the correspondence between query complexity and span program complexity (see Theorem 17). While the preservation of the correspondence between query complexity and span program complexity (in both directions) is not necessary for our results, it may be useful in future work for studying lower bounds on time and space simultaneously – somewhat analogous to branching programs, which capture both the time and space of classical algorithms.

The significance of Theorem 1 is that span program size has received extensive attention in theoretical computer science. Using results from [3], the connection in Theorem 1 immediately implies the following (Theorem 25):

▶ **Theorem 2.** *For almost all Boolean functions f on n bits, $\mathsf{S}_U^1(f) = \Omega(n)$.*

If we make a uniformity assumption that the quantum space complexity of an algorithm is at least the logarithm of its time complexity, then Theorem 2 would follow from a lower bound of $\Omega(2^n)$ on the quantum time complexity of almost all n-bit Boolean functions. Notwithstanding, the proof via span program size is evidence of the power of the technique.

In the pursuit of lower bounds on span program size of concrete functions, several nice expressions lower bounding $\mathsf{SP}(f)$ have been derived. By adapting one such lower bound on $\mathsf{SP}(f)$ to $\widetilde{\mathsf{SP}}(f)$, we get the following (see Lemma 29):

▶ **Theorem 3** (Informal). *For any Boolean function f, and* partial matrix $M \in (\mathbb{R} \cup \{\star\})^{f^{-1}(0)\times f^{-1}(1)}$ *with $\|M\|_\infty \leq 1$:*

$$\mathsf{S}_U(f) \geq \Omega\left(\log\left(\frac{\frac{1}{2}\text{-rank}(M)}{\max_{i\in[n]} \text{rank}(M \circ \Delta_i)}\right)\right),$$

where $\circ$ denotes the entrywise product, and $\Delta_i[x,y] = 1$ if $x_i \neq y_i$ and 0 else.

Above, $\frac{1}{2}$-rank denotes the approximate rank, or the minimum rank of any matrix $\widetilde{M}$ such that $|M[x,y] - \widetilde{M}[x,y]| \leq \frac{1}{2}$ for each x, y such that $M[x,y] \neq \star$. If we replace $\frac{1}{2}$-rank(M) with rank(M), we get the logarithm of an expression called the *rank measure*, introduced by Razborov [15]. The rank measure was shown by Gàl to be a lower bound on span program size, SP [7], and thus, our results imply that the log of the rank measure is a lower bound on S_U^1. It is straightforward to extend this proof to the approximate case to get Theorem 3.

Theorem 3 seems to give some hope of proving a non-trivial – that is, $\omega(\log n)$ – lower bound on the unitary space complexity of some explicit f, by exhibiting a matrix M for which the (approximate) rank measure is $2^{\omega(\log n)}$. In [15], Razborov showed that the rank

measure is a lower bound on the Boolean formula size of f, motivating significant attempts to prove lower bounds on the rank measure of explicit functions. The bad news is, circuit lower bounds have been described as "Complexity theory's Waterloo" [2]. Despite significant effort, no non-trivial lower bound on span program size for any f is known.

Due to the difficulty of proving explicit lower bounds on span program size, earlier work has considered the easier problem of lower bounding *monotone* span program size, $\mathsf{mSP}(f)$. A monotone span program is a span program where the columns of A are labelled by $(i, 1)$ for $i \in [n]$ (i.e. there are no columns associated with $(i, 0)$). For a monotone function f, the monotone span program size of f, $\mathsf{mSP}(f)$ is the minimum size of any *monotone span program* for f. We can similarly define the *approximate monotone span program size* of f, $\widetilde{\mathsf{mSP}}(f)$. Although $\log \widetilde{\mathsf{mSP}}(f)$ is *not* a lower bound on $\mathsf{S}_U(f)$, even for monotone f, it is a lower bound on the space complexity of any algorithm obtained by compiling a monotone span program. We show that such algorithms are equivalent to a more natural class of algorithms called monotone phase estimation algorithms. Informally, a phase estimation algorithm is an algorithm that works by performing phase estimation of some unitary that makes a single query to the input, and estimating the amplitude on a 0 in the phase register (see Definition 41). Phase estimation algorithms are completely general, in the sense that any unitary quantum algorithm can be transformed into a phase estimation in a way that asymptotically preserves its space and query complexity. A monotone phase estimation algorithm is a phase estimation algorithm where, loosely speaking, adding 0s to the input can only make the algorithm more likely to reject (see Definition 42). We can then prove the following (see Theorem 43):

▶ **Theorem 4** (Informal). *For any Boolean function f, any bounded error monotone phase estimation algorithm for f has space complexity at least $\log \widetilde{\mathsf{mSP}}(f)$, and any one-sided error monotone phase estimation algorithm for f has space complexity at least $\log \mathsf{mSP}(f)$.*

Fortunately, non-trivial lower bounds for the monotone span program complexity are known for explicit functions. In Ref. [3], Babai, Gàl and Wigderson showed a lower bound of $\mathsf{mSP}(f) \geq 2^{\Omega\left(\log^2(n)/\log\log(n)\right)}$ for some explicit function f, which was later improved to $\mathsf{mSP}(f) \geq 2^{\Omega(\log^2(n))}$ by Gàl [7]. In Ref. [19], a function f was exhibited with $\mathsf{mSP}(f) \geq 2^{n^\epsilon}$ for some constant $\epsilon \in (0, 1)$, and in the strongest known result, Pitassi and Robere exhibited a function f with $\mathsf{mSP}(f) \geq 2^{\Omega(n)}$ [14]. Combined with our results, each of these implies a lower bound on the space complexity of one-sided error monotone phase estimation algorithms. For example, the result of [14] implies a lower bound of $\Omega(n)$ on the space complexity of one-sided error monotone phase estimation algorithms for a certain satisfiability problem f. This lower bound, and also the one in [19], are proven by choosing f based on a constraint satisfaction problem with high *refutation width*, which is a measure related to the space complexity of certain classes of SAT solvers, so it is intuitively not surprising that these problems should require a large amount of space to solve with one-sided error.

For the case of bounded error space complexity, we also prove the following (see Theorem 32, Corollary 44):

▶ **Theorem 5** (Informal). *There exists a function $f : \{0, 1\}^n \to \{0, 1\}$ such that any bounded error monotone phase estimation algorithm for f has space complexity $(\log n)^{2-o(1)}$.*

This lower bound is non-trivial, although much less so than the best known lower bound of $\Omega(n)$ for the one-sided case. Our result also implies a new lower bound of $2^{(\log n)^{2-o(1)}}$ on the monotone span program complexity of the function f in Theorem 5.

To prove the lower bound in Theorem 5, we apply a new technique that leverages the best possible gap between the certificate complexity and approximate polynomial degree of a function, employing a function $g : \{0,1\}^{m^{2+o(1)}} \to \{0,1\}$ from [5][2], whose certificate complexity is $m^{1+o(1)}$, and whose approximate degree is $m^{2-o(1)}$. Following a strategy of [19], we use this g to construct a *pattern matrix* [20] (see Definition 37) and use this matrix in a monotone version of Theorem 3 (see Theorem 33). The fact that certificate complexity and approximate degree of total functions are related by $\widetilde{\deg}_{1/3}(g) \leq C(g)^2$ for all g is a barrier to proving a lower bound better than $(\log n)^2$ using this technique, but we also give a generalization that has the potential to prove significantly better lower bounds (see Lemma 40).

Discussion and open problems

The most conspicuous open problem of this work is to prove a lower bound of $\omega(\log n)$ on $\mathsf{S}_U(f)$ or even $\mathsf{S}^1_U(f)$ for some explicit decision function f. It is known that any space S quantum Turing machine can be simulated by a deterministic classical algorithm in space S^2 [21] so a lower bound of $\omega(\log^2 n)$ on classical space complexity would also give a non-trivial lower bound on quantum space complexity. If anything, the relationship to span program size is evidence that this task is extremely difficult.

We have shown a lower bound of $2^{(\log n)^{2-o(1)}}$ on the approximate monotone span program complexity of an explicit monotone function f, which gives a lower bound of $(\log n)^{2-o(1)}$ on the bounded error space complexity needed by a quantum algorithm of a very specific form: a monotone phase estimation algorithm. This is much worse than the best bound we can get in the one-sided case: a lower bound of $\Omega(n)$ for some explicit function. An obvious open problem is to try to get a better lower bound on the approximate monotone span program complexity of some explicit function.

Our lower bound of $(\log n)^{2-o(1)}$ only applies to the space complexity of monotone phase estimation algorithms and does not preclude the existence of a more space-efficient algorithm for f of a different form. We do know that phase estimation algorithms are fully general, in the sense that every problem has a space-optimal phase estimation algorithm. Does something similar hold for monotone phase estimation algorithms? This would imply that $\log \mathsf{m}\widetilde{\mathsf{SP}}(f)$ is a lower bound on $\mathsf{S}_U(f)$ for all monotone functions f.

In this work, we define an approximate version of the rank method, and monotone rank method, and in case of the monotone rank method, give an explicit non-trivial lower bound. The rank method is known to give lower bounds on formula size, and the monotone rank method on monotone formula size. An interesting question is whether the approximate rank method also gives lower bounds on some complexity theoretic quantity related to formulas.

Our results are a modest first step towards understanding unitary quantum space complexity, but even if we could lower bound the unitary quantum space complexity of an explicit function, there are several obstacles limiting the practical consequences of such a result. First, while an early quantum computer will have a small *quantum* memory, it is simple to augment it with a much larger classical memory. Thus, in order to achieve results with practical implications, we would need to study computational models that make a distinction between quantum and classical memories. We leave this as an important challenge for future work.

[2] An earlier version of this work used a function described in [1] with a 7/6-separation between certificate complexity and approximate degree. We thank Robin Kothari for pointing us to the improved result of [5].

Second, we are generally only interested in running quantum algorithms when we get an advantage over classical computers in the time complexity, so results that give a lower bound on the quantum space required if we wish to keep the time complexity small, such as time-space lower bounds, are especially interesting. While we do not address time-space lower bounds in this paper, one advantage of the proposed quantum space lower bound technique, via span programs, is that span programs are also known to characterize quantum query complexity, which is a lower bound on time complexity. We leave exploration of this connection for future work.

We mention two previous characterizations of $\mathsf{S}_U(f)$. Ref. [9] showed that $\mathsf{S}_U(f)$ is equal to the logarithm of the minimum width of a *matchgate circuit* computing f, and thus our results imply that this minimum matchgate width is approximately equal to the approximate span program size of f. Separately, in Ref. [6], Fefferman and Lin showed that for every function k, inverting $2^{k(n)} \times 2^{k(n)}$ matrices is complete for the class of problems f such that $\mathsf{S}_U(f) \leq k(n)$. Our results imply that evaluating an approximate span program of size $2^{k(n)}$ (for some suitable definition of the problem) is similarly complete for this class. Evaluating an approximate span program boils down to deciding if $\|A(x)^+|w_0\rangle\|$ is below a certain threshold, where $A(x)$ is the span program matrix A restricted to the rows labeled by $\{(i, x_i)\}_{i\in[n]}$, and $|w_0\rangle$ is some input-independent initial state; so these results are not unrelated[3]. We leave exploring these connections as future work.

Organization

The remainder of this paper is organized as follows. In Section 2, we present the necessary notation and quantum algorithmic preliminaries, and define quantum space complexity. In Section 3, we define span programs, and describe how they correspond to quantum algorithms. In particular, we describe how a span program can be "compiled" into a quantum algorithm, and in Section 3.2, show how a quantum algorithm can be turned into a span program, with both transformations moreorless preserving the relationships between span program size and algorithmic space, and between span program complexity and query complexity. From this correspondence, we obtain, in Section 4, expressions that lower bound the quantum space complexity of a function. While we do not know how to instantiate any of these expressions to get a non-trivial lower bound for a concrete function, in Section 5, we consider to what extent monotone span program lower bounds are meaningful lower bounds on quantum space complexity, and give the first non-trivial lower bound on the approximate monotone span program size of a function.

2 Preliminaries

We begin with some miscellaneous notation. For a vector $|v\rangle$, we let $\||v\rangle\|$ denote its ℓ_2-norm. In the following, let A be a matrix with i and j indexing its rows and columns. Define:

$$\|A\|_\infty = \max_{i,j} |A_{i,j}|, \quad \text{and} \quad \|A\| = \max\{\|A|v\rangle\| : \||v\rangle\| = 1\}.$$

[3] In the notation of Definition 12, $A(x) = A\Pi_{H(x)}$, and $|w_0\rangle = A^+|\tau\rangle$ for $|\tau\rangle$ the target. Then one can verify that the *positive witness size* of x is $w_+(x) = \left\|A(x)^+|w_0\rangle\right\|^2$ (see Definition 13).

Define the ε-rank of a matrix A as the minimum rank of any matrix B such that $\|A-B\|_\infty \leq \varepsilon$. For a matrix A with singular value decomposition $A = \sum_k \sigma_k |v_k\rangle\langle u_k|$, define:

$$\mathrm{col}(A) = \mathrm{span}\{|v_k\rangle\}_k,\ \ \mathrm{row}(A) = \mathrm{span}\{|u_k\rangle\}_k,\ \ \ker(A) = \mathrm{row}(A)^\perp,\ \ A^+ = \sum_k \frac{1}{\sigma_k}|u_k\rangle\langle v_k|.$$

The following lemma, from [12], is useful in the analysis of quantum algorithms.

▶ **Lemma 6** (Effective spectral gap lemma). *Fix orthogonal projectors Π_A and Π_B. Let $U = (2\Pi_A - I)(2\Pi_B - I)$, and let Π_Θ be the orthogonal projector onto the $e^{i\theta}$-eigenspaces of U such that $|\theta| \leq \Theta$. Then if $\Pi_A|u\rangle = 0$, $\|\Pi_\Theta \Pi_B|u\rangle\| \leq \frac{\Theta}{2}\||u\rangle\|$.*

In general, we will let Π_V denote the orthogonal projector onto V, for a subspace V.

Unitary quantum algorithms and space complexity

A *unitary quantum algorithm* $\mathcal{A} = \{\mathcal{A}_n\}_{n\in\mathbb{N}}$ is a family (parametrized by n) of sequences of $2^{s(n)}$-dimensional unitaries $U_1^{(n)}, \ldots, U_{T(n)}^{(n)}$, for some $s(n) \geq \log n$ and $T(n)$. (We will generally dispense with the explicit parametrization by n). For $x \in \{0,1\}^n$, let $\mathcal{O}_x$ be the unitary that acts as $\mathcal{O}_x|j\rangle = (-1)^{x_j}|j\rangle$ for $j \in [n]$, and $\mathcal{O}_x|0\rangle = |0\rangle$. We let $\mathcal{A}(x)$ denote the random variable obtained from measuring

$$U_T \mathcal{O}_x U_{T-1} \ldots \mathcal{O}_x U_1 |0\rangle$$

with some two-outcome measurement that should be clear from context. We call $T(n)$ the *query complexity* of the algorithm, and $S(n) = s(n) + \log T(n)$ the *space complexity*. By including a $\log T(n)$ term in the space complexity, we are implicitly assuming that the algorithm must maintain a counter to know which unitary to apply next. This is a fairly mild uniformity assumption (that is, any uniformly generated algorithm uses $\Omega(\log T)$ space), and it will make the statement of our results much simpler. The requirement that $s(n) \geq \log n$ is to ensure that the algorithm has enough space to store an index $i \in [n]$ into the input.

For a (partial) function $f : D \to \{0,1\}$ for $D \subseteq \{0,1\}^n$, we say that $\mathcal{A}$ computes f with bounded error if for all $x \in D$, $\mathcal{A}(x) = f(x)$ with probability at least $2/3$. We say that $\mathcal{A}$ computes f with one-sided error if in addition, for all x such that $f(x) = 1$, $\mathcal{A}(x) = f(x)$ with probability 1.

▶ **Definition 7** (Unitary Quantum Space). *For a family of functions $f : D \to \{0,1\}$ for $D \subseteq \{0,1\}^n$, the unitary space complexity of f, $\mathsf{S}_U(f)$, is the minimum $S(n)$ such that there is a family of unitary quantum algorithms with space complexity $S(n)$ that computes f with bounded error. Similarly, $\mathsf{S}_U^1(f)$ is the minimum $S(n)$ such that there is a family of unitary quantum algorithms with space complexity $S(n)$ that computes f with one-sided error.*

▶ Remark 8. Since T is the number of queries made by the algorithm, we may be tempted to assume that it is at most n, however, while every n-bit function can be computed in n queries, this may not be the case when space is restricted. For example, it is difficult to imagine an algorithm that uses $O(\log n)$ space and $o(n^{3/2})$ quantum queries to solve the following problem on $[q]^n \equiv \{0,1\}^{n\log q}$: Decide whether there exist distinct $i,j,k \in [n]$ such that $x_i + x_j + x_k = 0 \mod q$.

Phase estimation

For a unitary U acting on H and a state $|\psi\rangle \in H$, we will say we perform T *steps of phase estimation of* U *on* $|\psi\rangle$ when we compute:

$$\frac{1}{\sqrt{T}} \sum_{t=0}^{T-1} |t\rangle U^t |\psi\rangle,$$

and then perform a quantum Fourier transform over $\mathbb{Z}/T\mathbb{Z}$ on the first register, called the *phase register*. This procedure was introduced in [11]. It is easy to see that the complexity (either query or time) of phase estimation is $O(T)$ times the complexity of implementing a controlled call to U. The space complexity of phase estimation is $\log T + \log \dim(H)$. We will use the following properties:

▶ **Lemma 9** (Phase Estimation). *If $U|\psi\rangle = |\psi\rangle$, then performing T steps of phase estimation of U on $|\psi\rangle$ and measuring the phase register results in outcome 0 with probability 1. If $U|\psi\rangle = e^{i\theta}|\psi\rangle$ for $|\theta| \in (\pi/T, \pi]$, then performing T steps of phase estimation of U on $|\psi\rangle$ results in outcome 0 with probability at most $\frac{\pi}{T\theta}$.*

We note that we can increase the success probability to any constant by adding some constant number k of phase registers, and doing phase estimation k times in parallel, still using a single register for U, and taking the majority. This still has space complexity $\log \dim H + O(\log T)$.

Amplitude estimation

For a unitary U acting on H, a state $|\psi_0\rangle \in H$, and an orthogonal projector Π on H, we will say we perform M *steps of amplitude estimation of* U *on* $|\psi\rangle$ *with respect to* Π when we perform M steps of phase estimation of

$$U(2|\psi\rangle\langle\psi| - I)U^\dagger(2\Pi - I)$$

on $U|\psi\rangle$, then, if the phase register contains some $t \in \{0, \ldots, M-1\}$, compute $\tilde{p} = \sin^2 \frac{\pi t}{2M}$, which is an estimate of $\|\Pi U|\psi\rangle\|^2$ in a new register. The (time or query) complexity of this is $O(M)$ times the complexity of implementing a controlled call to U, implementing a controlled call to $2\Pi - I$, and generating $|\psi\rangle$. The space complexity is $\log T + \log \dim H + O(1)$. We have the following guarantee [4]:

▶ **Lemma 10.** *Let $p = \|\Pi U|\psi\rangle\|^2$. There exists $\Delta = \Theta(1/M)$ such that when $\tilde{p}$ is obtained as above from M steps of amplitude estimation, with probability at least $1/2$, $|\tilde{p} - p| \leq \Delta$.*

We will thus also refer to M steps of amplitude estimation as *amplitude estimation to precision* $1/M$.

3 Span Programs and Quantum Algorithms

In Section 3.1, we will define a span program, its size and complexity, and what it means for a span program to approximate a function f. In Section 3.2, we prove the following theorem, which implies Theorem 1:

▶ **Theorem 11.** *Let $f : D \to \{0,1\}$ for $D \subseteq \{0,1\}^n$ and let $\mathcal{A}$ be a unitary quantum algorithm using T queries, and space S to compute f with bounded error. Then for any constant $\kappa \in (0,1)$, there is a span program $P_\mathcal{A}$ with size $s(P_\mathcal{A}) \leq 2^{O(S)}$ that κ-approximates f with complexity $C_\kappa \leq O(T)$. If $\mathcal{A}$ decides f with one-sided error, then $P_\mathcal{A}$ decides f.*

3.1 Span Programs

Span programs were first introduced in the context of classical complexity theory in [10], where they were used to study counting classes for nondeterministic logspace machines. While span programs can be defined with respect to any field, we will consider span programs over $\mathbb{R}$ (or equivalently, $\mathbb{C}$, when convenient, see Remark 20). We use the following definition, slightly modified from [10]:

▶ **Definition 12** (Span Program and Size). *A* span program on $\{0,1\}^n$ *consists of:*

- *Finite inner product spaces* $\{H_{j,b}\}_{j\in[n],b\in\{0,1\}} \cup \{H_{\text{true}}, H_{\text{false}}\}$. *We then define* $H = \bigoplus_{j,b} H_{j,b} \oplus H_{\text{true}} \oplus H_{\text{false}}$, *and for every* $x \in \{0,1\}^n$, $H(x) = H_{1,x_1} \oplus \cdots \oplus H_{n,x_n} \oplus H_{\text{true}}$.[4]
- *A vector space* V.
- *A* target vector $|\tau\rangle \in V$.
- *A linear map* $A : H \to V$.

We specify this span program by $P = (H, V, |\tau\rangle, A)$, *and leave the decomposition of* H *implicit. The* size *of the span program is* $s(P) = \dim H$.

To recover the classical definition from [10], we can view $A = \sum_{j,b} A\Pi_{H_{j,b}}$ as a matrix, with each of the columns of $A\Pi_{H_{j,b}}$ labeled by (j,b).

Span programs were introduced to the study of quantum query complexity in [18]. In the context of quantum query complexity, $s(P)$ is no longer the relevant measure of the complexity of a span program. Instead, [18] introduce the following measures:

▶ **Definition 13** (Span Program Complexity and Witnesses). *For* $P = (H, V, |\tau\rangle, A)$ *a span program on* $\{0,1\}^n$ *and input* $x \in \{0,1\}^n$, *we say* x *is* accepted *by the span program if there exists* $|w\rangle \in H(x)$ *such that* $A|w\rangle = |\tau\rangle$, *and otherwise we say* x *is* rejected *by the span program. Let* P_0 *and* P_1 *be respectively the set of rejected and accepted inputs to* P. *For* $x \in P_1$, *define the* positive witness complexity of x *as:*

$$w_+(x,P) = w_+(x) = \min\{\||w\rangle\|^2 : |w\rangle \in H(x), A|w\rangle = |\tau\rangle\}.$$

Such a $|w\rangle$ *is called a* positive witness for x. *For a domain* $D \subseteq \{0,1\}^n$, *we define the* positive complexity of P *(with respect to* D*) as:*

$$W_+(P,D) = W_+ = \max_{x\in P_1\cap D} w_+(x,P).$$

For $x \in P_0$, *define the* negative witness complexity of x *as:*

$$w_-(x,P) = w_-(x) = \min\{\|\langle\omega|A\|^2 : \langle\omega| \in \mathcal{L}(V,\mathbb{R}), \langle\omega|\tau\rangle = 1, \langle\omega|A\Pi_{H(x)} = 0\}.$$

Above, $\mathcal{L}(V,\mathbb{R})$ *denotes the set of linear functions from* V *to* $\mathbb{R}$. *Such an* $\langle\omega|$ *is called a* negative witness for x. *We define the* negative complexity of P *(with respect to* D*) as:*

$$W_-(P,D) = W_- = \max_{x\in P_0\cap D} w_-(x,P).$$

Finally, we define the complexity of P *(with respect to* D*) by* $C(P,D) = \sqrt{W_+W_-}$.

For $f : D \to \{0,1\}$, we say a span program P *decides* f if $f^{-1}(0) \subseteq P_0$ and $f^{-1}(1) \subseteq P_1$.

[4] We remark that while H_{true} and H_{false} may be convenient in constructing a span program, they are not necessary. We can always consider a partial function f' defined on $(n+1)$-bit strings of the form $(x,1)$ for x in the domain of f, as $f(x)$, and let $H_{n+1,1} = H_{\text{true}}$ and $H_{n+1,0} = H_{\text{false}}$.

▶ **Definition 14.** *We define the* span program size *of a function f, denoted* $\mathsf{SP}(f)$*, as the minimum $s(P)$ over families of span programs that decide f.*

We note that originally, in [10], span program size was defined

$$s'(P) = \sum_{j,b} \dim(\mathrm{col}(A\Pi_{H_{j,b}})) = \sum_{j,b} \dim(\mathrm{row}(A\Pi_{H_{j,b}})).$$

This could differ from $s(P) = \dim(H) = \sum_{j,b} \dim(H_{j,b})$, because $\dim(H_{j,b})$ might be much larger than $\dim(\mathrm{row}(A\Pi_{H_{j,b}}))$. However, if $\dim(H_{j,b}) > \dim(\mathrm{row}(A\Pi_{H_{j,b}}))$ for some j, b, then it is a simple exercise to show that the dimension of $\dim(H_{j,b})$ can be reduced without altering the witness size of any $x \in \{0,1\}^n$, so the definition of $\mathsf{SP}(f)$ is the same as if we'd used $s'(P)$ instead of $s(P)$. In any case, we will not be relying on previous results about the span program size as a black-box, and will rather prove all required statements, so this difference has no impact on our results.

While span program size has only previously been relevant outside the realm of quantum algorithms, the complexity of a span program deciding f has a fundamental correspondence with the quantum query complexity of f. Specifically, a span program P can be turned into a quantum algorithm for f with query complexity $C(P, D)$, and moreover, for every f, there exists a span program such that the algorithm constructed in this way is optimal [17]. This second direction is not constructive: there is no known method for converting a quantum algorithm with query complexity T to a span program with complexity $C(P, D) = \Theta(T)$. However, if we relax the definition of which functions are decided by a span program, then this situation can be improved. The following is a slight relaxation of [8, Definition 2.6][5].

▶ **Definition 15** (A Span Program that Approximately Decides a Function)**.** *Let $f : D \to \{0,1\}$ for $D \subseteq \{0,1\}^n$ and $\kappa \in (0,1)$. We say that a span program P on $\{0,1\}^n$* κ-approximates *f if $f^{-1}(0) \subseteq P_0$, and for every $x \in f^{-1}(1)$, there exists an* approximate positive witness *$|\hat{w}\rangle$ such that $A|\hat{w}\rangle = |\tau\rangle$, and $\left\|\Pi_{H(x)^\perp}|\hat{w}\rangle\right\|^2 \leq \frac{\kappa}{W_-}$. We define the approximate positive complexity as*

$$\widehat{W}_+ = \widehat{W}_+^\kappa(P, D) = \max_{x \in f^{-1}(1)} \min \left\{ \||\hat{w}\rangle\|^2 : A|\hat{w}\rangle = |\tau\rangle, \left\|\Pi_{H(x)^\perp}|\hat{w}\rangle\right\|^2 \leq \frac{\kappa}{W_-} \right\}.$$

If P κ-approximates f, we define the complexity of P (wrt. D and κ) as $C_\kappa(P, D) = \sqrt{\widehat{W}_+ W_-}$.

If $\kappa = 0$, the span program in Definition 15 *decides* f (exactly), and $\widehat{W}_+ = W_+$. By [8], for any x,

$$\min \left\{ \left\|\Pi_{H(x)^\perp}|\hat{w}\rangle\right\|^2 : A|\hat{w}\rangle = |\tau\rangle \right\} = \frac{1}{w_-(x)}.$$

Thus, since $W_- = \max_{x \in f^{-1}(0)} w_-(x)$, for every $x \in f^{-1}(0)$, there does not exist an approximate positive witness with $\left\|\Pi_{H(x)^\perp}|\hat{w}\rangle\right\|^2 < \frac{1}{W_-}$. Thus, when a span program κ-approximates f, there is a gap of size $\frac{1-\kappa}{W_-}$ between the smallest positive witness error $\left\|\Pi_{H(x)^\perp}|\hat{w}\rangle\right\|^2$ of $x \in f^{-1}(1)$, the smallest positive witness error of $x \in f^{-1}(0)$.

[5] Which was already a relaxation of the notion of a span program deciding a function.

▶ **Definition 16.** *We define the* κ-approximate span program size *of a function* f*, denoted* $\widetilde{\mathsf{SP}}_\kappa(f)$*, as the minimum* $s(P)$ *over families of span programs that* κ*-approximate* f*. We let* $\widetilde{\mathsf{SP}}(f) = \widetilde{\mathsf{SP}}_{1/4}(f)$.

Then we have the following theorem, whose proof is nearly identical to that of [8, Lemma 3.6]. The only difference between [8, Lemma 3.6] and Theorem 17 below is that here we let an approximate positive witness for x be any witness with error $\left\|\Pi_{H(x)^\perp}|w\rangle\right\|^2$ at most κ/W_-, whereas in [8], an approximate positive witness must have error as small as possible. This relaxation has negligible effect on the proof.

▶ **Theorem 17.** *Let* $f : D \to \{0,1\}$ *for* $D \subseteq \{0,1\}^n$*, and let* P *be a span program that* κ*-approximates* f *with size* K *and complexity* C*, for some constant* $\kappa \in (0,1)$*. Then there exists a unitary quantum algorithm* $\mathcal{A}_P$ *that decides* f *with bounded error in space* $S = O(\log K + \log C)$ *using* $T = O(C)$ *queries to* x*.*

We note that the choice of $\kappa = 1/4$ in $\widetilde{\mathsf{SP}}(f)$ is arbitrary, as it is possible to modify a span program to reduce any constant κ to any other constant without changing the size or complexity asymptotically. This convenient observation is formalized in the following claim.

▷ Claim 18. Let P be a span program that κ-approximates $f : D \to \{0,1\}$ for some constant κ. For any constant $\kappa' \leq \kappa$, there exists a span program P' that κ'-approximates f with $s(P') = (s(P)+2)^{2^{\frac{\log \frac{1}{\kappa'}}{\log \frac{1}{\kappa}}}}$, and $C_{\kappa'}(P', D) \leq O\left(C_\kappa(P, D)\right)$.

We prove Claim 18 in Appendix A. We have the following corollary that will be useful later, where $\mathsf{m}\widetilde{\mathsf{SP}}_\kappa$ is the *monotone approximate span program size*, defined in Definition 30:

▶ **Corollary 19.** *For any* $\kappa, \kappa' \in (0,1)$ *with* $\kappa' < \kappa$*, and any Boolean function* f*,*

$$\widetilde{\mathsf{SP}}_\kappa(f) \geq \widetilde{\mathsf{SP}}_{\kappa'}(f)^{\frac{1}{2}\frac{\log \frac{1}{\kappa}}{\log \frac{1}{\kappa'}}} - 2.$$

If f *is monotone, we also have*

$$\mathsf{m}\widetilde{\mathsf{SP}}_\kappa(f) \geq \mathsf{m}\widetilde{\mathsf{SP}}_{\kappa'}(f)^{\frac{1}{2}\frac{\log \frac{1}{\kappa}}{\log \frac{1}{\kappa'}}} - 2.$$

Proof. Let P κ-approximate f with optimal size, so $s(P) = \widetilde{\mathsf{SP}}_\kappa(f)$. Then by Claim 18, there is a span program P' that κ'-approximates f with size

$$\widetilde{\mathsf{SP}}_{\kappa'}(f) \leq s(P') = \left(\widetilde{\mathsf{SP}}_\kappa(f) + 2\right)^{2^{\frac{\log \frac{1}{\kappa'}}{\log \frac{1}{\kappa}}}}.$$

The first result follows. The second is similar, but also includes the observation that if P is monotone, so is P'. ◀

▶ Remark 20. It can sometimes be useful to construct a span program over $\mathbb{C}$. However, for any span program over $\mathbb{C}$, P, there is a span program over $\mathbb{R}$, P', such that for all $x \in P_0$, $w_-(x, P') \leq w_-(x, P)$, for all $x \in P_1$, $w_+(x, P') \leq w_+(x, P)$, and $s(P') \leq 2s(P)$. Thus, we will restrict our attention to real span programs, but still allow constructions of span programs over $\mathbb{C}$ (in particular, in Section 3.2 and Section 5.2.1).

3.2 From Quantum Algorithms to Span Programs

In this section, we will show how to turn a unitary quantum algorithm into a span program, proving Theorem 11, which implies Theorem 1. The construction we use to prove Theorem 11 is based on a construction of Reichardt for turning any one-sided error quantum algorithm into a span program whose complexity matches the algorithm's query complexity [17, arXiv version]. We observe that the logarithm of the span program's size is closely related to the algorithm's space complexity. We also show that a similar construction works for two-sided error algorithms, but the resulting span program only approximately decides f.

The algorithm

Fix a function $f : D \to \{0,1\}$ for $D \subseteq \{0,1\}^n$, and a unitary quantum algorithm $\mathcal{A}$ such that on input $x \in f^{-1}(0)$, $\Pr[\mathcal{A}(x) = 1] \leq \frac{1}{3}$, and on input $x \in f^{-1}(1)$, $\Pr[\mathcal{A}(x) = 1] \geq 1 - \varepsilon$, for $\varepsilon \in \{0, \frac{1}{3}\}$, depending on whether we want to consider a one-sided error or a bounded error algorithm. Let $p_0(x) = \Pr[\mathcal{A}(x) = 0]$, so if $f(x) = 0$, $p_0(x) \geq 2/3$, and if $f(x) = 1$, $p_0(x) \leq \varepsilon$.

We can suppose $\mathcal{A}$ acts on three registers: a query register $\mathrm{span}\{|j\rangle : j \in [n] \cup \{0\}\}$; a workspace register $\mathrm{span}\{|z\rangle : z \in \mathcal{Z}\}$ for some finite set of symbols $\mathcal{Z}$ that contains 0; and an answer register $\mathrm{span}\{|a\rangle : a \in \{0,1\}\}$. The query operator $\mathcal{O}_x$ acts on the query register as $\mathcal{O}_x|j\rangle = (-1)^{x_j}|j\rangle$ if $j \geq 1$, and $\mathcal{O}_x|0\rangle = |0\rangle$. If $\mathcal{A}$ makes T queries, the final state of $\mathcal{A}$ is:

$$|\Psi_{2T+1}(x)\rangle = U_{2T+1}\mathcal{O}_x U_{2T-1} \dots U_3 \mathcal{O}_x U_1 |0,0,0\rangle$$

for some unitaries $U_{2T+1}, \dots, U_1$. The output bit of the algorithm, $\mathcal{A}(x)$, is obtained by measuring the answer register of $|\Psi_{2T+1}(x)\rangle$. We have given the input-independent unitaries odd indicies so that we may refer to the t-th query as U_{2t}.

Let $|\Psi_0(x)\rangle = |\Psi_0\rangle = |0,0,0\rangle$ denote the starting state, and for $t \in \{1, \dots, 2T+1\}$, let $|\Psi_t(x)\rangle = U_t \dots U_1 |\Psi_0\rangle$ denote the state after t steps.

The span program

We now define a span program $P_{\mathcal{A}}$ from $\mathcal{A}$. The space H will represent all three registers of the algorithm, with an additional time counter register, and an additional register to represent a query value b.

$$H = \mathrm{span}\{|t,b,j,z,a\rangle : t \in \{0,\dots,2T+1\}, b \in \{0,1\}, j \in [n] \cup \{0\}, z \in \mathcal{Z}, a \in \{0,1\}\}.$$

We define V and A as follows, where c is some constant to be chosen later:

$$V = \mathrm{span}\{|t,j,z,a\rangle : t \in \{0,\dots,2T+1\}, j \in [n] \cup \{0\}, z \in \mathcal{Z}, a \in \{0,1\}\}$$

$$A|t,b,j,z,a\rangle = \begin{cases} |t,j,z,a\rangle - |t+1\rangle U_{t+1}|j,z,a\rangle & \text{if } t \in \{0,\dots,2T\} \text{ is even} \\ |t,j,z,a\rangle - (-1)^b|t+1,j,z,a\rangle & \text{if } t \in \{0,\dots,2T\} \text{ is odd} \\ |t,j,z,a\rangle & \text{if } t = 2T+1,\ a = 1, \text{ and } b = 0 \\ \sqrt{cT}|t,j,z,a\rangle & \text{if } t = 2T+1,\ a = 0, \text{ and } b = 0 \\ 0 & \text{if } t = 2T+1 \text{ and } b = 1. \end{cases}$$

For $t \leq 2T$, $A|t,b,j,z,a\rangle$ should be intuitively understood as applying U_{t+1} to $|j,z,a\rangle$, and incrementing the counter register from $|t\rangle$ to $|t+1\rangle$. When t is even, this correspondence is clear (in that case, the value of b is ignored). When t is odd, so $U_{t+1} = \mathcal{O}_x$, then as long as $b = x_j$, $(-1)^b|t+1,j,z,a\rangle = |t+1\rangle U_{t+1}|j,z,a\rangle$. We thus define

$$H_{j,b} = \mathrm{span}\{|t,b,j,z,a\rangle : t \in \{0,\dots,2T\} \text{ is odd}, z \in \mathcal{Z}, a \in \{0,1\}\}.$$

For even t, applying U_{t+1} is independent of the input, so we make the corresponding states available to every input; along with states where the query register is set to $j = 0$, meaning $\mathcal{O}_x$ acts input-independently; and accepting states, whose answer register is set to 1 at time $2T + 1$:

$$\begin{aligned} H_{\text{true}} = \text{span}\{|t, b, j, z, a\rangle : t \in \{0, \ldots, 2T\} \text{ is even}, b \in \{0,1\}, j \in [n], z \in \mathcal{Z}, a \in \{0,1\}\} \\ \oplus \text{span}\{|t, b, 0, z, a\rangle : t \in \{0, \ldots, 2T\}, b \in \{0,1\}, z \in \mathcal{Z}, a \in \{0,1\}\} \\ \oplus \text{span}\{|2T+1, b, j, z, 1\rangle : b \in \{0,1\}, j \in [n] \cup \{0\}, z \in \mathcal{Z}\}. \end{aligned}$$

The remaining part of H will be assigned to H_{false}:

$$H_{\text{false}} = \text{span}\{|2T+1, b, j, z, 0\rangle : b \in \{0,1\}, j \in [n] \cup \{0\}, z \in \mathcal{Z}\}.$$

Note that in defining A, we have put a large factor of $\sqrt{cT}$ in front of $A|2T+1, 0, j, z, 0\rangle$, making the vectors in H_{false} very "cheap" to use. These vectors are never in $H(x)$, but will be used as the error part of approximate positive witnesses, and the $\sqrt{cT}$ ensures they contribute relatively small error.

Finally, we define:

$$|\tau\rangle = |0, 0, 0, 0\rangle = |0\rangle|\Psi_0\rangle.$$

Intuitively, we can construct $|\tau\rangle$, the initial state, using a final state that has 1 in the answer register, and using the transitions $|t, j, z, a\rangle - |t+1\rangle U_{t+1}|j, z, a\rangle$ to move from the final state to the initial state. In the following analysis, we make this idea precise.

Analysis of $P_{\mathcal{A}}$

We will first show that for every x there is an approximate positive witness with error depending on its probability of being rejected by $\mathcal{A}$, $p_0(x)$.

▶ **Lemma 21.** *For any $x \in \{0,1\}^n$, there exists an approximate positive witness $|w\rangle$ for x in $P_{\mathcal{A}}$ such that:*

$$\||w\rangle\|^2 \leq 2T + 2, \text{ and } \left\|\Pi_{H(x)^\perp}|w\rangle\right\|^2 \leq \frac{p_0(x)}{cT}.$$

In particular, if $f(x) = 1$,

$$\left\|\Pi_{H(x)^\perp}|w\rangle\right\|^2 \leq \frac{\varepsilon}{cT}.$$

Proof. Let Q_x be the linear isometry that acts as

$$Q_x|j, z, a\rangle = |x_j, j, z, a\rangle \qquad \forall j \in [n] \cup \{0\}, z \in \mathcal{Z}, a \in \{0,1\},$$

where we interpret x_0 as 0. Note that for all $|j, z, a\rangle$, and $t \in \{0, \ldots, 2T\}$, we have

$$A(|t\rangle Q_x|j, z, a\rangle) = |t, j, z, a\rangle - |t+1\rangle U_{t+1}|j, z, a\rangle.$$

Let $\Pi_a = \sum_{j \in [n] \cup \{0\}, z \in \mathcal{Z}} |j, z, a\rangle\langle j, z, a|$ be the orthogonal projector onto states of the algorithm with answer register set to a. We will construct a positive witness for x from the states of the algorithm on input x, as follows:

$$|w\rangle = \sum_{t=0}^{2T} |t\rangle Q_x |\Psi_t(x)\rangle + |2T+1\rangle|0\rangle\Pi_1|\Psi_{2T+1}(x)\rangle + \frac{1}{\sqrt{cT}}|2T+1\rangle|0\rangle\Pi_0|\Psi_{2T+1}(x)\rangle.$$

To see that this is a positive witness, we compute $A|w\rangle$, using the fact that $U_{t+1}|\Psi_t(x)\rangle = |\Psi_{t+1}(x)\rangle$:

$$\begin{aligned}
A|w\rangle &= \sum_{t=0}^{2T} \left(|t\rangle|\Psi_t(x)\rangle - |t+1\rangle U_{t+1}|\Psi_t(x)\rangle\right) \\
&\qquad + |2T+1\rangle\Pi_1|\Psi_{2T+1}(x)\rangle + |2T+1\rangle\Pi_0|\Psi_{2T+1}(x)\rangle \\
&= \sum_{t=0}^{2T} |t\rangle|\Psi_t(x)\rangle - \sum_{t=0}^{2T} |t+1\rangle|\Psi_{t+1}(x)\rangle + |2T+1\rangle|\Psi_{2T+1}(x)\rangle \\
&= \sum_{t=0}^{2T+1} |t\rangle|\Psi_t(x)\rangle - \sum_{t=1}^{2T+1} |t\rangle|\Psi_t(x)\rangle = |0\rangle|\Psi_0(x)\rangle = |\tau\rangle.
\end{aligned}$$

We next consider the error of $|w\rangle$ for x, given by $\left\|\Pi_{H(x)^\perp}|w\rangle\right\|^2$. Since $Q_x|j,z,a\rangle \in H(x)$ for all j,z,a, and $|2T+1,0\rangle\Pi_1|\Psi_{2T+1}(x)\rangle \in H_{\text{true}} \subset H(x)$, $\Pi_{H(x)^\perp}|w\rangle = \frac{1}{\sqrt{cT}}|2T+1\rangle|0\rangle\Pi_0|\Psi_{2T+1}(x)\rangle$, so

$$\left\|\Pi_{H(x)^\perp}|w\rangle\right\|^2 = \frac{1}{cT}\left\|\Pi_0|\Psi_{2T+1}(x)\rangle\right\|^2 = \frac{p_0(x)}{cT}.$$

Finally, we compute the positive witness complexity of $|w\rangle$:

$$\begin{aligned}
\||w\rangle\|^2 &= \sum_{t=0}^{2T} \|Q_x|\Psi_t(x)\rangle\|^2 + \|\Pi_1|\Psi_{2T+1}(x)\rangle\|^2 + \frac{1}{cT}\left\|\Pi_0|\Psi_{2T+1}(x)\rangle\right\|^2 \\
&\leq \sum_{t=0}^{2T} \||\Psi_t(x)\rangle\|^2 + \||\Psi_{2T+1}(x)\rangle\|^2 = 2T+2.
\end{aligned}$$

◀

Next, we upper bound $w_-(x)$ whenever $f(x) = 0$:

▶ **Lemma 22.** *For any x that is rejected by $\mathcal{A}$ with probability $p_0(x) > 0$,*

$$w_-(x) \leq \frac{(c+4)T}{p_0(x)}.$$

In particular, if $f(x) = 0$, $w_-(x) \leq \frac{c+4}{2/3}T$, so $W_- \leq \frac{c+4}{2/3}T$.

Proof. We will define a negative witness for x as follows. First, define

$$|\Psi^0_{2T+1}(x)\rangle = \Pi_0|\Psi_{2T+1}(x)\rangle,$$

the rejecting part of the final state. This is non-zero whenever $p_0(x) > 0$. Then for $t \in \{0,\ldots,2T\}$, define

$$|\Psi^0_t(x)\rangle = U^\dagger_{t+1}\ldots U^\dagger_{2T+1}|\Psi^0_{2T+1}(x)\rangle.$$

From this we can define

$$\langle\omega| = \sum_{t=0}^{2T+1} \langle t|\langle\Psi^0_t(x)|.$$

We first observe that

$$\langle\omega|\tau\rangle = \langle\Psi_0^0(x)|0,0,0\rangle = \langle\Psi_{2T+1}^0(x)|U_{2T+1}\dots U_1|0,0,0\rangle = \langle\Psi_{2T+1}^0(x)|\Psi_{2T+1}(x)\rangle = p_0(x).$$

Thus

$$\langle\bar{\omega}| = \frac{1}{p_0(x)}\langle\omega|$$

is a negative witness. Next, we show that $\langle\omega|A\Pi_{H(x)} = 0$. First, for $|t,x_j,j,z,a\rangle \in H_{j,x_j}$ (so $t < 2T$ is odd), we have

$$\begin{aligned}\langle\omega|A|t,x_j,j,z,a\rangle &= \langle\omega|(|t,j,z,a\rangle - (-1)^{x_j}|t+1\rangle|j,z,a\rangle)\\ &= \langle\Psi_t^0(x)|j,z,a\rangle - (-1)^{x_j}\langle\Psi_{t+1}^0(x)|j,z,a\rangle\\ &= \langle\Psi_{t+1}^0(x)|U_{t+1}|j,z,a\rangle - (-1)^{x_j}\langle\Psi_{t+1}^0(x)|j,z,a\rangle\\ &= \langle\Psi_{t+1}^0(x)|\mathcal{O}_x|j,z,a\rangle - (-1)^{x_j}\langle\Psi_{t+1}^0(x)|j,z,a\rangle = 0.\end{aligned}$$

The same argument holds for $|t,0,0,j,z,a\rangle \in H_{\text{true}}$. Similarly, for any $|t,b,j,z,a\rangle \in H_{\text{true}}$ with $t \le 2T$ even, we have

$$\begin{aligned}\langle\omega|A|t,b,j,z,a\rangle &= \langle\omega|(|t,j,z,a\rangle - |t+1\rangle U_{t+1}|j,z,a\rangle)\\ &= \langle\Psi_t^0(x)|j,z,a\rangle - \langle\Psi_{t+1}^0(x)|U_{t+1}|j,z,a\rangle = 0.\end{aligned}$$

Finally, for any $|2T+1,b,j,z,1\rangle \in H_{\text{true}}$, we have

$$\langle\omega|A|2T+1,b,j,z,1\rangle = \langle\omega|2T+1,j,z,1\rangle = \langle\Psi_{2T+1}^0(x)|j,z,1\rangle = 0.$$

Thus $\langle\omega|A\Pi_{H(x)} = 0$ and so $\langle\bar{\omega}|A\Pi_{H(x)} = 0$, and $\langle\bar{\omega}|$ is a negative witness for x in P. To compute its witness complexity, first observe that $\langle\omega|A = \langle\omega|A\Pi_{H(x)^\perp}$, and

$$\begin{aligned}A\Pi_{H(x)^\perp} = &\sum_{s=1}^{T}\sum_{\substack{j\in[n]\cup\{0\},\\ z\in\mathcal{Z},a\in\{0,1\}}}(|2s-1,j,z,a\rangle + (-1)^{x_j}|2s,j,z,a\rangle)\langle 2s-1,\bar{x}_j,j,z,a|\\ &+\sum_{j\in[n]\cup\{0\},z\in\mathcal{Z}}\sqrt{cT}|2T+1,j,z,0\rangle\langle 2T+1,0,j,z,0|\end{aligned}$$

so, using $\langle\Psi_{2s-1}^0(x)|j,z,a\rangle = \langle\Psi_{2s}^0(x)|U_{2s}|j,z,a\rangle = (-1)^{x_j}\langle\Psi_{2s}^0(x)|j,z,a\rangle$, we have:

$$\begin{aligned}&\langle\omega|A\Pi_{H(x)^\perp}\\ =&\sum_{s=1}^{T}\sum_{\substack{j\in[n]\cup\{0\},\\ z\in\mathcal{Z},a\in\{0,1\}}}(\langle\Psi_{2s-1}^0(x)|j,z,a\rangle + (-1)^{x_j}\langle\Psi_{2s}^0(x)|j,z,a\rangle)\langle 2s-1,\bar{x}_j,j,z,a|\\ &+\sum_{j\in[n]\cup\{0\},z\in\mathcal{Z}}\sqrt{cT}\langle\Psi_{2T+1}^0(x)|j,z,0\rangle\langle 2T+1,0,j,z,0|\\ =&\sum_{s=1}^{T}\sum_{j\in[n]\cup\{0\},z\in\mathcal{Z},a\in\{0,1\}}2(-1)^{x_j}\langle\Psi_{2s}^0(x)|j,z,a\rangle)\langle 2s-1,\bar{x}_j,j,z,a|\\ &+\sum_{j\in[n]\cup\{0\},z\in\mathcal{Z}}\sqrt{cT}\langle\Psi_{2T+1}^0(x)|j,z,0\rangle\langle 2T+1,0,j,z,0|.\end{aligned}$$

Thus, the complexity of $\langle\bar{\omega}|$ is:

$$\begin{aligned}
\|\langle\bar{\omega}|A\|^2 &= \frac{1}{p_0(x)^2}\left\|\langle\omega|A\Pi_{H(x)^\perp}\right\|^2 \\
&= \frac{1}{p_0(x)^2}\sum_{s=1}^{T}\sum_{\substack{j\in[n]\cup\{0\},\\ z\in\mathcal{Z},\\ a\in\{0,1\}}} 4\left|\langle\Psi_{2s}^0(x)|j,z,a\rangle\right|^2 + \frac{1}{p_0(x)^2}\sum_{\substack{j\in[n]\cup\{0\},\\ z\in\mathcal{Z}}} cT\left|\langle\Psi_{2T+1}^0(x)|j,z,0\rangle\right|^2 \\
&= \frac{4}{p_0(x)^2}\sum_{s=1}^{T}\left\||\Psi_{2s}^0(x)\rangle\right\|^2 + \frac{cT}{p_0(x)^2}\left\||\Psi_{2T+1}^0(x)\rangle\right\|^2.
\end{aligned}$$

Because each U_t is unitary, we have $\left\||\Psi_{2s}^0(x)\rangle\right\|^2 = \left\||\Psi_{2T+1}^0(x)\rangle\right\|^2 = p_0(x)$, thus:

$$\|\langle\bar{\omega}|A\|^2 = \frac{4T}{p_0(x)} + \frac{cT}{p_0(x)} \le \frac{4+c}{2/3}T \text{ when } f(x)=0. \qquad \blacktriangleleft$$

We conclude the proof of Theorem 11 with the following corollary, from which Theorem 11 follows immediately, by appealing to Claim 18 with $\kappa = \frac{9}{10}$ and κ' any constant in $(0,1)$.

▶ **Corollary 23.** *Let $c = 5$, in the definition of $P_{\mathcal{A}}$. Then:*

- $s(P_{\mathcal{A}}) = 2^{S+O(1)}$
- *If $\mathcal{A}$ decides f with one-sided error, then $P_{\mathcal{A}}$ decides f with complexity $C \le O(T)$.*
- *If $\mathcal{A}$ decides f with bounded error, then $P_{\mathcal{A}}$ $\frac{9}{10}$-approximates f with complexity $C_\kappa \le O(T)$.*

Proof. We first compute $s(P_{\mathcal{A}}) = \dim H$ using the fact that the algorithm uses space

$$S = \log\dim\operatorname{span}\{|j,z,a\rangle : j\in[n]\cup\{0\}, z\in\mathcal{Z}, a\in\{0,1\}\} + \log T:$$

$$\dim H = (\dim\operatorname{span}\{|t,b\rangle : t\in\{0,\dots,2T+1\}, b\in\{0,1\}\})2^{S-\log T} = 2^{S+O(1)}.$$

We prove the third statement, as the second is similar. By Lemma 22, using $c=5$, we have

$$W_- \le \frac{5+4}{2/3}T = \frac{27}{2}T.$$

By Lemma 21, we can see that for every x such that $f(x)=1$, there is an approximate positive witness $|w\rangle$ for x with error at most:

$$\frac{\varepsilon}{cT} = \frac{1/3}{5T} \le \frac{1}{15T}\frac{\frac{27}{2}T}{W_-} = \frac{9}{10}\frac{1}{W_-}.$$

Furthermore, $\||w\rangle\|^2 \le 2T+2$, so $\widehat{W}_+ \le 2T+2$. Observing $C_\kappa = \sqrt{W_-\widehat{W}_+} \le \sqrt{27T(T+1)}$ completes the proof. $\blacktriangleleft$

4 Span Programs and Space Complexity

Using the transformation from algorithms to span programs from Section 3.2, we immediately have the following connections between span program size and space complexity.

▶ **Theorem 24.** *For any $f: D\to\{0,1\}$ for $D\subseteq\{0,1\}^n$, we have*

$$\mathsf{S}_U(f) \ge \Omega\left(\log\widetilde{\mathsf{SP}}(f)\right) \qquad \text{and} \qquad \mathsf{S}_U^1(f) \ge \Omega\left(\log\mathsf{SP}(f)\right).$$

Theorem 24 is a corollary of Theorem 11. Theorem 17 shows that the lower bound for $\mathsf{S}_U(f)$ in Theorem 24 is part of a *tight* correspondence between space complexity and $\log s(P) + \log C(P)$.

Theorem 2.9 of [3] gives a lower bound of $\mathsf{SP}(f) \geq \Omega(2^{n/3}/(n \log n)^{1/3})$ for almost all n-bit Boolean functions. Combined with Theorem 24, we immediately have:

▶ **Theorem 25.** *For almost all Boolean functions $f : \{0,1\}^n \to \{0,1\}$, $\mathsf{S}^1_U(f) = \Omega(n)$.*

Ideally, we would like to use the lower bound in Theorem 24 to prove a non-trivial lower bound for $\mathsf{S}_U(f)$ or $\mathsf{S}^1_U(f)$ for some concrete f. Fortunately, there are somewhat nice expressions lower bounding $\mathsf{SP}(f)$ [15, 7], which we extend to lower bounds of $\widetilde{\mathsf{SP}}(f)$ in the remainder of this section. However, on the unfortunate side, there has already been significant motivation to instantiate these expressions to non-trivial lower bounds for concrete f, with no success. There has been some success in *monotone* versions of these lower bounds, which we discuss more in Section 5.

For a function $f : D \to \{0,1\}$ for $D \subseteq \{0,1\}^n$, and an index $j \in [n]$, we let $\Delta_{f,j} \in \{0,1\}^{f^{-1}(0)\times f^{-1}(1)}$ be defined by $\Delta_{f,j}[y,x] = 1$ if and only if $x_j \neq y_j$. When f is clear from context, we simply denote this by Δ_j. The following tight characterization of $\mathsf{SP}(f)$ may be found in, for example, [13].

▶ **Lemma 26.** *For any $f : D \to \{0,1\}$ for $D \subseteq \{0,1\}^n$,*

$$\begin{aligned} \mathsf{SP}(f) = \text{minimize} \quad & \sum_{j\in[n]} \operatorname{rank}(\Lambda_j) \\ \text{subject to} \quad & \forall j \in [n], \Lambda_j \in \mathbb{R}^{f^{-1}(0)\times f^{-1}(1)} \\ & \sum_{j\in[n]} \Lambda_j \circ \Delta_j = J, \end{aligned}$$

where J is the $f^{-1}(0) \times f^{-1}(1)$ all-ones matrix.

By Theorem 24, the logarithm of the above is a lower bound on $\mathsf{S}^1_U(f)$. We modify Lemma 26 to get the following approximate version, whose logarithm lower bounds $\mathsf{S}_U(f)$ when $\kappa = \frac{1}{4}$.

▶ **Lemma 27.** *For any $\kappa \in [0,1)$, and $f : D \to \{0,1\}$ for $D \subseteq \{0,1\}^n$,*

$$\begin{aligned} \widetilde{\mathsf{SP}}_\kappa(f) \geq \text{minimize} \quad & \sum_{j\in[n]} \operatorname{rank}(\Lambda_j) \qquad (1) \\ \text{subject to} \quad & \forall j \in [n], \Lambda_j \in \mathbb{R}^{f^{-1}(0)\times f^{-1}(1)} \\ & \left\| \sum_{j\in[n]} \Lambda_j \circ \Delta_j - J \right\|_\infty \leq \sqrt{\kappa}. \end{aligned}$$

Proof. Fix a span program that κ-approximates f with $s(P) = \widetilde{\mathsf{SP}}_\kappa(f)$, and let $\{\langle\omega_y| : y \in f^{-1}(0)\}$ be optimal negative witnesses, and $\{|w_x\rangle : x \in f^{-1}(1)\}$ be approximate positive witnesses with $\left\|\Pi_{H(x)}|w_x\rangle\right\|^2 \leq \frac{\kappa}{W_-}$. Letting $\Pi_{j,b}$ denote the projector onto $H_{j,b}$, define

$$\Lambda_j = \sum_y |y\rangle\langle\omega_y|A\Pi_{j,\bar{y}_j} \sum_x \Pi_{j,x_j}|w_x\rangle\langle x|,$$

so Λ_j has rank at most $\dim H_j$, and so $\sum_{j\in[n]} \operatorname{rank}(\Lambda_j) \leq s(P) = \widetilde{\mathsf{SP}}_\kappa(f)$.

We now show that $\{\Lambda_j\}_j$ is a feasible solution. Let $|\mathsf{err}(x)\rangle$ be the positive witness error of $|w_x\rangle$, $|\mathsf{err}(x)\rangle = \Pi_{H(x)^\perp}|w_x\rangle = \sum_{j=1}^n \Pi_{j,\bar{x}_j}|w_x\rangle$. Then we have:

$$\begin{aligned}
\langle y|\sum_{j=1}^n \Lambda_j \circ \Delta_j|x\rangle &= \langle\omega_y|A\sum_{j:x_j\neq y_j}\Pi_{j,x_j}|w_x\rangle \\
&= \langle\omega_y|A\left(|w_x\rangle - \sum_{j:x_j=y_j}\Pi_{j,x_j}|w_x\rangle - |\mathsf{err}(x)\rangle\right) \\
&= \langle\omega_y|\tau\rangle - \langle\omega_y|A\sum_{j:x_j=y_j}\Pi_{H(y)}\Pi_{j,x_j}|w_x\rangle - \langle\omega_y|A|\mathsf{err}(x)\rangle \\
&= 1 - 0 - \langle\omega_y|A|\mathsf{err}(x)\rangle \\
\left|1 - \langle y|\sum_{j=1}^n \Lambda_j\circ\Delta_j|x\rangle\right| &\leq \|\langle\omega_y|A\|\,\||\mathsf{err}(x)\rangle\| = \sqrt{w_-(y)\frac{\kappa}{W_-}} \leq \sqrt{\kappa}.
\end{aligned}$$

Above we used the fact that $\langle\omega_y|A\Pi_{H(y)} = 0$. Thus, $\{\Lambda_j\}_j$ is a feasible solution with objective value $\leq \widetilde{\mathsf{SP}}_\kappa(f)$, so the result follows. ◀

As a corollary of the above, and the connection between span program size and unitary quantum space complexity stated in Theorem 24, the logarithm of the expression in (1) with $\kappa = \frac{1}{4}$ is a lower bound on $\mathsf{S}_U(f)$, and with $\kappa = 0$, it is a lower bound on $\mathsf{S}_U^1(f)$. However, as stated, it is difficult to use this expression to prove an explicit lower bound, because it is a minimization problem. We will shortly give a lower bound in terms of a maximization problem, making it possible to obtain explicit lower bounds by exhibiting a feasible solution.

A *partial matrix* is a matrix $M \in (\mathbb{R}\cup\{\star\})^{f^{-1}(0)\times f^{-1}(1)}$. A *completion* of M is any $\overline{M} \in \mathbb{R}^{f^{-1}(0)\times f^{-1}(1)}$ such that $\overline{M}[y,x] = M[y,x]$ whenever $M[y,x]\neq\star$. For a partial matrix M, define $\mathrm{rank}(M)$ to be the smallest rank of any completion of M, and ε-$\mathrm{rank}(M)$ to be the smallest rank of any $\tilde{M}$ such that $|M[y,x]-\tilde{M}[y,x]|\leq\varepsilon$ for all y,x such that $M[y,x]\neq\star$. Let $M\circ\Delta_i$ to be the partial matrix defined:

$$M\circ\Delta_i[y,x] = \begin{cases} M[y,x] & \text{if } \Delta_i[y,x]=1 \\ 0 & \text{if } \Delta_i[y,x]=0. \end{cases}$$

Then we have the following:

▶ **Lemma 28.** *For all Boolean functions $f: D\to\{0,1\}$, with $D\subseteq\{0,1\}^n$, and all partial matrices $M\in(\mathbb{R}\cup\{\star\})^{f^{-1}(0)\times f^{-1}(1)}$ such that $\max\{|M[y,x]| : M[y,x]\neq\star\}\leq 1$:*

$$\mathsf{S}_U^1(f) \geq \Omega\left(\log\left(\frac{\mathrm{rank}(M)}{\max_{i\in[n]}\mathrm{rank}(M\circ\Delta_i)}\right)\right).$$

In [15], Razborov showed that the expression on the right-hand side in Lemma 28 is a lower bound on the logarithm of the *formula size* of f (Ref. [7] related this to $\mathsf{SP}(f)$). Later, in [16], Razborov noted that when restricted to non-partial matrices, this can never give a better bound than n. Thus, to prove a non-trivial lower bound on $\mathsf{S}_U^1(f)$ using this method, one would need to use a partial matrix. We prove the following generalization to the approximate case.

▶ **Lemma 29.** *For all Boolean functions $f: D\to\{0,1\}$, with $D\subseteq\{0,1\}^n$, and all partial matrices $M\in(\mathbb{R}\cup\{\star\})^{f^{-1}(0)\times f^{-1}(1)}$ such that $\max\{|M[y,x]| : M[y,x]\neq\star\}\leq 1$:*

$$\mathsf{S}_U(f) \geq \Omega\left(\log\left(\frac{\frac{1}{2}\text{-}\mathrm{rank}(M)}{\max_{i\in[n]}\mathrm{rank}(M\circ\Delta_i)}\right)\right).$$

Proof. Let $\{\Lambda_j\}_j$ be an optimal feasible solution for the expression from Lemma 27, so

$$\widetilde{\mathsf{SP}}_\kappa(f) \geq \sum_{j\in[n]} \operatorname{rank}(\Lambda_j), \quad \text{and} \quad \left\| \sum_{j\in[n]} \Lambda_j \circ \Delta_j - J \right\|_\infty \leq \sqrt{\kappa}.$$

Let $\overline{M}_j$ be a completion of $M \circ \Delta_j$ with $\operatorname{rank}(M \circ \Delta_j) = \operatorname{rank}(\overline{M}_j)$. Then for any x, y such that $M[y,x] \neq \star$:

$$\begin{aligned}\left| \left(\sum_{j\in[n]} \overline{M}_j \circ \Lambda_j \right)[y,x] - M[y,x] \right| &= \left| \sum_{j\in[n]} M[y,x]\Delta_j[y,x]\Lambda_j[y,x] - M[y,x] \right| \\ &\leq |M[y,x]| \left\| \sum_{j\in[n]} \Delta_j \circ \Lambda_j - J \right\|_\infty \leq \sqrt{\kappa}.\end{aligned}$$

Thus

$$\sqrt{\kappa}\text{-}\operatorname{rank}(M) \leq \operatorname{rank}\left(\sum_{j\in[n]} \overline{M}_j \circ \Lambda_j \right) \leq \sum_{j\in[n]} \operatorname{rank}(\overline{M}_j \circ \Lambda_j).$$

Using the fact that for any matrices B and C, $\operatorname{rank}(B \circ C) \leq \operatorname{rank}(B)\operatorname{rank}(C)$, we have

$$\sqrt{\kappa}\text{-}\operatorname{rank}(M) \leq \sum_{j\in[n]} \operatorname{rank}(\Lambda_j)\operatorname{rank}(\overline{M}_j) \leq \widetilde{\mathsf{SP}}_\kappa(f) \max_{j\in[n]} \operatorname{rank}(M \circ \Delta_j).$$

Setting $\kappa = \frac{1}{4}$, and noting that by Theorem 24, $\mathsf{S}_U(f) \geq \log \widetilde{\mathsf{SP}}(f) = \log \widetilde{\mathsf{SP}}_{1/4}(f)$ completes the proof. ◀

Unfortunately, as far as we are aware, nobody has used this lower bound to successfully prove any concrete formula size lower bound of $2^{\omega(\log n)}$, so it seems to be quite difficult. However, there has been some success proving lower bounds in the monotone span program case, even without resorting to partial matrices, which we discuss in the next section.

5 Monotone Span Programs and Monotone Algorithms

A monotone function is a Boolean function in which $y \leq x$ implies $f(y) \leq f(x)$, where $y \leq x$ should be interpreted bitwise. In other words, flipping 0s to 1s in the input either keeps the function value the same, or changes it from 0 to 1. A monotone span program is a span program in which $H_{i,0} = \{0\}$ for all i, so only 1-valued queries contribute to $H(x)$, and $H(y) \subseteq H(x)$ whenever $y \leq x$. A monotone span program can only decide or approximate a monotone function.

▶ **Definition 30.** *For a monotone function f, define the* monotone span program size, *denoted $\mathsf{mSP}(f)$, as the minimum $s(P)$ over (families of) monotone span programs P such that P decides f; and the* approximate monotone span program size, *denoted $\widetilde{\mathsf{mSP}}_\kappa(f)$, as the minimum $s(P)$ over (families of) monotone span programs P such that P κ-approximates f. We let $\widetilde{\mathsf{mSP}}(f) = \widetilde{\mathsf{mSP}}_{1/4}(f)$.*

In contrast to $\mathsf{SP}(f)$, there are non-trivial lower bounds for $\mathsf{mSP}(f)$ for explicit monotone functions f. However, this does *not* necessarily give a lower bound on $\mathsf{SP}(f)$, and in particular, may not be a lower bound on the one-sided error quantum space complexity of f. However,

lower bounds on $\log \mathsf{mSP}(f)$ or $\log \widetilde{\mathsf{mSP}}(f)$ do give lower bounds on the space complexity of quantum algorithms obtained from monotone span programs, and as we will soon see, $\log \mathsf{mSP}(f)$ and $\log \widetilde{\mathsf{mSP}}(f)$ are lower bounds on the space complexity of *monotone phase estimation algorithms*, described in Section 5.2. The strongest known lower bound on $\mathsf{mSP}(f)$ is the following:

▶ **Theorem 31** ([14]). *There is an explicit Boolean function $f : D \to \{0,1\}$ for $D \subseteq \{0,1\}^n$ such that*

$$\log \mathsf{mSP}(f) \geq \Omega(n).$$

We will adapt some of the techniques used in existing lower bounds on mSP to show a lower bound on $\widetilde{\mathsf{mSP}}(f)$ for some explicit f:

▶ **Theorem 32.** *There is an explicit Boolean function $f : D \to \{0,1\}$ for $D \subseteq \{0,1\}^n$ such that for any constant κ,*

$$\log \widetilde{\mathsf{mSP}}_\kappa(f) \geq (\log n)^{2-o(1)}.$$

In particular, this implies a lower bound of $2^{(\log n)^{2-o(1)}}$ on $\mathsf{mSP}(f)$ for the function f in Theorem 32. We prove Theorem 32 in Section 5.1. Theorem 32 implies that any quantum algorithm for f obtained from a monotone span program must have space complexity $(\log n)^{2-o(1)}$, which is slightly better than the trivial lower bound of $\Omega(\log n)$. In Section 5.2, we describe a more natural class of algorithms called monotone phase estimation algorithms such that $\log \widetilde{\mathsf{mSP}}(f)$ is a lower bound on the quantum space complexity of any such algorithm computing f with bounded error. Then for the specific function f from Theorem 32, any monotone phase estimation algorithm for f must use space $(\log n)^{2-o(1)}$.

5.1 Monotone Span Program Lower Bounds

Our main tool in proving Theorem 32 will be the following.

▶ **Theorem 33.** *For any Boolean function $f : D \to \{0,1\}$, $D \subseteq \{0,1\}^n$, and any constant $\kappa \in [0,1)$:*

$$\widetilde{\mathsf{mSP}}_\kappa(f) \geq \max_{M \in \mathbb{R}^{f^{-1}(0) \times f^{-1}(1)} : \|M\|_\infty \leq 1} \frac{\sqrt{\kappa}\text{-rank}(M)}{\max_{j \in [n]} \text{rank}(M \circ \Delta_{j,1})},$$

where $\Delta_{j,1}[y,x] = 1$ if $y_i = 0$ and $x_i = 1$, and 0 else.

When, $\kappa = 0$, the right-hand side of the equation in Theorem 33 is the (monotone) *rank measure*, defined in [15], and shown in [7] to lower bound monotone span program size. We extend the proof for the $\kappa = 0$ case to get a lower bound on approximate span program size. We could also allow for partial matrices M, as in the non-monotone case (Lemma 29) but unlike the non-monotone case, it is not necessary to consider partial matrices to get non-trivial lower bounds.

Proof. Fix a monotone span program that κ-approximates f with size $\widetilde{\mathsf{mSP}}_\kappa(f)$. Let $\{\langle \omega_y| : y \in f^{-1}(0)\}$ be optimal negative witnesses, and let $\{|w_x\rangle : x \in f^{-1}(1)\}$ be approximate positive witnesses with $\left\|\Pi_{H(x)^\perp}|w_x\rangle\right\|^2 \leq \frac{\kappa}{W_-}$. Letting $\Pi_{j,b}$ denote the projector onto $H_{j,b}$, define

$$\begin{aligned}\Lambda_j &= \sum_{y\in f^{-1}(0)} |y\rangle\langle\omega_y|A\Pi_{j,\bar{y}_j} \sum_{x\in f^{-1}(1)} \Pi_{j,x_j}|w_x\rangle\langle x| \\ &= \sum_{\substack{y\in f^{-1}(0):\\ y_j=0}} |y\rangle\langle\omega_y|A\Pi_{j,1} \sum_{\substack{x\in f^{-1}(1):\\ x_j=1}} \Pi_{j,1}|w_x\rangle\langle x|,\end{aligned}$$

so Λ_j has rank at most $\dim H_j$, and so $\sum_{j\in[n]} \text{rank}(\Lambda_j) \leq s(P) = \widetilde{\mathsf{mSP}}_\kappa(f)$. Furthermore, Λ_j is only supported on (y,x) such that $y_j = 0$ and $x_j = 1$, so $\Lambda_j \circ \Delta_{j,1} = \Lambda_j$. Denoting the error of $|w_x\rangle$ as $|\mathsf{err}(x)\rangle = \Pi_{H(x)^\perp}|w_x\rangle = \sum_{j:x_j=0} \Pi_{j,1}|w_x\rangle$, we have

$$\begin{aligned}\langle y|\sum_{j\in[n]} \Lambda_j|x\rangle &= \sum_{j:y_j=0,x_j=1} \langle\omega_y|A\Pi_{j,1}|w_x\rangle = \langle\omega_y|A\sum_{j:y_j=0}\Pi_{j,1}\sum_{j:x_j=1}\Pi_{j,1}|w_x\rangle \\ &= \langle\omega_y|A(|w_x\rangle - |\mathsf{err}(x)\rangle) = \langle\omega_y|A|w_x\rangle - \langle\omega_y|A|\mathsf{err}(x)\rangle\end{aligned}$$

$$\left|1 - \langle y|\sum_{j\in[n]} \Lambda_j|x\rangle\right| \leq 1 - 1 + \|\langle\omega_y|A\| \, \||\mathsf{err}(x)\rangle\| \leq \sqrt{W_-}\sqrt{\frac{\kappa}{W_-}} = \sqrt{\kappa}.$$

Then for any $M \in \mathbb{R}^{f^{-1}(0)\times f^{-1}(1)}$ with $\|M\|_\infty \leq 1$, we have:

$$\left\|M - M\circ\sum_{j\in[n]}\Lambda_j\right\|_\infty \leq \|M\|_\infty \left\|J - \sum_{j\in[n]}\Lambda_j\right\|_\infty \leq \sqrt{\kappa}.$$

Thus

$$\begin{aligned}\sqrt{\kappa}\text{-rank}(M) &\leq \text{rank}\left(M\circ\sum_{j\in[n]}\Lambda_j\right) \leq \sum_{j\in[n]}\text{rank}(M\circ\Lambda_j) = \sum_{j\in[n]}\text{rank}(M\circ\Delta_{j,1}\circ\Lambda_j) \\ &\leq \sum_{j\in[n]}\text{rank}(M\circ\Delta_{j,1})\text{rank}(\Lambda_j) \leq \widetilde{\mathsf{mSP}}_\kappa(f)\max_{j\in[n]}\text{rank}(M\circ\Delta_{j,1}).\end{aligned}$$ ◀

To show a lower bound on $\widetilde{\mathsf{mSP}}(f)$ for *some* explicit $f : \{0,1\}^n \to \{0,1\}$, it turns out to be sufficient to find some high approximate rank matrix $M \in \mathbb{R}^{Y\times X}$ for finite sets X and Y, and a *rectangle cover* of M, $\Delta_1, \ldots, \Delta_n$, where each $\Delta_i \circ M$ has low rank. Specifically, we have the following lemma, which, with rank in place of approximate rank, has been used extensively in previous monotone span program lower bounds.

▶ **Lemma 34.** *Let $M \in \mathbb{R}^{Y\times X}$ with $\|M\|_\infty \leq 1$, for some finite sets X and Y and $X_1, \ldots, X_n \subseteq X$, $Y_1, \ldots, Y_n \subseteq Y$ be such that for all $(x,y) \in X\times Y$, there exists $j \in [n]$ such that $(x,y) \in X_j \times Y_j$. Define $\Delta_j \in \{0,1\}^{Y\times X}$ by $\Delta_j[y,x] = 1$ if and only if $(y,x) \in Y_j \times X_j$. There exists a monotone function $f : D \to \{0,1\}$ for $D \subseteq \{0,1\}^n$ such that for any constant $\kappa \in [0,1)$:*

$$\widetilde{\mathsf{mSP}}_\kappa(f) \geq \frac{\sqrt{\kappa}\text{-rank}(M)}{\max_{j\in[n]}\text{rank}(M\circ\Delta_j)}.$$

Proof. For each $y \in Y$, define $t^y \in \{0,1\}^n$ by:

$$t^y_j = \begin{cases} 0 & \text{if } y \in Y_j \\ 1 & \text{else.}\end{cases}$$

Similarly, for each $x \in X$, define $s^x \in \{0,1\}^n$ by:

$$s^x_j = \begin{cases} 1 & \text{if } x \in X_j \\ 0 & \text{else.} \end{cases}$$

For every $(y,x) \in Y \times X$, there is some j such that $y_j \in Y_j$ and $x_j \in X_j$, so it can't be the case that $s^x \leq t^y$. Thus, we can define f as the unique monotone function such that $f(s) = 1$ for every $s \in \{0,1\}^n$ such that $s^x \leq s$ for some $x \in X$, and $f(t) = 0$ for all $t \in \{0,1\}^n$ such that $t \leq t^y$ for some $y \in Y$. Then we can define a matrix $M' \in \mathbb{R}^{f^{-1}(0) \times f^{-1}(1)}$ by $M'[t^y, s^x] = M[y,x]$ for all $(y,x) \in Y \times X$, and 0 elsewhere. We have $\varepsilon\text{-rank}(M') = \varepsilon\text{-rank}(M)$ for all ε, and $\text{rank}(M' \circ \Delta_{j,1}) = \text{rank}(M \circ \Delta_j)$ for all j. The result then follows from Theorem 33. ◀

We will prove Theorem 32 by constructing an M with high approximate rank, and a good rectangle cover $\{X_j \times Y_j\}_j$. Following [19] and [14], we will make use of a technique due to Sherstov for proving communication lower bounds, called the *pattern matrix method* [20]. We begin with some definitions.

▶ **Definition 35** (Fourier spectrum). *For a real-valued function $p : \{0,1\}^m \to \mathbb{R}$, its Fourier coefficients are defined, for each $S \subseteq [m]$:*

$$\hat{p}(S) = \frac{1}{2^m} \sum_{z \in \{0,1\}^m} p(z)\chi_S(z),$$

where $\chi_S(z) = (-1)^{\sum_{i \in S} z_i}$. It is easily verified that $p = \sum_{S \subseteq [m]} \hat{p}(S)\chi_S$.

▶ **Definition 36** (Degree and approximate degree). *The degree of a function $p : \{0,1\}^m \to \mathbb{R}$ is defined $\deg(p) = \max\{|S| : \hat{p}(S) \neq 0\}$. For any $\varepsilon \geq 0$, $\widetilde{\deg}_\varepsilon(p) = \min\{\deg(\tilde{p}) : \|p - \tilde{p}\|_\infty \leq \varepsilon\}$.*

Pattern matrices, defined by Sherstov in [20], are useful for proving lower bounds in communication complexity, because their rank and approximate rank are relatively easy to lower bound. In [19], Robere, Pitassi, Rossman and Cook first used this analysis to give lower bounds on $\mathsf{mSP}(f)$ for some f. We now state the definition, using the notation from [14], which differs slightly from [20].

▶ **Definition 37** (Pattern matrix). *For a real-valued function $p : \{0,1\}^m \to \mathbb{R}$, and a positive integer λ, the (m, λ, p)-pattern matrix is defined as $F \in \mathbb{R}^{\{0,1\}^{\lambda m} \times ([\lambda]^m \times \{0,1\}^m)}$ where for $y \in \{0,1\}^{\lambda m}$, $x \in [\lambda]^m$, and $w \in \{0,1\}^m$,*

$$F[y,(x,w)] = f(y|_x \oplus w),$$

where by $y|_x$, we mean the m-bit string containing one bit from each λ-sized block of y as specified by the entries of x: $(y^{(1)}_{x_1}, y^{(2)}_{x_2}, \ldots, y^{(m)}_{x_m})$, where $y^{(i)} \in \{0,1\}^\lambda$ is the i-th block of y.

For comparison, what [20] calls an (n,t,p)-pattern matrix would be a $(t, n/t, p)$-pattern matrix in our notation. As previously mentioned, a pattern matrix has the nice property that its rank (or even approximate rank) can be lower bounded in terms of properties of the Fourier spectrum of p. In particular, the following is proven in [20]:

▶ **Lemma 38.** *Let F be the (m, λ, p)-pattern matrix for $p : \{0,1\}^m \to \{-1,+1\}$. Then for any $\varepsilon \in [0,1]$ and $\delta \in [0, \varepsilon]$, we have:*

$$\text{rank}(F) = \sum_{S \subseteq [m] : \hat{p}(S) \neq 0} \lambda^{|S|} \quad \text{and} \quad \delta\text{-rank}(F) \geq \lambda^{\widetilde{\deg}_\varepsilon(p)} \frac{(\varepsilon - \delta)^2}{(1+\delta)^2}.$$

This shows that we can use functions p of high approximate degree to construct pattern matrices $F \in \mathbb{R}^{\{0,1\}^{\lambda m} \times ([\lambda]^m \times \{0,1\}^m)}$ of high approximate rank. To apply Lemma 34, we also need to find a good rectangle cover of some F.

A *b-certificate* for a function p on $\{0,1\}^m$ is an assignment $\alpha : S \to \{0,1\}$ for some $S \subseteq [m]$ such that for any $x \in \{0,1\}^m$ such that $x_j = \alpha(j)$ for all $j \in S$, $f(x) = b$. The size of a certificate is $|S|$. The following shows how to use the certificates of p to construct a rectangle cover of its pattern matrix.

▶ **Lemma 39.** *Let $p : \{0,1\}^m \to \{-1,+1\}$, and suppose there is a set of ℓ certificates for p of size at most C such that every input satisfies at least one certificate. Then for any positive integer λ, there exists a function $f : \{0,1\}^n \to \{0,1\}$ for $n = \ell(2\lambda)^C$ such that for any $\kappa \in (0,1)$ and $\varepsilon \in [\sqrt{\kappa}, 1]$:*

$$\widetilde{\mathsf{mSP}}_\kappa(f) \geq \Omega\left((\varepsilon - \sqrt{\kappa})^2 \lambda^{\widetilde{\deg}_\varepsilon(p)}\right).$$

Proof. For $i = 1, \ldots, \ell$, let $\alpha_i : S_i \to \{0,1\}$ for $S_i \subset [m]$ of size $|S_i| \leq C$ be one of the ℓ certificates. That is, for each i, there is some $v_i \in \{-1,+1\}$ such that for any $x \in \{0,1\}^m$, if $x_j = \alpha_i(j)$ for all $j \in S_i$, then $p(x) = v_i$ (so α_i is a v_i-certificate).

We let F be the (m, λ, p)-pattern matrix, which has $\|F\|_\infty = 1$ since p has range $\{-1,+1\}$. We will define a rectangle cover as follows. For every $i \in [\ell]$, $k \in [\lambda]^{S_i}$, and $b \in \{0,1\}^{S_i}$, define:

$$X_{i,k,b} = \{(x,w) \in [\lambda]^m \times \{0,1\}^m : \forall j \in S_i, w_j = b_j, x_j = k_j\}$$
$$Y_{i,k,b} = \{y \in \{0,1\}^{\lambda m} : \forall j \in S_i, y^{(j)}_{k_j} = b_j \oplus \alpha_i(j)\}.$$

We first note that this is a rectangle cover. Fix any $y \in \{0,1\}^{\lambda m}$, $x \in [\lambda]^m$ and $w \in \{0,1\}^m$. First note that for any i, if we let b be the restriction of w to S_i, and k the restriction of x to S_i, we have $(x,w) \in X_{i,k,b}$. This holds in particular for i such that α_i is a certificate for $y|_x \oplus w$, and by assumption there is at least one such i. For such an i, we have $y^{(j)}_{x_j} \oplus w_j = \alpha(j)$ for all $j \in S_i$, so $y \in Y_{i,k,b}$. Thus, we can apply Lemma 34.

Note that if $(x,w) \in X_{i,k,b}$, and $y \in Y_{i,k,b}$, then $(y|_x \oplus w)[j] = y^{(j)}_{x_j} \oplus w_j = \alpha_i(j)$ for all $j \in S_i$, so $p(y|_x \oplus w) = v_i$. Letting $\Delta_{i,k,b}[y,(x,w)] = 1$ if $y \in Y_{i,k,b}$ and $(x,w) \in X_{i,k,b}$, and 0 else, we have that if $y \in Y_{i,k,b}$ and $(x,w) \in X_{i,k,b}$, $(F \circ \Delta_{i,k,b})[y,(x,w)] = p(y|_x \oplus w) = v_i$, and otherwise, $(F \circ \Delta_{i,k,b})[y,(x,w)] = 0$. Thus $\operatorname{rank}(F \circ \Delta_{i,k,b}) = \operatorname{rank}(v_i \Delta_{i,k,b}) = 1$. Then by Lemma 34, there exists $f : \{0,1\}^n \to \{0,1\}$ where $n = \sum_{i=1}^{\ell} (2\lambda)^{|S_i|} \leq \ell(2\lambda)^C$ such that:

$$\widetilde{\mathsf{mSP}}_\kappa(f) \geq \sqrt{\kappa}\text{-}\operatorname{rank}(F) \geq \lambda^{\widetilde{\deg}_\varepsilon(p)} \frac{(\varepsilon - \sqrt{\kappa})^2}{(1+\sqrt{\kappa})^2}, \text{ by Lemma 38.} \quad ◀$$

We now prove Theorem 32, restated below:

▶ **Theorem 32.** *There is an explicit Boolean function $f : D \to \{0,1\}$ for $D \subseteq \{0,1\}^n$ such that for any constant κ,*

$$\log \widetilde{\mathsf{mSP}}_\kappa(f) \geq \Omega((\log n)^{2-o(1)}).$$

Proof. By [5, Theorem 38], there is a function p with $\widetilde{\deg}_{1/3}(p) \geq C(p)^{2-o(1)}$, which is, up to the $o(1)$ in the exponent, the best possible separation between these two quantities. In particular, this function has $\widetilde{\deg}_{1/3}(p) \geq M^{2-o(1)}$, and $C(p) \leq M^{1+o(1)}$, where $C(p)$ is the certificate complexity of p, for some parameter M (see [5] equations (64) and (65), where

p is referred to as F), and p is a function on $M^{2+o(1)}$ variables (see [5], discussion above equation (64)). Thus, there are at most $\binom{M^{2+o(1)}}{M^{1+o(1)}}$ possible certificates of size $M^{1+o(1)}$ such that each input satisfies at least one of them.

Then by Lemma 39 there exists a function $f : \{0,1\}^n \to \{0,1\}$ for some n such that $n \leq \binom{M^{2+o(1)}}{M^{1+o(1)}}(2\lambda)^{M^{1+o(1)}}$ such that for constant $\kappa < 1/36$ and constant λ:

$$\log \widetilde{\mathsf{mSP}}_\kappa(f) \geq \Omega(\widetilde{\deg}_{1/3}(p) \log \lambda) \geq M^{2-o(1)}.$$

Then we have:

$$\log n \leq \log \binom{M^{2+o(1)}}{M^{1+o(1)}} + M^{1+o(1)} \log(2\lambda) = O(M^{1+o(1)} \log M) = M^{1+o(1)}.$$

Thus, $\log \widetilde{\mathsf{mSP}}_\kappa(f) \geq (\log n)^{2-o(1)}$, and the result for *any* κ follows using Corollary 19. ◀

Since for all total functions p, $\widetilde{\deg}_{1/3}(p) \leq C(p)^2$, where $C(p)$ is the certificate complexity of p, Lemma 39 can't prove a lower bound better than $\log \widetilde{\mathsf{mSP}}(p) \geq (\log n)^2$ for any n-bit function. We state a more general version of Lemma 39 that might have the potential to prove a better bound, but we leave this as future work.

▶ **Lemma 40.** *Fix $p : \{0,1\}^m \to \{-1,+1\}$. For $i = 1, \ldots, \ell$, let $\alpha_i : S_i \to \{0,1\}$ for $S_i \subseteq [m]$ be a partial assignment such that every $z \in \{0,1\}^m$ satisfies at least one of the assignments. Let p_i denote the restriction of p to strings z satisfying the assignment α_i. Then for every positive integer λ, there exists a function $f : \{0,1\}^n \to \{0,1\}$, where $n = \sum_{i=1}^{\ell}(2\lambda)^{|S_i|}$ such that for any $\kappa \in (0,1)$ and $\varepsilon \in [\sqrt{\kappa}, 1]$:*

$$\widetilde{\mathsf{mSP}}_\kappa(f) \geq \Omega\left(\frac{(\varepsilon - \sqrt{\kappa})^2 \lambda^{\widetilde{\deg}_\varepsilon(p)}}{\max_{i \in [\ell]} \sum_{S \subseteq [m] \setminus S_i : \hat{p}_i(S) \neq 0} \lambda^{|S|}}\right).$$

To make use of this lemma, one needs a function p of high approximate degree, such that for every input, there is a small assignment that lowers the degree to something small. This generalizes Lemma 39 because a certificate is an assignment that lowers the degree of the remaining sub-function to constant. However, we note that a p with these conditions is necessary but may not be sufficient for proving a non-trivial lower bound, because while $\sum_{S : \hat{p}_i(S) \neq 0} \lambda^{|S|} \geq \lambda^{\deg(p_i)}$, it may also be much larger if p_i has a dense Fourier spectrum.

Proof. Let F be the (m, λ, p)-pattern matrix. Let $\{X_{i,k,b} \times Y_{i,k,b}\}_{i,k,b}$ be the same rectangle covered defined in the proof of Lemma 39, with the difference that since the α_i are no longer certificates, the resulting submatrices of F may not have constant rank.

Let $\Delta_{i,k,b} = \sum_{y \in Y_{i,k,b}} |y\rangle \sum_{(x,w) \in X_{i,k,b}} \langle x, w|$. Then

$$F \circ \Delta_{i,k,b} = \sum_{y \in Y_{i,k,b}, (x,w) \in X_{i,k,b}} p(y|_x \oplus w) |y\rangle\langle x, w|.$$

Note that when $y \in Y_{i,k,b}$ and $(x,w) \in X_{i,b,k}$, $y|_x \oplus w$ satisfies α_i, so $p(y|_x \oplus w) = p_i(y'|_{x'} \oplus w')$, where y', x' and w' are restrictions of $y \in (\{0,1\}^\lambda)^m$, $x \in [\lambda]^m$ and $w \in \{0,1\}^m$ to $[m] \setminus S_i$. Thus, continuing from above, and rearranging registers, we have:

$$\begin{aligned} F \circ \Delta_{i,k,b} &= \sum_{y' \in (\{0,1\}^\lambda)^{[m] \setminus S_i}} \sum_{\substack{x' \in [\lambda]^{[m] \setminus S_i}, \\ w' \in \{0,1\}^{[m] \setminus S_i}}} p_i(y'|_{x'} \oplus w') |y'\rangle\langle x', w'| \otimes \sum_{\substack{\bar{y} \in (\{0,1\}^\lambda)^{S_i} : \\ \bar{y}|_k = b \oplus \alpha_i}} |\bar{y}\rangle\langle k, b| \\ &= F_i \otimes J_{2^{(\lambda-1)|S_i|}, 1} \end{aligned}$$

where F_i is the (m, λ, p_i)-pattern matrix, and $J_{a,b}$ is the all-ones matrix of dimension a by b, which always has rank 1 for $a, b > 0$. Thus

$$\operatorname{rank}(F \circ \Delta_{i,k,b}) = \operatorname{rank}(F_i)\operatorname{rank}(J_{2^{(\lambda-1)|S_i|},1}) = \operatorname{rank}(F_i) = \sum_{S\subseteq[m]\setminus S_i : \hat{p}_i(S)\neq 0} \lambda^{|S|},$$

by [20]. This part of the proof follows [19, Lemma IV.6].

Then by Lemma 34 and Lemma 38, we have:

$$\widetilde{\mathsf{mSP}}_\kappa(f) \geq \Omega\left(\frac{\sqrt{\kappa}\text{-rank}(F)}{\max_{i,k,b}\operatorname{rank}(F\circ\Delta_{i,k,b})}\right) \geq \Omega\left(\frac{\left(\frac{\varepsilon-\sqrt{\kappa}}{1+\sqrt{\kappa}}\right)^2 \lambda^{\deg_\varepsilon(p)}}{\max_i \sum_{S\subseteq[m]\setminus S_i:\hat{p}_j(S)\neq 0}\lambda^{|S|}}\right). \quad \blacktriangleleft$$

5.2 Monotone Algorithms

In Theorem 32, we showed a non-trivial lower bound on $\log \widetilde{\mathsf{mSP}}(f)$ for some explicit monotone function f. Unlike lower bounds on $\log \widetilde{\mathsf{SP}}(f)$, this does not give us a lower bound on the quantum space complexity of f, however, at the very least it gives us a lower bound on the quantum space complexity of a certain type of quantum algorithm. Of course, this is naturally the case, since a lower bound on $\widetilde{\mathsf{mSP}}(f)$ gives us a lower bound on the quantum space complexity of any algorithm for f that is obtained from a monotone span program. However, this is not the most satisfying characterization, as it is difficult to imagine what this class of algorithms looks like.

In this section, we will consider a more natural class of algorithms whose space complexity is lower bounded by $\widetilde{\mathsf{mSP}}(f)$, and in some cases $\mathsf{mSP}(f)$. We will call a quantum query algorithm a *phase estimation algorithm* if it works by estimating the amplitude on $|0\rangle$ in the phase register after running phase estimation of a unitary that makes one query. We assume that the unitary for which we perform phase estimation is of the form $U\mathcal{O}_x$. This is without loss of generality, because the most general form is a unitary $U_2\mathcal{O}_xU_1$, but we have $(U_2\mathcal{O}_xU_1)^t|\psi_0\rangle = U_1^\dagger(U\mathcal{O}_x)^t|\psi_0'\rangle$ where $|\psi_0'\rangle = U_1|\psi_0\rangle$, and $U = U_1U_2$. The weight on a phase of $|0\rangle$ is not affected by this global (t-independent) $U_1^\dagger$. Thus, we define a phase estimation algorithm as follows:

▶ **Definition 41.** *A* phase estimation algorithm $\mathcal{A} = (U, |\psi_0\rangle, \delta, T, M)$ *for* $f : D \to \{0,1\}$, $D \subseteq \{0,1\}^n$, *is defined by (families of):*

- *a unitary* U *acting on* $\mathcal{H} = \operatorname{span}\{|j, z\rangle : j \in [n], z \in \mathcal{Z}\}$ *for some finite set* $\mathcal{Z}$*;*
- *an initial state* $|\psi_0\rangle \in \mathcal{H}$*;*
- *a bound* $\delta \in [0, 1/2)$*;*
- *positive integers* T *and* $M \leq \frac{1}{\sqrt{\delta}}$*;*

such that for any $M' \geq M$ *and* $T' \geq T$*, the following procedure computes* f *with bounded error:*

1. *Let* $\Phi(x)$ *be the algorithm that runs phase estimation of* $U\mathcal{O}_x$ *on* $|\psi_0\rangle$ *for* T' *steps, and then computes a bit* $|b\rangle_A$ *in a new register* A*, such that* $b = 0$ *if and only if the phase estimate is* 0*.*
2. *Run* M' *steps of amplitude estimation to estimate the amplitude on* $|0\rangle_A$ *after application of* $\Phi(x)$*. Output* 0 *if the amplitude is* $> \delta$*.*

The query complexity *of the algorithm is* $O(MT)$*, and, the space complexity of the algorithm is* $\log\dim\mathcal{H} + \log T + \log M + 1$*.*

We insist that the algorithm work not only for M and T but for any larger integers as well, because we want to ensure that the algorithm is successful because M and T are large enough, and not by some quirk of the particular chosen values. When $\delta = 0$, the algorithm has one-sided error (see Lemma 46).

We remark on the generality of this form of algorithm. Any algorithm can be put into this form by first converting it to a span program, and then compiling that into an algorithm, preserving both the time and space complexity, asymptotically. However, we will consider a special case of this type of algorithm that is *not* fully general.

▶ **Definition 42.** *A* monotone *phase estimation algorithm is a phase estimation algorithm such that if* $\Pi_0(x)$ *denotes the orthogonal projector onto the* $(+1)$*-eigenspace of* $U\mathcal{O}_x$*, then for any* $x \in \{0,1\}^n$*,* $\Pi_0(x)|\psi_0\rangle$ *is in the* $(+1)$*-eigenspace of* $\mathcal{O}_x$*.*

Let us consider what is "monotone" about this definition. The algorithm rejects if $|\psi_0\rangle$ has high overlap with the $(+1)$-eigenspace of $U\mathcal{O}_x$, i.e., $\Pi_0(x)|\psi_0\rangle$ is large. In a monotone phase estimation algorithm, we know that the only contribution to $\Pi_0(x)|\psi_0\rangle$ is in the $(+1)$-eigenspace of $\mathcal{O}_x$, which is exactly the span of $|j,z\rangle$ such that $x_j = 0$. Thus, only 0-queries can contribute to the algorithm rejecting.

As a simple example, Grover's algorithm is a monotone phase estimation algorithm. Specifically, let $|\psi_0\rangle = \frac{1}{\sqrt{n}}\sum_{j=1}^{n}|j\rangle$ and $U = (2|\psi_0\rangle\langle\psi_0| - I)$. Then $U\mathcal{O}_x$ is the standard Grover iterate, and $|\psi_0\rangle$ is in the span of $e^{i\theta}$-eigenvectors of $U\mathcal{O}_x$ with $\sin|\theta| = \sqrt{|x|/n}$, so phase estimation can be used to distinguish the case $|x| = 0$ from $|x| \geq 1$. So $\Pi_0(x)|\psi_0\rangle$ is either 0, when $|x| \neq 0$, or $|\psi_0\rangle$, when $|x| = 0$. In both cases, it is in the $(+1)$-eigenspace of $\mathcal{O}_x$.

It is clear that a monotone phase estimation algorithm can only decide a monotone function. However, while any quantum algorithm can be converted to a phase estimation algorithm, it is not necessarily the case that any quantum algorithm for a monotone function can be turned into a monotone phase estimation algorithm. Thus lower bounds on the quantum space complexity of any monotone phase estimation algorithm for f do not imply lower bounds on $\mathsf{S}_U(f)$. Nevertheless, if we let $\mathsf{mS}_U(f)$ represent the minimum quantum space complexity of any monotone phase estimation algorithm for f, then a lower bound on $\mathsf{mS}_U(f)$ at least tells us that if we want to compute f with space less than said bound, we must use a non-monotone phase estimation algorithm.

Similarly, we let $\mathsf{mS}^1_U(f)$ denote the minimum quantum space complexity of any monotone phase estimation algorithm with $\delta = 0$ that computes f (with one-sided error).

The main theorem of this section states that any monotone phase estimation algorithm for f with space S can be converted to a monotone span program of size $2^{\Theta(S)}$ that approximates f, so that lower bounds on $\mathsf{m}\widetilde{\mathsf{SP}}(f)$ imply lower bounds on $\mathsf{mS}_U(f)$; and that any monotone phase estimation algorithm with $\delta = 0$ and space S can be converted to a monotone span program of size $2^{\Theta(S)}$ that decides f (exactly) so that lower bounds on $\mathsf{mSP}(f)$ imply lower bounds on $\mathsf{mS}^1_U(f)$. These conversions also preserve the query complexity. We now formally state this main result.

▶ **Theorem 43.** *Let* $\mathcal{A} = (U, |\psi_0\rangle, \delta, T, M)$ *be a monotone phase estimation algorithm for* f *with space complexity* $S = \log\dim\mathcal{H} + \log T + \log M + 1$ *and query complexity* $O(TM)$*. Then there is a monotone span program with complexity* $O(TM)$ *and size* $2\dim\mathcal{H} \leq 2^S$ *that approximates* f*. If* $\delta = 0$*, then this span program decides* f *(exactly). Thus*

$$\mathsf{mS}_U(f) \geq \log \mathsf{m}\widetilde{\mathsf{SP}}(f) \quad \textit{and} \quad \mathsf{mS}^1_U(f) \geq \log \mathsf{mSP}(f).$$

We prove this theorem in Section 5.2.1. As a corollary, lower bounds on $\mathsf{mSP}(f)$, such as the one from [14], imply lower bounds on $\mathsf{mS}^1_U(f)$; and lower bounds on $\mathsf{m}\widetilde{\mathsf{SP}}(f)$ such as the one in Theorem 32, imply lower bounds on $\mathsf{mS}_U(f)$. In particular:

▶ **Corollary 44.** *Let* $f : \{0,1\}^n \to \{0,1\}$ *be the function described in Theorem 32. Then* $\mathsf{mS}_U(f) \geq (\log n)^{2-o(1)}$*. Let* $g : \{0,1\}^n \to \{0,1\}$ *be the function described in Theorem 31. Then* $\mathsf{mS}^1_U(g) \geq \Omega(n)$*.*

We emphasize that while this does not give a lower bound on the quantum space complexity of f, or the one-sided quantum space complexity of g, it does show that any algorithm that uses $(\log n)^c$ space to solve f with bounded error, for $c < 2$, or $o(n)$ space to solve g with one-sided error, must be of a different form than that described in Definition 41 and Definition 42.

In a certain sense, monotone phase estimation algorithms completely characterize those that can be derived from monotone span programs, because the algorithm we obtain from compiling a monotone span program is a monotone phase estimation algorithm, as stated below in Lemma 45. However, not all monotone phase estimation algorithms can be obtained by compiling monotone span programs, and similarly, we might hope to show that an even larger class of algorithms can be converted to monotone span programs, in order to give more strength to lower bounds on $\mathsf{mS}_U(f)$.

▶ **Lemma 45.** *Let P be an approximate monotone span program for f with size S and complexity C. Then there is a monotone algorithm for f with query complexity $O(C)$ and space complexity $O(\log S + \log C)$.*

Proof. Fix a monotone span program, and assume it has been appropriately scaled. Without loss of generality, we can let $H_j = H_{j,1} = \text{span}\{|j,z\rangle : z \in \mathcal{Z}_j\}$ for some finite set $\mathcal{Z}_j$. Then, $\mathcal{O}_x = I - 2\Pi_{H(x)}$, which is only true because the span program is monotone. Let $U = 2\Pi_{\text{row}(A)} - I$. Then $U\mathcal{O}_x = (2\Pi_{\ker(A)} - I)(2\Pi_{H(x)} - I)$ is the *span program unitary*, described in [8]. The algorithm obtained from compiling a span program works by performing $O(C)$ steps of phase estimation of this unitary, applied to $|w_0\rangle = A^+|\tau\rangle$, and estimating the amplitude on 0 in the phase register to constant precision (see [8, Lemma 3.6]). This is clearly a phase estimation algorithm for f with query complexity $O(C)$ and space complexity $O(\log S + \log C)$.

The algorithm is a monotone phase estimation algorithm because $U = 2\Pi_{\text{row}(A)} - I$ is a reflection, and $|\psi_0\rangle = |w_0\rangle = A^+|\tau\rangle$ is in the $(+1)$-eigenspace of U, $\text{row}(A)$. Since U is a reflection, the $(+1)$-eigenspace of $U\mathcal{O}_x$ is exactly $(\ker(A) \cap H(x)) \oplus (\text{row}(A) \cap H(x)^\perp)$, and so $\Pi_0(x)|w_0\rangle \in \text{row}(A) \cap H(x)^\perp \subset H(x)^\perp$. ◀

5.2.1 Monotone Algorithms to (Approximate) Monotone Span Programs

In this section, we prove Theorem 43. Throughout this section, we fix a phase estimation algorithm $\mathcal{A} = (U, |\psi_0\rangle, \delta, T, M)$ that computes f, with U acting on $\mathcal{H}$. For any $x \in \{0,1\}^n$ and $\Theta \in [0, \pi]$, we let $\Pi_\Theta(x)$ denote the orthogonal projector onto the span of $e^{i\theta}$-eigenvectors of $U\mathcal{O}_x$ for $|\theta| \leq \Theta$. We will let $\Pi_x = \sum_{j\in[n], z\in\mathcal{Z}: x_j=1} |j,z\rangle\langle j,z|$.

We begin by drawing some conclusions about the necessary relationship between the eigenspaces of $U\mathcal{O}_x$ and a function f whenever a monotone phase estimation computes f. The proofs are somewhat dry and are relegated to Appendix B.

▶ **Lemma 46.** *Fix a phase estimation algorithm with $\delta = 0$ that solves f with bounded error. Then if $f(x) = 0$,*

$$\|\Pi_0(x)|\psi_0\rangle\|^2 \geq \frac{1}{M^2},$$

and for any $d < \sqrt{8}/\pi$, if $f(x) = 1$, then

$$\left\|\Pi_{d\pi/T}(x)|\psi_0\rangle\right\|^2 = 0,$$

and the algorithm always outputs 1, so it has one-sided error.

▶ **Lemma 47.** *Fix a phase estimation algorithm with $\delta \neq 0$ that solves f with bounded error. Then there is some constant $c > 0$ such that if $f(x) = 0$,*

$$\|\Pi_0(x)|\psi_0\rangle\|^2 \geq \max\{\delta(1+c), 1/M^2\}$$

and if $f(x) = 1$, for any $d < \sqrt{8}/\pi$,

$$\left\|\Pi_{d\pi/T}(x)|\psi_0\rangle\right\|^2 \leq \frac{\delta}{1 - \frac{d^2\pi^2}{8}}.$$

To prove Theorem 43, we will define a monotone span program $P_{\mathcal{A}}$ as follows:

$$\begin{aligned} H_{\text{true}} &= \text{span}\{|j,z\rangle : j \in [n], z \in \mathcal{Z}\} = \mathcal{H} \\ H_{j,1} &= H_j = \text{span}\{|j,z,1\rangle : z \in \mathcal{Z}\} \\ A|j,z,1\rangle &= \frac{1}{2}(|j,z\rangle - (-1)^1|j,z\rangle) = |j,z\rangle \\ A|j,z\rangle &= (I - U^\dagger)|j,z\rangle \\ |\tau\rangle &= |\psi_0\rangle. \end{aligned} \tag{2}$$

We first show that $\Pi_0(x)|\psi_0\rangle$ is (up to scaling) a negative witness for x, whenever it is nonzero:

▶ **Lemma 48.** *For any $x \in \{0,1\}^n$, we have*

$$w_-(x) = \frac{1}{\|\Pi_0(x)|\psi_0\rangle\|^2}.$$

In particular, when $\Pi_0(x)|\psi_0\rangle \neq 0$, $\Pi_0(x)|\psi_0\rangle / \|\Pi_0(x)|\psi_0\rangle\|^2$ is an optimal negative witness for x.

Proof. Suppose $\Pi_0(x)|\psi_0\rangle \neq 0$, and let $|\omega\rangle = \Pi_0(x)|\psi_0\rangle / \|\Pi_0(x)|\psi_0\rangle\|^2$. We will first show that this is a negative witness, and then show that no negative witness can have better complexity. First, we notice that

$$\langle\omega|\tau\rangle = \langle\omega|\psi_0\rangle = \frac{\langle\psi_0|\Pi_0(x)|\psi_0\rangle}{\|\Pi_0(x)|\psi_0\rangle\|^2} = 1.$$

Next, we will see that $\langle\omega|A\Pi_{H(x)} = 0$. By the monotone phase estimation property, $\mathcal{O}_x\Pi_0(x)|\psi_0\rangle = \Pi_0(x)|\psi_0\rangle$, and so $\mathcal{O}_x|\omega\rangle = |\omega\rangle$, and thus $\Pi_x|\omega\rangle = 0$, where Π_x is the projector onto $|j,z\rangle$ such that $x_j = 1$. Note that $H(x) = \text{span}\{|j,z,1\rangle : x_j = 1, z \in \mathcal{Z}\} \oplus \text{span}\{|j,z\rangle : j \in [n], z \in \mathcal{Z}\}$. Thus $\Pi_{H(x)} = \Pi_{H_{\text{true}}} + \Pi_x \otimes |1\rangle\langle 1|$. We have:

$$\langle\omega|A(\Pi_x \otimes |1\rangle\langle 1|) = \langle\omega|\Pi_x = 0.$$

Since $|\omega\rangle$ is in the $(+1)$-eigenspace of $U\mathcal{O}_x$, we have $U\mathcal{O}_x|\omega\rangle = |\omega\rangle$ so since $\mathcal{O}_x|\omega\rangle = |\omega\rangle$, $U|\omega\rangle = |\omega\rangle$. Thus

$$\langle\omega|A\Pi_{H_{\text{true}}} = \langle\omega|(I - U^\dagger) \otimes \langle 1| = (\langle\omega| - \langle\omega|) \otimes \langle 1| = 0.$$

Thus $|\omega\rangle$ is a zero-error negative witness for x. Next, we argue that it is optimal.

Suppose $|\omega\rangle$ is any optimal negative witness for x, with size $w_-(x)$. Then since $\langle\omega|\Pi_x = \langle\omega|A(\Pi_x \otimes |1\rangle\langle 1|)$ must be 0, $\mathcal{O}_x|\omega\rangle = (I - 2\Pi_x)|\omega\rangle = |\omega\rangle$, and since $\langle\omega|A\Pi_{H_{\text{true}}} = \langle\omega|(I - U^\dagger)$ must be 0, $U|\omega\rangle = |\omega\rangle$. Thus $|\omega\rangle$ is a 1-eigenvector of $U\mathcal{O}_x$, so

$$\|\Pi_0(x)|\psi_0\rangle\|^2 \geq \left\|\frac{|\omega\rangle\langle\omega|}{\||\omega\rangle\|^2}|\psi_0\rangle\right\|^2 = \frac{|\langle\omega|\psi_0\rangle|^2}{\||\omega\rangle\|^2} = \frac{1}{\||\omega\rangle\|^2}.$$

We complete the proof by noticing that since $\langle\omega|A\Pi_{H_{\text{true}}} = 0$, we have $\langle\omega|A = \langle\omega|\langle 1|$, and $w_-(x) = \|\langle\omega|A\|^2 = \||\omega\rangle\|^2$. ◀

Next we find approximate positive witnesses.

▶ **Lemma 49.** *For any $\Theta \geq 0$, the span program $P_{\mathcal{A}}$ has approximate positive witnesses for any x with error at most $\|\Pi_\Theta(x)|\psi_0\rangle\|^2$ and complexity at most $\frac{5\pi^2}{4\Theta^2}$.*

Proof. We first define a vector $|v\rangle$ by:

$$|v\rangle = (I - (U\mathcal{O}_x)^\dagger)^+(I - \Pi_\Theta(x))|\psi_0\rangle.$$

Note that $I - (U\mathcal{O}_x)^\dagger$ is supported everywhere except the $(+1)$-eigenvectors of $(U\mathcal{O}_x)^\dagger$, which are exactly the $(+1)$-eigenvectors of $U\mathcal{O}_x$. Thus, $(I - \Pi_\Theta(x))|\psi_0\rangle$ is contained in this support.

Next we define $|w\rangle = \left(|\psi_0\rangle - (I - U^\dagger)|v\rangle\right)|1\rangle + |v\rangle$. Then we have:

$$A|w\rangle = |\psi_0\rangle - (I - U^\dagger)|v\rangle + (I - U^\dagger)|v\rangle = |\psi_0\rangle = |\tau\rangle.$$

So $|w\rangle$ is a positive witness, and we next compute its error for x:

$$\begin{aligned}\left\|\Pi_{H(x)^\perp}|w\rangle\right\|^2 &= \left\|\Pi_{\bar{x}}\left(|\psi_0\rangle - (I - U^\dagger)|v\rangle\right)\right\|^2 \\ &= \left\||\psi_0\rangle - \Pi_{\bar{x}}(I - U^\dagger)(I - (U\mathcal{O}_x)^\dagger)^+(I - \Pi_\Theta(x))|\psi_0\rangle\right\|^2.\end{aligned}$$

Above, $\Pi_{\bar{x}} = I - \Pi_x$. We now observe that

$$\Pi_{\bar{x}}(I - \mathcal{O}_x U^\dagger) = \Pi_{\bar{x}}\left(\Pi_{\bar{x}} - (\Pi_{\bar{x}} - \Pi_x)U^\dagger\right) = \Pi_{\bar{x}}(I - U^\dagger).$$

Thus, continuing from above, we have:

$$\begin{aligned}\left\|\Pi_{H(x)^\perp}|w\rangle\right\|^2 &= \left\||\psi_0\rangle - \Pi_{\bar{x}}(I - \mathcal{O}_x U^\dagger)(I - \mathcal{O}_x U^\dagger)^+(I - \Pi_\Theta(x))|\psi_0\rangle\right\|^2 \\ &= \||\psi_0\rangle - \Pi_{\bar{x}}(I - \Pi_\Theta(x))|\psi_0\rangle\|^2 = \|\Pi_{\bar{x}}\Pi_\Theta(x)|\psi_0\rangle\|^2 \leq \|\Pi_\Theta(x)|\psi_0\rangle\|^2.\end{aligned}$$

Now we compute the complexity of $|w\rangle$. First, let $U\mathcal{O}_x = \sum_j e^{i\theta_j}|\lambda_j\rangle\langle\lambda_j|$ be the eigenvalue decomposition of $U\mathcal{O}_x$. Then

$$(I - (U\mathcal{O}_x)^\dagger)^+ = \sum_{j:\theta_j \neq 0} \frac{1}{1 - e^{-i\theta_j}}|\lambda_j\rangle\langle\lambda_j| \quad \text{and} \quad I - \Pi_\Theta(x) = \sum_{j:|\theta_j|>\Theta} |\lambda_j\rangle\langle\lambda_j|.$$

We can thus bound $\||v\rangle\|^2$:

$$\begin{aligned}\||v\rangle\|^2 = \left\|(I - (U\mathcal{O}_x)^\dagger)^+(I - \Pi_\Theta(x))|\psi_0\rangle\right\|^2 &= \left\|\sum_{j:|\theta_j|>\Theta} \frac{1}{1 - e^{-i\theta_j}}\langle\lambda_j|\psi_0\rangle|\lambda_j\rangle\right\|^2 \\ = \sum_{j:|\theta_j|>\Theta} \frac{1}{4\sin^2\frac{\theta_j}{2}}|\langle\lambda_j|\psi_0\rangle|^2 &\leq \frac{\pi^2}{4\Theta^2}.\end{aligned}$$

Next, using $\mathcal{O}_x + 2\Pi_x = I - 2\Pi_x + 2\Pi_x = I$, we compute:

$$\begin{aligned}&\left\||\psi_0\rangle - (I - U^\dagger)|v\rangle\right\|^2 \\ =\ &\left\||\psi_0\rangle - (I - \mathcal{O}_x U^\dagger - 2\Pi_x U^\dagger)(I - \mathcal{O}_x U^\dagger)^+(I - \Pi_\Theta(x))|\psi_0\rangle\right\|^2 \\ =\ &\left\||\psi_0\rangle - (I - \Pi_\Theta(x))|\psi_0\rangle + 2\Pi_x U^\dagger(I - (U\mathcal{O}_x)^\dagger)^+(I - \Pi_\Theta(x))|\psi_0\rangle\right\|^2\end{aligned}$$

$$\begin{aligned}
\ldots &\leq \left(\||\Pi_\Theta(x)|\psi_0\rangle\| + 2 \left\| \Pi_x U^\dagger \sum_{j:|\theta_j|>\Theta} \frac{1}{1-e^{-i\theta_j}} \langle\lambda_j|\psi_0\rangle |\lambda_j\rangle \right\| \right)^2 \\
&\leq \left(\||\Pi_\Theta(x)|\psi_0\rangle\| + 2\sqrt{\sum_{j:|\theta_j|>\Theta} \frac{1}{4\sin^2\frac{\theta_j}{2}} |\langle\lambda_j|\psi_0\rangle|^2} \right)^2 \\
&\leq \left(\||\Pi_\Theta(x)|\psi_0\rangle\| + \frac{\pi}{\Theta} \|(I-\Pi_\Theta(x))|\psi_0\rangle\| \right)^2 \leq \frac{\pi^2}{\Theta^2}.
\end{aligned}$$

Then we have the complexity of $|w\rangle$:

$$\||w\rangle\|^2 = \left\||\psi_0\rangle - (I-U^\dagger)|v\rangle\right\|^2 + \||v\rangle\|^2 \leq \frac{\pi^2}{\Theta^2} + \frac{\pi^2}{4\Theta^2} = \frac{5\pi^2}{4\Theta^2}. \quad \blacktriangleleft$$

We conclude with the following two corollaries, whose combination gives Theorem 43.

▶ **Corollary 50.** *Let $\mathcal{A} = (U, |\psi_0\rangle, 0, T, M)$ be a monotone phase estimation algorithm for f with space complexity $S = \log\dim\mathcal{H} + \log T + \log M + 1$ and query complexity $O(TM)$. Then there is a monotone span program that decides f (exactly) whose size is $2\dim\mathcal{H} \leq 2^S$ and whose complexity is $O(TM)$.*

Proof. If $f(x) = 0$, then by Lemma 46, we have $\|\Pi_0(x)|\psi_0\rangle\|^2 \geq \frac{1}{M^2}$, so by Lemma 48, $w_-(x) \leq M^2$. Thus $W_- \leq M^2$.

If $f(x) = 1$, then by Lemma 46, we have $\left\|\Pi_{2/T}(x)|\psi_0\rangle\right\|^2 = 0$, so by Lemma 49, there's an exact positive witness for x with complexity $O(T^2)$. Thus $W_+ \leq O(T^2)$, and so the span program $P_\mathcal{A}$ from (2) has complexity $O(TM)$. The size of the span program $P_\mathcal{A}$ is $\dim H = 2\dim\mathcal{H}$. $\blacktriangleleft$

▶ **Corollary 51.** *Let $\mathcal{A} = (U, |\psi_0\rangle, \delta, T, M)$ be a monotone phase estimation algorithm for f with space complexity $S = \log\dim\mathcal{H} + \log T + \log M + 1$ and query complexity $O(TM)$. Then there is a constant $\kappa \in (0,1)$ such that there exists a monotone span program that κ-approximates f whose size is $2\dim\mathcal{H} \leq 2^S$ and whose complexity is $O(TM)$.*

Proof. If $f(x) = 0$, then by Lemma 47, we have $\|\Pi_0(x)|\psi_0\rangle\|^2 > \delta(1+c)$ for some constant $c > 0$. Thus, by Lemma 48, $W_- \leq \frac{1}{(1+c)\delta}$.

If $f(x) = 1$, then by Lemma 49, setting $\Theta = d\pi/T$ for $d = \frac{2}{\pi}\sqrt{\frac{c}{1+c}}$, (where c is the constant from above), by Lemma 49 there is an approximate positive witness for x with error $e_x = \left\|\Pi_{2\sqrt{\frac{c}{1+c}}/T}(x)|\psi_0\rangle\right\|^2$ and complexity $O(T^2)$. By Lemma 47, we have

$$e_x \leq \frac{\delta}{1-\frac{d^2\pi^2}{8}} = \frac{\delta}{1-\frac{c}{2(1+c)}} = \frac{\delta(1+c)}{1+c-c/2} \leq \frac{1}{1+c/2}\frac{1}{W_-}.$$

Thus, letting $\kappa = \frac{1}{1+c/2} < 1$, we have that $P_\mathcal{A}$ κ-approximates f. Since the positive witness complexity is $O(T^2)$, and by Lemma 47, we also have $W_- \leq O(M^2)$, the complexity of $P_\mathcal{A}$ is $O(TM)$. The size of $P_\mathcal{A}$ is $\dim H = 2\dim\mathcal{H}$. $\blacktriangleleft$

References

1 S. Aaronson, S. Ben-David, and R. Kothari. Separations in query complexity using cheat sheets. In *Proceedings of the forty-eighth annual ACM Symposium on Theory of Computing (STOC 2016)*, pages 863–876, 2016. arXiv:1511.01937.

2 S. Arora and B. Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009.

3 L. Babai, A. Gál, and A. Wigderson. Superpolynomial Lower Bounds for Monotone Span Programs. *Combinatorica*, 19:301–319, 1999.

4 G. Brassard, P. Høyer, M. Mosca, and A. Tapp. Quantum Amplitude Amplification and Estimation. In S. J. Lomonaca and H. E. Brandt, editors, *Quantum Computation and Quantum Information: A Millennium Volume*, volume 305 of *AMS Contemporary Mathematics Series Millennium Volume*, pages 53–74. AMS, 2002. arXiv:quant-ph/0005055v1.

5 M. Bun and J. Thaler. A Nearly Optimal Lower Bound on the Approximate Degree of AC^0. In *Proceedings of the IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS 2017)*, 2017. arXiv:1703.05784.

6 B. Fefferman and C. Lin. A Complete Characterization of Unitary Quantum Space. In *Proceedings of the 2018 ACM Conference on Innovations in Theoretical Computer Science (ITCS 2018)*, pages 4:1–4:21, 2018. arXiv:1604.01384.

7 A. Gàl. A characterization of span program size and improved lower bounds for monotone span programs. *Computational Complexity*, 10(4):277–296, 2001.

8 T. Ito and S. Jeffery. Approximate Span Programs. *Algorithmica*, 81(6):2158–2195, 2019. arXiv:1507.00432.

9 R. Jozsa, B. Kraus, A. Miyake, and J. Watrous. Matchgate and space-bounded quantum computations are equivalent. *Proceedings of the Royal Society A*, 466(2115), 2009. doi:10.1098/rspa.2009.0433.

10 M. Karchmer and A. Wigderson. On span programs. In *Proceedings of the IEEE 8th Annual Conference on Structure in Complexity Theory*, pages 102–111, 1993.

11 A. Kitaev. Quantum measurements and the Abelian stabilizer problem, 1995. arXiv:quant-ph/9511026.

12 T. Lee, R. Mittal, B. Reichardt, R. Špalek, and M. Szegedy. Quantum Query Complexity of State Conversion. In *Proceedings of the 52nd Annual IEEE Symposium on Foundations of Computer Science (FOCS 2011)*, pages 344–353, 2011.

13 S. V. Lokam. *Complexity Lower Bounds using Linear Algebra*. Now Publishers Inc., Hanover, MA, USA, 2009. doi:10.1561/0400000011.

14 T. Pitassi and R. Robere. Strongly Exponential Lower Bounds for Monotone Computation. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing (STOC 2017)*, pages 1246–1255, 2017.

15 A. A. Razborov. Applications of matrix methods to the theory of lower bounds in computational complexity. *Combinatorica*, 10(1):810093, 1990.

16 A. A. Razborov. On Submodular Complexity Measures. In *Poceedings of the London Mathematical Society symposium on Boolean function complexity*, pages 76–83, 1992.

17 B. Reichardt. Span programs and quantum query complexity: The general adversary bound is nearly tight for every Boolean function. In *Proceedings of the 50th IEEE Symposium on Foundations of Computer Science (FOCS 2009)*, pages 544–551, 2009. arXiv:quant-ph/0904.2759.

18 B. Reichardt and R. Špalek. Span-program-based quantum algorithm for evaluating formulas. *Theory of Computing*, 8(13):291–319, 2012.

19 R. Robere, T. Pitassi, B. Rossman, and S. A. Cook. Exponential Lower Bounds for Monotone Span Programs. In *Proceedings of the 57th IEEE Symposium on Foundations of Computer Science (FOCS 2016)*, pages 406–415, 2016. doi:10.1109/FOCS.2016.51.

20 Alexander A. Sherstov. The Pattern Matrix Method. *SIAM Journal on Computing*, 40(6):1969–2000, 2009.

21 J. Watrous. Space-Bounded Quantum Complexity. *Journal of Computer and System Sciences*, 59(2):281–326, 1999.

A Proof of Claim 18

In this section, we prove Claim 18, restated below:

▷ Claim 18. Let P be a span program that κ-approximates $f : D \to \{0,1\}$ for some constant κ. For any constant $\kappa' \leq \kappa$, there exists a span program P' that κ'-approximates f with $s(P') = (s(P)+2)^{2^{\frac{\log \frac{1}{\kappa'}}{\log \frac{1}{\kappa}}}}$, and $C_{\kappa'}(P', D) \leq O\left(C_\kappa(P, D)\right)$.

Let $|w_0\rangle = A^+|\tau\rangle$. We say a span program is *normalized* if $\||w_0\rangle\| = 1$. A span program can easily be normalized by scaling $|\tau\rangle$, which also scales all positive witnesses and inverse scales all negative witnesses. However, we sometimes want to normalize a span program, while also keeping all negative witness sizes bounded by a constant. We can accomplish this using the following construction, from [8].

▶ **Theorem 52.** *Let $P = (H, V, |\tau\rangle, A)$ be a span program on $\{0,1\}^n$, and let $N = \||w_0\rangle\|^2$. For a positive real number β, define a span program $P^\beta = (H^\beta, V^\beta, |\tau^\beta\rangle, A^\beta)$ as follows, where $|\hat{0}\rangle$ and $|\hat{1}\rangle$ are not in H or V:*

$$H^\beta_{j,b} = H_{j,b}, \quad H^\beta_{\text{true}} = H_{\text{true}} \oplus \text{span}\{|\hat{1}\rangle\}, \quad H^\beta_{\text{false}} = H_{\text{false}} \oplus \text{span}\{|\hat{0}\rangle\}$$

$$V^\beta = V \oplus \text{span}\{|\hat{1}\rangle\}, \quad A^\beta = \beta A + |\tau\rangle\langle\hat{0}| + \frac{\sqrt{\beta^2+N}}{\beta}|\hat{1}\rangle\langle\hat{1}|, \quad |\tau^\beta\rangle = |\tau\rangle + |\hat{1}\rangle.$$

Then we have the following:

- $\left\|(A^\beta)^+|\tau^\beta\rangle\right\| = 1$;
- *for all $x \in P_1$, $w_+(x, P^\beta) = \frac{1}{\beta^2} w_+(x, P) + 2$;*
- *for all $x \in P_0$, $w_-(x, P^\beta) = \beta^2 w_-(x, P) + 1$.*

▶ **Corollary 53.** *Let P be a span program on $\{0,1\}^n$, and P^β be defined as above for $\beta = \frac{1}{\sqrt{W_-(P)}}$. If P κ-approximates f, then P^β $\sqrt{\kappa}$-approximates f, with $W_-(P^\beta) \leq 2$, $\widehat{W}_+(P^\beta) \leq W_-(P)\widehat{W}_+(P) + 2$ and $s(P^\beta) \leq s(P) + 2$.*

Proof. First note that by Theorem 52, $W_-(P^\beta) \leq 2$. Let $|w\rangle$ be an approximate positive witness for x in P, with $\left\|\Pi_{H(x)^\perp}|w\rangle\right\|^2 \leq \frac{\kappa}{W_-(P)}$ and $\||w\rangle\|^2 \leq \widehat{W}_+(P)$. Define

$$|w'\rangle = \frac{1}{\beta(1+\kappa)}|w\rangle + \frac{\beta}{\sqrt{\beta^2+N}}|\hat{1}\rangle + \frac{\kappa}{1+\kappa}|\hat{0}\rangle.$$

One can check that $A^\beta|w'\rangle = |\tau^\beta\rangle$.

$$\begin{aligned}
\left\|\Pi_{H^\beta(x)^\perp}|w'\rangle\right\|^2 &= \frac{1}{\beta^2(1+\kappa)^2}\left\|\Pi_{H(x)^\perp}|w\rangle\right\|^2 + \frac{\kappa^2}{(1+\kappa)^2} \\
&\leq \frac{1}{\beta^2(1+\kappa)^2}\frac{\kappa}{W_-(P)} + \frac{\kappa^2}{(1+\kappa)^2} \\
&= \frac{\kappa+\kappa^2}{(1+\kappa)^2} \leq \frac{2\kappa(1+\kappa)}{W_-(P^\beta)(1+\kappa)^2} = \frac{1}{W_-(P^\beta)}\frac{2\kappa}{1+\kappa} \leq \frac{\sqrt{\kappa}}{W_-(P^\beta)},
\end{aligned}$$

where we have used $W_-(P^\beta) \leq 2$. We upper bound $\widehat{W}_+(P^\beta)$ by noting that:

$$\||w'\rangle\|^2 \leq \frac{1}{\beta^2(1+\kappa)^2}\widehat{W}_+(P) + \frac{\beta^2}{\beta^2+N} + \frac{\kappa^2}{(1+\kappa)^2} \leq W_-(P)\widehat{W}_+(P) + 2.$$

Finally, $s(P^\beta) = s(P) + 2$ because of the two extra degrees of freedom $|\hat{0}\rangle$ and $|\hat{1}\rangle$. ◀

Proof of Claim 18. We will first show how, given a span program P such that $\||w_0\rangle\|^2 \leq 1$, and P κ-approximates f, we can get a span program P' such that $\||w_0'\rangle\|^2 \leq 1$, $W_-(P') \leq W_-(P)^2$, P' κ^2-approximates f, $\widehat{W}_+(P') \leq 4\widehat{W}_+(P)$, and $s(P') = s(P)^2$.

Define P' as follows, where S is a *swap* operator, which acts as $S(|u\rangle|v\rangle) = |v\rangle|u\rangle$ for all $|u\rangle, |v\rangle \in H$:

$$H'_{j,b} = H_{j,b} \otimes H, \qquad A' = (A \otimes A)\left(\frac{I_{H\otimes H} + S}{2}\right), \qquad |\tau'\rangle = |\tau\rangle|\tau\rangle.$$

Observe that for any $|u\rangle, |v\rangle \in H$, we have

$$A'(|u\rangle|v\rangle - |v\rangle|u\rangle) = 0, \quad \text{and} \quad A'|u\rangle|u\rangle = A|u\rangle \otimes A|u\rangle.$$

Note that $A'(|w_0\rangle|w_0\rangle) = |\tau'\rangle$, so $\left\|A'^{+}|\tau'\rangle\right\| \leq \||w_0\rangle|w_0\rangle\| \leq 1$.

If $\langle\omega|$ is a negative witness for x in P, it is easily verified that $\langle\omega'| = \langle\omega| \otimes \langle\omega|$ is a negative witness in P', and

$$\|\langle\omega'|A'\|^2 = \left\|\frac{1}{2}(\langle\omega|A) \otimes (\langle\omega|A) + \frac{1}{2}(\langle\omega|A) \otimes (\langle\omega|A)\right\|^2 = \|\langle\omega|A\|^4,$$

so $w_-(x, P') \leq w_-(x, P)^2$, and $W_-(P') \leq W_-(P)^2$.

If $|w\rangle$ is an approximate positive witness for x in P, then define

$$|w'\rangle = |w\rangle|w\rangle - \Pi_{H(x)^\perp}|w\rangle\Pi_{H(x)}|w\rangle + \Pi_{H(x)}|w\rangle\Pi_{H(x)^\perp}|w\rangle - \Pi_{H(x)}|w\rangle\Pi_{\ker(A)}|w\rangle.$$

We have

$$\begin{aligned} A'|w'\rangle &= A|w\rangle A|w\rangle - \frac{1}{2}\left(A\Pi_{H(x)}|w\rangle \otimes A\Pi_{\ker(A)}|w\rangle + A\Pi_{\ker(A)}|w\rangle \otimes A\Pi_{H(x)}|w\rangle\right) \\ &= |\tau\rangle|\tau\rangle = |\tau'\rangle. \end{aligned}$$

We can bound the error as:

$$\begin{aligned} \left\|\Pi_{H'(x)^\perp}|w'\rangle\right\|^2 &= \left\|(\Pi_{H(x)^\perp} \otimes I)|w'\rangle\right\|^2 = \left\|\Pi_{H(x)^\perp}|w\rangle|w\rangle - \Pi_{H(x)^\perp}|w\rangle\Pi_{H(x)}|w\rangle\right\|^2 \\ &= \left\|\Pi_{H(x)^\perp}|w\rangle\Pi_{H(x)^\perp}|w\rangle\right\|^2 \leq \frac{\kappa^2}{W_-(P)^2} \leq \frac{\kappa^2}{W_-(P')}. \end{aligned}$$

Next, observe that

$$\begin{aligned} &(\Pi_{H(x)} + \Pi_{H(x)^\perp}) \otimes (\Pi_{H(x)} + \Pi_{H(x)^\perp}) - \Pi_{H(x)^\perp} \otimes \Pi_{H(x)} + \Pi_{H(x)} \otimes \Pi_{H(x)^\perp} \\ &= \Pi_{H(x)} \otimes \Pi_{H(x)} + \Pi_{H(x)} \otimes \Pi_{H(x)^\perp} + \Pi_{H(x)^\perp} \otimes \Pi_{H(x)^\perp} + \Pi_{H(x)} \otimes \Pi_{H(x)^\perp} \\ &= \Pi_{H(x)} \otimes I + I \otimes \Pi_{H(x)^\perp} \end{aligned}$$

so $|w'\rangle = \Pi_{H(x)}|w\rangle \otimes |w\rangle + |w\rangle \otimes \Pi_{H(x)^\perp}|w\rangle - \Pi_{H(x)}|w\rangle \otimes \Pi_{\ker(A)}|w\rangle$.

Thus, using the assumption $\||w_0\rangle\| \leq 1$, and the fact that $\Pi_{\text{row}(A)}|w\rangle = |w_0\rangle$:

$$\begin{aligned} \||w'\rangle\|^2 &= \left\|\Pi_{H(x)}|w\rangle|w\rangle + |w\rangle\Pi_{H(x)^\perp}|w\rangle - \Pi_{H(x)}|w\rangle\Pi_{\ker(A)}|w\rangle\right\|^2 \\ &= \left\|\Pi_{H(x)}|w\rangle\Pi_{\text{row}(A)}|w\rangle + |w\rangle\Pi_{H(x)^\perp}|w\rangle\right\|^2 \\ &= \left\|\Pi_{H(x)}|w\rangle|w_0\rangle\right\|^2 + \left\||w\rangle\Pi_{H(x)^\perp}|w\rangle\right\|^2 + 2\left\|\Pi_{H(x)}|w\rangle\right\|^2 \langle w_0|\Pi_{H(x)^\perp}|w\rangle \\ &\leq \widehat{W}_+(P) + \widehat{W}_+(P)\frac{\kappa}{W_-(P)} + 2\widehat{W}_+(P)\sqrt{\frac{\kappa}{W_-(P)}} \leq (1 + \kappa + 2\sqrt{\kappa})\widehat{W}_+(P). \end{aligned}$$

Note that we could assume that $\widehat{W}_-(P) \geq 1$ because $\|w_0\| \leq 1$.

We complete the proof by extending to the general case. Let P be any span program that κ-approximates f. By applying Theorem 52 and Corollary 53, we can get a span program, P_0, with $\||w_0\rangle\| = 1$, $W_-(P_0) \leq 2$, $\widehat{W}_+(P_0) \leq C(P)^2 + 2$, and $s(P_0) = s(P) + 2$, that $\sqrt{\kappa}$-approximates f. We can then apply the construction described above, iteratively, d times, to get a span program P_d that $\sqrt{\kappa}^{2^d} = \kappa^{2^{d-1}}$-approximates f, with

$$s(P_d) = s(P_0)^{2^d} = (s(P)+2)^{2^d},$$

$$W_-(P_d) \leq 2^{2^d}, \qquad \text{and} \qquad \widehat{W}_+(P_d) \leq 4^d \widehat{W}_+(P_0) \leq 4^d C(P)^2 + 2 \cdot 4^d.$$

Setting $d = \log\left(\frac{\log \frac{1}{\kappa'}}{\log \frac{1}{\kappa}}\right) + 1$ gives the desired κ'. ◁

B Proofs of Lemma 46 and Lemma 47

We will prove the lemmas as a collection of claims. Fix $T' \geq T$ and $M' \geq M$ with which to run the algorithm. Suppose $\Phi(x)$ outputs $|\psi(x)\rangle = \sqrt{p_x}|0\rangle_A|\Phi_0(x)\rangle + \sqrt{1-p_x}|1\rangle_A|\Phi_1(x)\rangle$, and let $\tilde{p}$ denote the estimate output by the algorithm. We will let $U\mathcal{O}_x = \sum_j e^{i\sigma_j(x)}|\lambda_j^x\rangle\langle\lambda_j^x|$ be an eigenvalue decomposition.

▷ Claim 54. If $f(x) = 0$ then $\|\Pi_0(x)|\psi_0\rangle\|^2 \geq \frac{1}{M^2}$.

Proof. Since the algorithm computes f with bounded error, the probability of accepting x is at most $1/3$, so $\tilde{p} \leq \delta$ with probability at most $1/3$.

Amplitude estimation is just phase estimation of a unitary W_Φ such that $|\psi(x)\rangle$ is in the span of $e^{\pm 2i\theta_x}$-eigenvectors of W_Φ, where $p_x = \sin^2\theta_x$, $\theta_x \in [0, \pi/2)$ [4]. One can show that the probability of outputting an estimate $\tilde{p} = 0$ is $\sin^2(M'\theta_x)/(M'^2\sin^2(\theta_x))$, so

$$\frac{1}{3} \geq \frac{\sin^2(M'\theta_x)}{M'^2\sin^2(\theta_x)}.$$

If $M'\theta_x \leq \frac{\pi}{2}$, then this would give $\frac{1}{3} \geq \frac{4}{\pi^2}$, which is a contradiction. Thus, we have:

$$M'\theta_x > \frac{\pi}{2} \quad \Rightarrow \quad \frac{2\theta_x}{\pi} > \frac{1}{M'} \quad \Rightarrow \quad \sin\theta_x > \frac{1}{M'} \quad \Rightarrow \quad \sqrt{p_x} > \frac{1}{M'}.$$

Since $\Phi(x)$ is the result of running phase estimation, we have

$$p_x = \sum_j |\langle\lambda_j^x|\psi_0\rangle|^2 \frac{\sin^2(T'\sigma_j(x)/2)}{T'^2\sin^2(\sigma_j(x)/2)} \leq \|\Pi_\Theta(x)|\psi_0\rangle\|^2 + \frac{\pi^2}{T'^2\Theta^2},$$

for any Θ. In particular, if Δ is less than the spectral gap of $U\mathcal{O}_x$, we have $\|\Pi_\Delta(x)|\psi_0\rangle\| = \|\Pi_0(x)|\psi_0\rangle\|$, so

$$\frac{1}{M'^2} < \|\Pi_0(x)|\psi_0\rangle\|^2 + \frac{\pi^2}{T'^2\Delta^2}.$$

This is true for any choices $T' \geq T$ and $M' \geq M$, so we must have:

$$\frac{1}{M^2} \leq \|\Pi_0(x)|\psi_0\rangle\|^2. \qquad \triangleleft$$

▷ Claim 55. If $f(x) = 1$ and $\delta = 0$, then for any $d < \frac{\sqrt{8}}{\pi}$, $\left\|\Pi_{d\pi/T}(x)|\psi_0\rangle\right\|^2 = 0$.

Proof. Suppose towards a contradiction that $\left\|\Pi_{d\pi/T}(x)|\psi_0\rangle\right\|^2 > 0$. Then $p_x > 0$, and some sufficiently large $M' \geq M$ would detect this and cause the algorithm to output 0, so we must actually have $\left\|\Pi_{d\pi/T}(x)|\psi_0\rangle\right\|^2 = 0$. In fact, in order to sure that no large enough value M' detects amplitude > 0 on $|0\rangle_A$, we must have $p_x = 0$ whenever $f(x) = 1$. That means that when $f(x) = 1$, the algorithm never outputs 0, so the algorithm has one-sided error. ◁

▷ Claim 56. There is some constant c such that if $f(x) = 0$ and $\delta > 0$ then $\|\Pi_0(x)|\psi_0\rangle\|^2 > \delta(1 + c)$.

Proof. Recall that $\tilde{p} \in \{\sin^2(\pi m/M') : m = 0, \ldots, M' - 1\}$. We will restrict our attention to choices M' such that for some integer d,

$$\sin^2 \frac{d\pi}{M'} \leq \delta < \sin^2 \frac{(d + 1/3)\pi}{M'}.$$

To see that such a choice exists, let τ be such that $\delta = \sin^2 \tau$, and note that the condition holds as long as $d \leq \frac{\tau M'}{\pi} < d + 1/3$ for some d, which is equivalent to saying that $\lfloor \frac{3\tau M'}{\pi} \rfloor = 0 \mod 3$. If $K = \lfloor \frac{1}{2} \frac{\pi}{3\tau} \rfloor$, then for any $M' \geq M$, and $\ell \geq 0$, define $M_\ell = M' + \ell K$. Then for any $\ell > 0$,

$$\frac{3\tau}{\pi} M_\ell - \frac{3\tau}{\pi} M_{\ell-1} = \frac{3\tau}{\pi} K \in \left[\frac{1}{2} - \frac{3\tau}{\pi}, \frac{1}{2}\right],$$

so there must be one $\ell \in \{0, \ldots, 6\}$ such that $\lfloor \frac{3\tau}{\pi} M_\ell \rceil = 0 \mod 3$. In particular, there is some choice M_ℓ satisfying the condition such that (using some $M' \leq \frac{1}{\sqrt{\delta}}$):

$$\sqrt{\delta} M_\ell \leq \sqrt{\delta} \left(\frac{1}{\sqrt{\delta}} + 6\frac{\pi}{6\tau}\right) = 1 + \frac{\pi \sin \tau}{\tau} \leq 1 + \pi. \tag{3}$$

We will use this value as our M' for the remainder of this proof.

Let $p_x = \sin^2 \theta_x$ for $\theta_x \in [0, \pi/2]$. Let z be an integer such that $\Delta = \theta_x - \pi z/M'$ has $|\Delta| \leq \frac{\pi}{2M'}$. Then the outcome $\tilde{p} = \sin^2 \frac{\pi z}{M'}$ has probability:

$$\frac{1}{M'^2} \left| \sum_{t=0}^{M'-1} e^{i2t(\theta_x - \pi z/M')} \right|^2 = \frac{1}{M'^2} \left| \sum_{t=0}^{M'-1} e^{i2t\Delta} \right|^2 = \frac{\sin^2(M'\Delta)}{M'^2 \sin^2 \Delta} \geq \frac{4}{\pi^2},$$

since $|M'\Delta| \leq \frac{\pi}{2}$. Thus, by correctness, we must have $\sin^2(\pi z/M') > \delta \geq \sin^2 \frac{d\pi}{M'}$. Thus $z > d$, so

$$\frac{(d+1)\pi}{M'} \leq \frac{z\pi}{M'} = \theta_x - \Delta \leq \theta_x + \frac{\pi}{2M'}.$$

Thus:

$$\frac{(d + 1/3)\pi}{M'} + \frac{2\pi}{3M'} \leq \theta_x + \frac{\pi}{2M'}$$

$$\sin\left(\frac{(d + 1/3)\pi}{M'} + \frac{\pi}{6M'}\right) \leq \sin \theta_x$$

$$\sin\left(\frac{(d + 1/3)\pi}{M'}\right) \cos \frac{\pi}{6M'} + \cos\left(\frac{(d + 1/3)\pi}{M'}\right) \sin \frac{\pi}{6M'} \leq \sqrt{p_x}$$

$$\sqrt{\delta}\sqrt{1 - \sin^2 \frac{\pi}{6M'}} + \sqrt{1 - \delta} \sin \frac{\pi}{6M'} \leq \sqrt{p_x}$$

When $\sin^2 \frac{\pi}{6M'} \leq 1 - \delta$, which we can assume, the above expression is minimized when $\sin^2 \frac{\pi}{6M'}$ is as small as possible. We have, using $M' \leq \frac{1+\pi}{\sqrt{\delta}}$, from (3):

$$\sin^2 \frac{\pi}{6M'} \geq \frac{4}{36M'^2} \geq \frac{\delta}{9(1+\pi)^2}.$$

Thus, continuing from above, letting $k = \frac{1}{9(1+\pi)^2}$, we have:

$$\sqrt{\delta}\sqrt{1-k\delta} + \sqrt{1-\delta}\sqrt{k\delta} \leq \sqrt{p_x}$$
$$\delta(1-k\delta) + (1-\delta)k\delta + 2\delta\sqrt{k(1-\delta)(1-k\delta)} \leq p_x$$

Next, notice that $(1-k\delta)(1-\delta)$ is minimized when $\delta = \frac{1+k}{2k}$, but $\delta \leq \frac{1}{2} < \frac{1+k}{2k}$, so we have, using $k < 1$ and $\delta \leq 1/2$:

$$\delta(1 + k(1-2\delta) + 2\sqrt{k}\sqrt{(1-k/2)(1-1/2)}) \leq p_x$$
$$\delta(1 + 0 + \sqrt{k}) \leq p_x.$$

Since $\Phi(x)$ is the result of running phase estimation of $U\mathcal{O}_x$ for $T' \geq T$ steps, we have:

$$p_x = \sum_j |\langle \lambda_j^x | \psi_0 \rangle|^2 \frac{\sin^2(\frac{T'\sigma_j(x)}{2})}{(T')^2 \sin^2(\frac{\sigma_j(x)}{2})},$$

so in particular, for any $\Theta \in [0, \pi)$, we have

$$p_x \leq \|\Pi_\Theta(x)|\psi_0\rangle\|^2 + \sum_{j:|\sigma_j(x)|>\Theta} |\langle \lambda_j^x | \psi_0 \rangle|^2 \frac{1}{(T')^2 \sin^2(\frac{\Theta}{2})}.$$
$$\leq \|\Pi_\Theta(x)|\psi_0\rangle\|^2 + \|(I - \Pi_\Theta(x))|\psi_0\rangle\|^2 \frac{\pi^2}{(T')^2\Theta^2}.$$

In particular, for any $\Theta < \Delta$ where Δ is the spectral gap of $U\mathcal{O}_x$, we have $\|\Pi_\Theta(x)|\psi_0\rangle\| = \|\Pi_0(x)|\psi_0\rangle\|$, so for any $T' \geq T$, we have

$$\|\Pi_0(x)|\psi_0\rangle\|^2 + \frac{\pi^2}{(T')^2\Delta^2} \geq p_x \geq \delta(1+\sqrt{k}).$$

Since this holds for any $T' \geq T$, we get $\|\Pi_0(x)|\psi_0\rangle\|^2 \geq \delta(1+\sqrt{k})$. The proof is completed by letting $c = \sqrt{k}$. ◁

▷ Claim 57. If $f(x) = 1$ and $\delta > 0$ then $\left\|\Pi_{d\pi/T}(x)|\psi_0\rangle\right\|^2 (1 - d^2\pi^2/8) \leq \delta$.

Proof. If $|\lambda\rangle$ is an $e^{i\theta}$-eigenvector of $U\mathcal{O}_x$ for some $|\theta| \leq d\pi/T < \sqrt{8}/T$, then the probability of measuring 0 in the phase register upon performing T steps of phase estimation is:

$$p_x(\theta) := \frac{1}{T^2}\left|\sum_{t=0}^{T-1} e^{it\theta}\right|^2 = \frac{\sin^2 \frac{T\theta}{2}}{T^2 \sin^2 \frac{\theta}{2}}.$$

Let $\varepsilon(x) = 1 - \frac{\sin^2 x}{x^2}$ for any x. It is simple to verify that $\varepsilon(x) \leq x^2/2$ for any x, and $\varepsilon(x) \in [0,1]$ for any x. So we have:

$$p_x(\theta) \geq \frac{(T\theta/2)^2(1-\varepsilon(T\theta/2))}{T^2(\theta/2)^2(1-\varepsilon(\theta/2))} \geq 1 - \varepsilon(T\theta/2) \geq 1 - \frac{T^2\theta^2}{8}.$$

Thus, we conclude that

$$p_x \geq \left\|\Pi_{d\pi/T}(x)|\psi_0\rangle\right\|^2 \left(1 - \frac{T^2}{8}\frac{d^2\pi^2}{T^2}\right) = \left\|\Pi_{d\pi/T}(x)|\psi_0\rangle\right\|^2 \left(1 - \frac{d^2\pi^2}{8}\right).$$

If this is $> \delta$, then with some sufficiently large $M' \geq M$, amplitude estimation would detect this and cause the algorithm to output 0 with high probability. ◁

DEEP-FRI: Sampling Outside the Box Improves Soundness

Eli Ben-Sasson
StarkWare Industries Ltd., Netanya, Israel
eli@starkware.co

Lior Goldberg
StarkWare Industries Ltd., Netanya, Israel
lior@starkware.co

Swastik Kopparty
StarkWare Industries Ltd., Netanya, Israel
swastik.kopparty@gmail.com

Shubhangi Saraf
Department of Mathematics and Department of Computer Science, Rutgers University
shubhangi.saraf@gmail.com

Abstract

Motivated by the quest for scalable and succinct zero knowledge arguments, we revisit worst-case-to-average-case reductions for linear spaces, raised by [Rothblum, Vadhan, Wigderson, STOC 2013]. The previous state of the art by [Ben-Sasson, Kopparty, Saraf, CCC 2018] showed that if some member of an affine space U is δ-far in relative Hamming distance from a linear code V – this is the worst-case assumption – then most elements of U are almost-δ-far from V – this is the average case. However, this result was known to hold only below the "double Johnson" function of the relative distance δ_V of the code V, i.e., only when $\delta < 1 - (1 - \delta_V)^{1/4}$.

First, we increase the soundness-bound to the "one-and-a-half Johnson" function of δ_V and show that the average distance of U from V is nearly δ for any worst-case distance δ smaller than $1 - (1 - \delta_V)^{1/3}$. This bound is tight, which is somewhat surprising because the one-and-a-half Johnson function is unfamiliar in the literature on error correcting codes.

To improve soundness further for Reed Solomon codes we sample outside the box. We suggest a new protocol in which the verifier samples a single point z outside the box D on which codewords are evaluated, and asks the prover for the value at z of the interpolating polynomial of a random element of U. Intuitively, the answer provided by the prover "forces" it to choose one codeword from a list of "pretenders" that are close to U. We call this technique Domain Extending for Eliminating Pretenders (DEEP).

The DEEP method improves the soundness of the worst-case-to-average-case reduction for RS codes up their list decoding radius. This radius is bounded from below by the Johnson bound, implying average distance is approximately δ for all $\delta < 1 - (1 - \delta_V)^{1/2}$. Under a plausible conjecture about the list decoding radius of Reed-Solomon codes, average distance from V is approximately δ for all δ. The DEEP technique can be generalized to all linear codes, giving improved reductions for capacity-achieving list-decodable codes.

Finally, we use the DEEP technique to devise two new protocols:

- An Interactive Oracle Proof of Proximity (IOPP) for RS codes, called DEEP-FRI. The soundness of the protocol improves upon that of the FRI protocol of [Ben-Sasson et al., ICALP 2018] while retaining linear arithmetic proving complexity and logarithmic verifier arithmetic complexity.
- An Interactive Oracle Proof (IOP) for the Algebraic Linking IOP (ALI) protocol used to construct zero knowledge scalable transparent arguments of knowledge (ZK-STARKs) in [Ben-Sasson et al., eprint 2018]. The new protocol, called DEEP-ALI, improves soundness of this crucial step from a small constant $< 1/8$ to a constant arbitrarily close to 1.

2012 ACM Subject Classification Theory of computation → Interactive proof systems; Mathematics of computing → Coding theory

Keywords and phrases Interactive Oracle Proofs of Proximity, STARK, Low Degree Testing

11th Innovations in Theoretical Computer Science Conference (ITCS 2020).
Editor: Thomas Vidick; Article No. 5; pp. 5:1–5:32

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.5

Funding *Swastik Kopparty*: Research supported in part by NSF grants CCF-1253886, CCF-1540634, CCF-1814409 and CCF-1412958, and BSF grant 2014359. Some of this research was done while visiting the Institute for Advanced Study.
Shubhangi Saraf: Research supported in part by NSF grants CCF-1350572, CCF-1540634 and CCF-1412958, BSF grant 2014359, a Sloan research fellowship and the Simons Collaboration on Algorithms and Geometry. Some of this research was done while visiting the Institute for Advanced Study.

Acknowledgements We thank Rishabh Bhadauria for pointing out an omission in a previous version of the paper.

1 Introduction

Arithmetization is a marvelous technique that can be used to reduce problems in computational complexity, like verifying membership in a nondeterministic language, to questions about membership of vectors in algebraic codes like Reed-Solomon (RS) and Reed-Muller (RM) codes [17, 15]. One of the end-points of such a reduction is the RS proximity testing (RPT) problem. It is a problem of inherent theoretical interest, but also of significant practical importance because it is used in recent constructions of succinct zero knowledge (ZK) arguments including Ligero [1], Aurora [5], and Scalable Transparent ARguments of Knowledge (ZK-STARKs) [2]. We discuss this connection after describing the problem and our results.

In the RPT problem a verifier is given oracle access to a function $f : D \to \mathbb{F}$, we call $D \subset \mathbb{F}$ the *evaluation domain*, and is tasked with distinguishing between the "good" case that f is a polynomial of degree at most d and the "bad" case in which f is δ-far in relative Hamming distance from all degree-d polynomials. To achieve succinct verification time, poly-logarithmic in d, we must allow the verifier some form of interaction with a prover – the party claiming that $\deg(f) \leq d$. Initially, this interaction took the form of oracle access to a probabilistically checkable proof of proximity (PCPP) [7] provided by the prover in addition to f. Indeed, in this model the RPT problem can be "solved" with PCPPs of quasilinear size $|D|\mathsf{poly}\log|D|$, constant query complexity and constant soundness [11, 13]. However, the concrete complexity of prover time, verifier time and communication complexity are rather large, even when considering practical settings that involve moderately small instance sizes.

To improve prover, verifier, and communication complexity for concrete (non-asymptotic) size problems, the Interactive Oracle Proofs of Proximity (IOPP) model is more suitable [18, 6, 4]. This model can be viewed as a multi-round PCPP. Instead of having the prover write down a single proof π, in the IOPP setting the proof oracle is produced over a number of rounds of interaction, during which the verifier sends random bits and the prover responds with additional (long) messages to which the verifier is allowed oracle access. The additional rounds of interaction allow for a dramatic improvement in the asymptotic and concrete complexity of solving the RPT problem. In particular, the Fast RS IOPP (FRI) protocol of [3] has linear prover arithmetic complexity, logarithmic verifier arithmetic complexity and constant soundness. Our results regarding the RPT problem improve the soundness of this protocol and offer a better protocol in terms of soundness in the high-error regime (also known as the "list decoding" regime).

Soundness analysis of FRI reduces to the following natural "worst-case-to-average-case" question regarding linear spaces, which is also independently very interesting for the case of general (non-RS) codes. This question was originally raised in a different setting by [20] and we start by discussing it for general linear codes before focusing on the special, RS code, case.

1.1 Maximum distance vs average distance to a linear code

Recall that the relative Hamming distance between $u, v \in \mathbb{F}^D$, denoted $\Delta(u, v)$, is the fraction of entries $i \in D$ such that $u_i \neq v_i$, and the relative Hamming distance between u and a set $V \subset \mathbb{F}^D$ is $\Delta(u, U) = \min_{v \in V} \Delta(u, v)$. Suppose that $U \subset \mathbb{F}^D$ is a "line", a 1-dimensional[1] affine space over $\mathbb{F}$. Let $u^* \in \mathbb{F}^D$ denote the origin of this line and u be its slope, so that $U = \{u_x = u^* + xu \mid x \in \mathbb{F}\}$. For a fixed linear space $V \subset \mathbb{F}^D$, pick u^* to be the element in U that is farthest from V, denoting by $\delta_{\max}$ its relative Hamming distance (from V). This is our worst-case assumption. Letting $\delta_x = \Delta(u_x, V)$, what can be said about the *expected* distance $\mathbf{E}_{x \in \mathbb{F}}[\delta_x]$ of u_x from V?

Rothblum, Vadhan and Wigderson showed that $\mathbf{E}_x[\delta_x] \geq \frac{\delta_{\max}}{2} - o(1)$ for all spaces U and V, where, here and below, $o(1)$ denotes negligible terms that approach 0 as $|\mathbb{F}| \to \infty$ [20]. A subset of the co-authors of this paper [9] improved this to $\mathbf{E}[\delta_x] \geq 1 - \sqrt{1 - \delta_{\max}} - o(1)$, showing the average distance scales roughly like the Johnson list-decoding function of $\delta_{\max}$, where $J(x) := 1 - \sqrt{1 - x}$. In both of these bounds the expected distance is strictly smaller than $\delta_{\max}$. However, the latter paper also showed that when V is a (linear) error correcting code with large relative distance δ_V, if $\delta_{\max}$ is smaller than the "double Johnson" function of δ_V, given by $J^{(2)}(x) := J(J(x))$, then the average distance hardly deteriorates,

$$\mathbf{E}[\delta_x] \geq \min\left(\delta_{\max}, J^{(2)}(\delta_V)\right) - o(1) = \min\left(\delta_{\max}, 1 - \sqrt[4]{1 - \delta_V}\right) - o(1) \tag{1}$$

and the equation above summarizes the previous state of affairs on this matter.

Our first result is an improvement of Equation (1) to the "one-and-a-half-Johnson" function $J^{(1.5)}(x) = 1 - (1 - x)^{1/3}$. Lemma 1 says that for codes V of relative Hamming distance δ_V,

$$\mathbf{E}[\delta_x] \geq \min\left(\delta_{\max}, J^{(1.5)}(\delta_V)\right) - o(1) = \min\left(\delta_{\max}, 1 - \sqrt[3]{1 - \delta_V}\right) - o(1). \tag{2}$$

Our second result shows that Equation (2) is tight, even for the special case of V being an RS code. We find this result somewhat surprising because the $J^{(1.5)}(x)$ function is not known to be related to any meaningful coding theoretic notion. The counter-example showing the tightness of Equation (2) arises for very special cases, in which (i) $\mathbb{F}$ is a binary field (of characteristic 2), (ii) the rate ρ is precisely $1/8 = 2^{-3}$ and, most importantly, (iii) the evaluation domain D equals all of $\mathbb{F}$.Roughly speaking, the counter-example uses functions $u^*, u : \mathbb{F}_{2^n} \to \mathbb{F}_{2^n}$ that are $3/4 = 1 - \rho^{2/3}$-far from polynomials of degree $\rho 2^n$ yet *pretend* to be low-degree because for all $x \in \mathbb{F}_{2^n} \setminus \{0\}$ the function $u^* + xu$ is $1/2 = \sqrt[3]{\rho}$-close to a polynomial of degree $\rho 2^n$. See Lemma 22 for details.

Our next set of results, which we discuss below, show how to go beyond the above limitation through a new interactive proximity proving technique.

1.2 Domain Extension for Eliminating Pretenders (DEEP)

The case that interests us most is when V is an RS code (although we will return to the discussion of general linear codes later). Henceforth, the RS code of rate ρ evaluated over D is

$$\mathsf{RS}[\mathbb{F}, D, \rho] := \{f : D \to \mathbb{F} \mid \deg(f) < \rho|D|\}.$$

RS codes are maximum distance separable (MDS), meaning that $\delta_V = 1 - \rho$ and so Equation (2) simplifies to

$$\mathbf{E}[\delta_x] \geq \min(\delta_{\max}, 1 - \sqrt[3]{\rho}) - o(1). \tag{3}$$

[1] The generalization of our results to spaces U of dimension > 1 is straightforward by partitioning U into lines through u^* and applying these results to each line.

This improved bound can be translated, using some extra work, to FRI soundness analysis with similar guarantees. Specifically, Equation (3) implies that for $f : D \to \mathbb{F}$ that is δ-far from $\mathsf{RS}[\mathbb{F}, D, \rho]$, the soundness error of a single invocation of the FRI QUERY test (which requires $\log |D|$ queries) is at most $\max\{1-\delta, \sqrt[3]{\rho}\}$, and this can be plugged into ZK-STARKs like [2] and ZK-SNARGs like Aurora [5]. Roughly speaking, if the rejection probability is of δ-far words is $\max(\delta, \delta_0)$ then to reach soundness error less than $2^{-\lambda}$ for codes of blocklength n, communication complexity (and verifier complexity) scale roughly like $\frac{\lambda}{\log \delta_0} \cdot c \cdot \log n$ for some constant c. Thus, the improvement from Equation (1) to Equation (2) translates to a 25% reduction in verifier complexity (from $\frac{4\lambda}{\log \rho} \cdot c \cdot \log n$ to $\frac{3\lambda}{\log \rho} \cdot c \cdot \log n$).

To break the soundness bound of Equation (2) and thereby further reduce verifier complexity in the afore-mentioned systems, we suggest a new method. We discuss it first for RS codes, then generalize to arbitrary linear codes. If $u^*, u : D \to \mathbb{F}$ are indeed the evaluation of two degree d polynomials, say, P^* and P, our verifier will artificially *extend the domain* D to a larger one $\bar{D}$, sample uniformly $z \in \bar{D}$ and ask for the evaluation of $P^*(z)$ and $P(z)$. The answers provided by the prover can now be applied to modify each of u^* and u in a *local* manner to reflect the new knowledge, and along the way also prune down the large list of polynomials which u^* and u might pretend to be. If $\alpha_z^* = P^*(z), \alpha_z = P(z)$ are the honest prover's answers to the query z, then $(X - z)$ divides $P^*(X) - \alpha_z^*$ and likewise $(X - z)|P(X) - \alpha_z$. Letting $\alpha_x = \alpha^* + x\alpha$ and $P_x(X) = P^*(X) + xP(X)$ it follows that $(X - z)|P_x(X) - \alpha_x$. Consider now the soundness of this procedure. In the extreme case that u^* has a small list of polynomials that, each, somewhat agree with it, then with high probability over z, any answer provided by the prover will agree with at most one of the polynomials in this list. The proof of our main technical result, Theorem 3, formalizes this intuition. For radius δ, let L_δ^* be the maximal list size,

$$L_\delta^* = \max_{u^* \in \mathbb{F}^D} |\{v \in V \mid \Delta(u^*, v) < \delta\}|$$

where Δ denotes relative Hamming distance. Let $V|_{u_x(z)=\alpha_x}$ be the restriction of V to codewords that are evaluations of polynomials of degree at most d that, additionally, evaluate to α_x on z. Our main Theorem 3 shows that if $\Delta(u^*, V) = \delta_{\max}$ then for any pair of answers α_z^*, α_z given in response to query z,

$$\mathbf{E}_{z,x}\left[\Delta(u_x, V|_{u_x(z)=\alpha_x})\right] \geq \delta_{\max} - L_\delta^* \cdot \left(\frac{\rho|D|}{|\bar{D}|}\right)^{1/3} - o(1). \tag{4}$$

The Johnson bound (Theorem 20) says that when $\delta < J(1-\rho) = 1 - \sqrt{\rho}$ we have $L_\delta^* = O(1)$ and this improves the worst-case-to-average-case result from that of Equation (2) to a bound that matches the Johnson bound:

$$\mathbf{E}_{z,x}[\Delta(u_x, V|_{u_x(z)=\alpha_x})|] \geq \min(\delta_{\max}, J(\delta_V)) - o(1) = \min(\delta_{\max}, \sqrt{\rho}) - o(1). \tag{5}$$

The exact behavior of the list size of Reed-Solomon codes beyond the Johnson bound is a famous open problem. It may be the case that the list size is small for radii far greater than the Johnson bound; in fact, for most domains D this is roughly known to hold [21]. If it holds that that list sizes are small all the way up to radius equal to the distance $\delta_V = 1 - \rho$ (i.e., if Reed-Solomon codes meet list-decoding capacity), then Equation (5) implies that the technique suggested here has optimal soundness for (nearly) all distance parameters.

1.3 DEEP-FRI as an application of domain extended sampling

Applying the technique of domain extension for eliminating pretenders to the FRI protocol requires a modification that we discuss next, after recalling that protocol.

Fast RS IOPP (FRI)

The FRI protocol can be described as a process of "randomly folding" an inverse Fast Fourier Transform (iFFT). The classical iFFT receives as input a function $f : \langle\omega\rangle \to \mathbb{F}$ where ω generates a multiplicative group of order 2^k for integer k. The iFFT computes (in arithmetic complexity $O(k2^k)$) the interpolating polynomial $\tilde{f}(X)$ of the function f. This computation follows by computing (in linear time) a pair of functions $f_0, f_1 : \langle\omega^2\rangle \to \mathbb{F}$, recalling $|\langle\omega^2\rangle| = \frac{1}{2}|\langle\omega\rangle|$. Their interpolants $\tilde{f}_0, \tilde{f}_1$ are then used to compute in linear time the original interpolant $\tilde{f}$ of f.

The FRI protocol is an IOPP, which means that a prover interacts with a verifier by submitting oracles in response to randomness received by the verifier (as explained earlier in this section). The prover's goal in the FRI protocol is to convince the verifier that the function f, to which the verifier has oracle access, is a member of the RS code evaluated over $\langle\omega\rangle$ and of rate ρ. Thus, the protocol starts with the prover committing to f as above. The verifier now samples a random $x^{(0)} \in \mathbb{F}$ and sends it to the prover, completing the first round of interaction. In the second round, the prover commits to a function $f^{(1)} : \langle\omega^2\rangle \to \mathbb{F}$ that is supposedly the $x^{(0)}$-linear combination of f_0, f_1 used in the iFFT above; namely, $f^{(1)}$ should be equal to $f_0 + x^{(0)} f_1$. It turns out that if f is indeed of degree less than $\rho|\langle\omega\rangle|$ then for all $x^{(0)}$ sampled by the verifier, the second function $f^{(1)}$ is of degree less than $\rho|\langle\omega^2\rangle|$ as well. The tricky part is showing that when f is δ-far from $\mathsf{RS}[\mathbb{F}, \langle\omega\rangle, \rho]$ this also holds with high probability (over x) for $f^{(1)}$ and some δ' that is as close as possible to δ. (One can show that invariably we have $\delta' \leq \delta$, i.e., the green line of Figure 1 is an upper bound on soundness of both FRI and the new DEEP-FRI protocol described below.)

The worst-case-to-average-case results of Equation (2) and Lemma 1 can be converted to similar improvements for FRI, showing that for $\delta < 1 - \sqrt[3]{\rho}$ we have $\delta' \approx \delta$. This follows directly from the techniques of [9, Section 7] (see the red line in Figure 1). But to use the new DEEP technique of Equation (4) and Theorem 3 in order to improve soundness of an RS-IOPP, we need to modify the FRI protocol, leading to a new protocol that is aptly called DEEP-FRI. Instead of constructing $f^{(1)}$ directly, our verifier first samples $z^{(0)} \in \mathbb{F}$ and queries the prover for the evaluation of the interpolant of $f^{(0)}$ on $z^{(0)}$ and $-z^{(0)}$. After the answers $\alpha_{z^{(i)}}, \alpha_{-z^{(i)}}$ have been recorded, the verifier proceeds by sampling $x^{(0)}$ and expects the prover to submit $f^{(1)}$, which is the linear combination of f'_0, f'_1 derived from the modification f' of f that takes into account the answers $\alpha_{z^{(i)}}, \alpha_{-z^{(i)}}$. Assuming $\tilde{f}$ is the interpolant of f, an honest prover would set $f'(X) := (\tilde{f}(X) - U(X))/(Z(X))$ where $U(X)$ is the degree ≤ 1 polynomial that evaluates to $\alpha_{z^{(0)}}$ on $z^{(0)}$ and to $\alpha_{-z^{(0)}}$ on $-z^{(0)}$ and $Z(X)$ is the monic degree 2 polynomial whose roots are $z^{(0)}$ and $-z^{(0)}$. As shown in Section 4, the soundness bounds of Equation (4) and Theorem 3 now apply to DEEP-FRI. This shows that the soundness of DEEP-FRI, i.e., the rejection probability of words that are δ-far from $\mathsf{RS}[\mathbb{F}, D, \rho]$) is roughly δ for any δ that is smaller than the maximal radius for which list-sizes are "small". Figure 1 summarizes the results described here.

Practical Implications to reducing RPT query complexity

A number of recent implementations of zero knowledge (ZK) argument systems rely on RPT protocols [1, 2, 5]. These systems are notable for their (i) lack of reliance on number theoretic cryptographic assumptions (like hardness of the discrete log problem) and their (ii) *transparency* – all verifier messages are public random coins (so-called Arthur-Merlin

protocols)[2]. The FRI and DEEP-FRI protocols are efficient RPT solutions and, in fact, the FRI protocol is already deployed in code in the Github repositories libSTARK and libIOP. To construct a ZK system with sufficiently small soundness error, smaller than $2^{-\lambda}$, one needs to bound the soundness error of the RPT used in it. Viewed from this angle, the previous analysis of FRI by [9] proved that $\frac{\lambda\rho}{4}$ queries to the FRI protocol suffice to reduce soundness error below $2^{-\lambda}$. Lemma 2 lowers this number to $\frac{\lambda\rho}{3}$ for FRI. Theorem 8 shows that a number $\frac{\lambda\rho}{2}$ of queries for the same target soundness is enough for DEEP-FRI, and assuming Conjecture 21 the number of queries reduces to $\lambda\rho$, which is optimal as it matches the soundness error upper bound for FRI and DEEP-FRI.

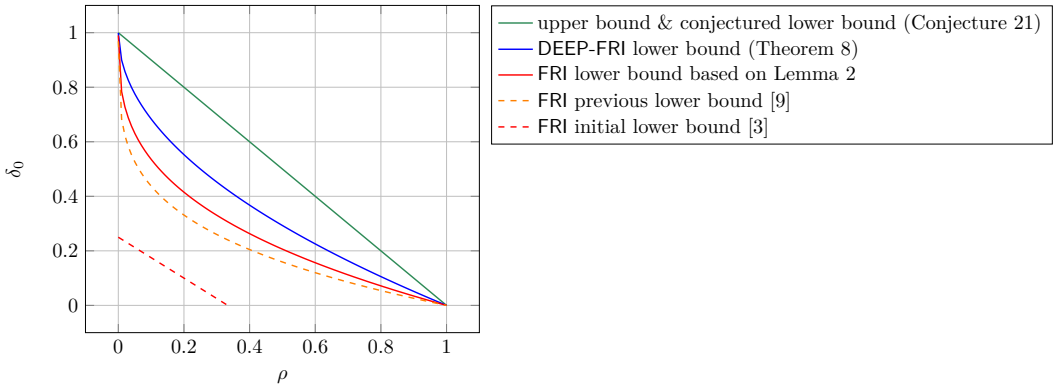


Figure 1 FRI and DEEP-FRI soundness threshold δ_0 as a function of RS code rate ρ, for a single invocation of the QUERY phase, as field size $q \to \infty$. $\delta_0(\rho)$ is defined to be the largest distance parameter δ for which soundness (rejection probaiblity) of a single invocation of the FRI/DEEP-FRI QUERY is $\delta - o(1)$. Higher lines are better. The top line is the trivial upper bound on soundness which applies to both FRI and DEEP-FRI; the bottom line is the soundness of the original analysis of [3]. Dashed lines represent prior results. The red line is the (tight) soundness lower bound for FRI and the blue line is a lower bound on DEEP-FRI soundness. Under a plausible conjecture for Reed-Solomon list-decodability (Conjecture 21), the actual soundness is as high as the green line.

Organization of the rest of the paper

Section 2 gives an improved worst-to-average case reductions for general spaces. Section 3 presents our main technical result, showing that the DEEP method improves worst-to-average case reductions for RS codes up to the Johnson bound (provably) and perhaps even beyond. Details of the two main applications of the DEEP method follow: the DEEP-FRI protocol in Section 4 obtains better soundness than the state of the art FRI protocol, and the DEEP-ALI protocol which improves soundness in ZK-STARK systems is explained in Section 5.

2 Improved High-error Distance Preservation

Our first result gives better distance preservation results for linear codes V of relative distance λ. The previous state-of-the-art [9] said that when a 1-dimensional affine space U contains some element u^* that is $\delta_{\max} = \Delta(u^*, V)$ far from V, then

$$\mathbf{E}_{u \in U}[\Delta(u, V)] \geq \min(\delta_{\max}, 1 - J^{(2)}(\lambda)) - o(1).$$

[2] By contrast, the ZK-SNARK used, e.g., in the Zcash cryptocurrency, relies on a setup phase involving non-transparent randomness (and revealing the random coins that are used to create the system would compromise its security).

The following lemma improves the average-case distance to

$$\mathbf{E}_{u\in U}[\Delta(u,V)] \geq \min(\delta_{\max}, 1 - J^{(1.5)}(\lambda)) - o(1).$$

Later on, in Appendix B, we will show that this result is tight (for a sub-family of RS codes).

▶ **Lemma 1** (One-and-half Johnson distance preservation). *Let $V \subseteq \mathbb{F}_q^n$ be a linear code of distance $\lambda = \Delta(V) \triangleq \min\{\Delta(v,v') \mid v \neq v', v, v' \in V\}$. Let $\epsilon, \delta > 0$ with $\epsilon < 1/3$ and $\delta < 1 - (1 - \lambda + \epsilon)^{1/3}$.*

Suppose $u^* \in \mathbb{F}_q^n$ is such that $\Delta(u^*, V) > \delta + \epsilon$. Then for all $u \in \mathbb{F}_q^n$, there are at most $2/\epsilon^2$ values of $x \in \mathbb{F}_q$ such that $\Delta(u^* + xu, V) < \delta$.

This result is the contra-positive statement of the following, more informative, version of it, that we prove below.

▶ **Lemma 2** (One-and-half Johnson distance preservation – positive form). *Let $V \subseteq \mathbb{F}_q^D$ be a linear code of distance $\lambda = \Delta(V)$. Let $\epsilon, \delta > 0$ with $\epsilon < 1/3$ and $\delta < 1 - (1 - \lambda + \epsilon)^{1/3}$. Let $u, u^* \in \mathbb{F}_q^D$ satisfy*

$$\Pr_{x\in\mathbb{F}_q}[\Delta(u^* + xu, V) < \delta] \geq \frac{2}{\epsilon^2 q}. \tag{6}$$

Then there exist $v, v^* \in V$ and $C \subseteq D$ such that the following three statements hold simultaneously:

- $|C| \geq (1 - \delta - \epsilon)|D|$,
- $u|_C = v|_C$, *and*
- $u^*|_C = v^*|_C$.

Observe that if u, u^* satisfy Equation (6) then the v, v^*, C deduced by Lemma 2 have the property that for *all* $x \in \mathbb{F}_q$, we have $\Delta(u^* + xu, v^* + xv) \leq \delta + \epsilon$. In other words, the existence of v, v^* and C almost completely explains Equation (6).

Quantitatively weaker statements in this vein were proved by [16, 3] in the low-error case, and by [12, 9] in the high-error case. The proofs of the latter two results used combinatorial tools (the Kővári-Sós-Turán bound and the Johnson bound respectively) that are closely related to one another. Our improved proof below is direct, and is based on the same convexity principle that underlies both the Kővári-Sós-Turán and Johnson bounds.

Proof. Let $u_x = u^* + xu$. Let $A = \{x \mid \Delta(u^* + xu, V) < \delta\}$. For each $x \in A$, let $v_x \in V$ be an element of V that is closest to u_x, and let $S_x \subseteq D$ be the agreement set of u_x and v_x, defined as $S_x = \{y \in D \mid u_x(y) = v_x(y)\}$.

For x, β, γ picked uniformly from A and y picked uniformly from D, we have:

$$\begin{aligned}\mathbf{E}_{x,\beta,\gamma}[|S_x \cap S_\beta \cap S_\gamma|/|D|] &= \mathbf{E}_{y,x,\beta,\gamma}[1_{y\in S_x\cap S_\beta\cap S_\gamma}] = \mathbf{E}_y[\mathbf{E}_x[1_{y\in S_x}]^3] \\ &\geq \mathbf{E}_{y,x}[1_{y\in S_x}]^3 \geq (1-\delta)^3 > 1 - \lambda + \epsilon.\end{aligned}$$

The second equality above follows from the independence of the events $y \in S_x, y \in S_\beta, y \in S_\gamma$ given $y \in D$. The first inequality is Jensen's and the last inequality is by assumption on $\delta, \lambda, \epsilon$.

Thus $\Pr_{x,\beta,\gamma}[|S_x \cap S_\beta \cap S_\gamma| \geq (1 - \lambda)|D|] \geq \epsilon$. Note that

$$\Pr_{x,\beta,\gamma}[x, \beta, \gamma \text{ are not all distinct}] < 3/|A|.$$

Since $|A| \geq 2/\epsilon^2 > \frac{6}{\epsilon}$ (recall $\epsilon < 1/3$), we have that $3/|A| \leq \epsilon/2$ and hence x, β, γ are all distinct with probability at least $1 - \epsilon/2$. Thus with probability at least $\epsilon/2$ over the choice of x, β, γ, we have that x, β, γ are all distinct and $|S_x \cap S_\beta \cap S_\gamma| > (1 - \lambda)|D|$. This means

that there are distinct x_0, β_0 such that $\Pr_\gamma[|S_{x_0} \cap S_{\beta_0} \cap S_\gamma| > (1-\lambda)|D|] \geq \epsilon/2$. Fix a γ where this happens. Let $S = S_{x_0} \cap S_{\beta_0} \cap S_\gamma$. We have that $(x_0, u_{x_0}), (\beta_0, u_{\beta_0}), (\gamma, u_\gamma)$ are collinear. Thus $(x_0, u_{x_0}|_S), (\beta_0, u_{\beta_0}|_S), (\gamma, u_\gamma|_S)$ are all collinear. By definition of S, we get that: $(x_0, v_{x_0}|_S), (\beta_0, v_{\beta_0}|_S), (\gamma, v_\gamma|_S)$ are all collinear. Since $|S| > (1-\lambda)|D|$ (and recalling that λ is the distance of V), we get that v_γ is determined by $v_\gamma|_S$ via a linear map. This means that $(x_0, v_{x_0}), (\beta_0, v_{\beta_0}), (\gamma, v_\gamma)$ are all collinear.

Thus $\epsilon/2$-fraction of the $\gamma \in A$ have the "good" property that (γ, v_γ) is on the line passing through (x_0, v_{x_0}) and (β_0, v_{β_0}). Write this line as $v^* + xv$ and notice that for all "good" γ we have $v_\gamma = v^* + \gamma v$. Let $A' \subseteq A$ denote the set of good elements for this line, recording that $|A'| \geq |A| \cdot \epsilon/2 \geq 1/\epsilon$.

Thus for $x \in A'$, $\Delta(u^* + xu, v^* + xv) < \delta$. Consider the set $C \subset D$ that is defined by $C = \{y \in D \mid u^*(y) = v^*(y) \text{ AND } u(y) = v(y)\}$. For each $y \in D \setminus C$ there exists at most a single value of $x \in \mathbb{F}_q$ satisfying $u^*(y) + x \cdot u(y) = v^*(y) + x \cdot v(y)$ because $(u^*(y) - v^*(y)) + x \cdot (u(y) - v(y))$ has at most one value x on which it vanishes.

This implies

$$\delta \geq \mathbf{E}_{x \in A'}[\Delta_D(u_x, v_x)] \geq \frac{|D \setminus C|}{|D|} \cdot \left(1 - \frac{1}{|A'|}\right) \geq \left(1 - \frac{|C|}{|D|}\right) \cdot (1 - \epsilon) \geq 1 - \frac{|C|}{|D|} - \epsilon.$$

Rearranging, we get $\frac{|C|}{|D|} \geq 1 - (\delta + \epsilon)$ and this completes the proof. ◀

3 The DEEP Theorem – Using Domain Extension for Eliminating Pretenders (DEEP) and Improving Soundness

We now come to the statement of our improved-soundness distance preservation result. We describe it first for the special case of RS codes. A weighted variant of the theorem is shown in Appendix C.1 because it is used later in the DEEP-FRI protocol (Section 4). We end with Appendix C.2 in which we present a general version of the folowing result, that applies to all linear codes.

3.1 DEEP Theorem for RS codes

The vectors u^*, u discussed in the previous section are now viewed as functions $u^*, u : D \to \mathbb{F}_q$ and we are interested in the distance of a random linear combination $u_x = u^* + x \cdot u$ from the code $V = \mathsf{RS}[\mathbb{F}_q, D, \rho]$, where $x \in \mathbb{F}_q$ is sampled uniformly. Lemma 1 established that if $\max(\Delta(u^*, V), \Delta(u, V)) = \delta_{\max}$, then with high probability (over x), the function u_x will have distance at least $\approx \min(\delta_{\max}, 1 - \rho^{1/3})$ from V.

Lemma 2 roughly gets used in the following way in the FRI protocol. There are two functions $u^*, u : D \to \mathbb{F}_q$ and there is a prover who claims that both are evaluations of low degree polynomials. In order to verify this, the verifier uniformly samples $x \in \mathbb{F}_q$ and considers the function $u_x = u^* + x \cdot u$. Lemma 2 shows that if any of u^*, u is far from being evaluations of a low degree polynomial, then so is $u^* + x \cdot u$. This then gets exploited in the FRI protocol using FFT type ideas.

We now precede the random process of sampling $x \in \mathbb{F}_q$ with a step of *domain extension*, explained next. Assume a prover claims that both u and u^* are evaluations of low degree polynomials (say $P(Y)$ and $P^*(Y)$). So these polynomials can be evaluated also outside of D. Based on this, a verifier first samples $z \in \mathbb{F}_q$ uniformly and asks the prover to reply with two field elements $a^*, a \in \mathbb{F}_q$ which are supposedly equal to $P^*(z), P(z)$, respectively. After receiving these answers, the verifier proceeds as before, sampling uniformly $x \in \mathbb{F}_q$. Then,

setting $b = a^* + x \cdot a$, we examine the distance of u_x from the sub-code $V_{z,b} \subset V$ comprised of all members of V whose interpolating polynomial evaluates to b on input z. The code $V_{z,b}$ is the additive coset (shifted by b) of a low-degree ideal, the ideal generated by $(X - z)$ (cf. Lemma 6).

Using the Johnson Bound (Theorem 20) we prove that with high probability u_x is at least $\approx \min(\delta_{\max}, 1 - \rho^{1/2})$ far from $V_{z,b}$. Assuming RS codes have a larger list-decoding radius (Conjecture 21), we show that with high probability u_x is $\approx \delta_{\max}$-far from $V_{z,b}$ for nearly all values of $\delta_{\max}$. Later, in Section 4, we shall use the improved distance preservation to construct the DEEP-FRI protocol for testing proximity to the RS code with improved soundness.

The statement we give below is given more generally in terms of the list size bound $\mathcal{L}(\mathbb{F}_q, D, d = \rho|D|, \delta)$; we instantiate it later with the Johnson bound and with Conjecture 21. It is useful to keep in mind that this will be used in a setting where q is much larger than $|D|$ (and hence d), and where L^*_δ is small.

▶ **Theorem 3** (DEEP method for RS codes). *Let $\rho > 0$ and let $V = \mathsf{RS}[\mathbb{F}_q, D, \rho]$. For $z, b \in \mathbb{F}_q$, we let*

$$V_{z,b} = \{Q(Y)|_D \in V \mid Q(z) = b\}.$$

For $\delta > 0$ let $L^*_\delta = \mathcal{L}(\mathbb{F}_q, D, d = \rho|D|, \delta)$.

Let $u, u^* \in \mathbb{F}_q^D$. For each $z \in \mathbb{F}_q$, let $B_z(X) \in \mathbb{F}_q[X]$ be an arbitrary linear function. Suppose that for some $1/3 > \epsilon > 0$ the following holds,

$$\Pr_{x,z\in\mathbb{F}_q}[\Delta(u^* + xu, V_{z,B_z(x)}) < \delta] \geq \max\left(2L^*_\delta\left(\frac{d}{q} + \epsilon\right)^{1/3}, \frac{4}{\epsilon^2 q}\right). \tag{7}$$

Then there exist $v, v^* \in V$ and $C \subset D$ such that: (i) $|C| \geq (1 - \delta - \epsilon)|D|$, (ii) $u|_C = v|_C$, and (iii) $u^*|_C = v^*|_C$. Consequently, we have $\Delta(u, V), \Delta(u^*, V) \leq \delta + \epsilon$.

Proof. To simplify notation set $\eta = \max\left(2L^*_\delta\left(\frac{d}{q} + \epsilon\right)^{1/3}, \frac{4}{\epsilon^2 q}\right)$, and let $u_x = u^* + xu$. Let $\mathcal{E}[x, z]$ denote the event "$\exists P(Y) \in \mathsf{List}(u_x, V, \delta), P(z) = B_z(x)$". The assumption of Equation (7) now reads as $\Pr_{x,z\in\mathbb{F}_q}[\mathcal{E}[x, z]] \geq \eta$. Thus we get, $\Pr_x[\Pr_z[\mathcal{E}[x, z]] \geq \eta/2] \geq \eta/2$. Let $A = \{x \in \mathbb{F}_q \mid \Pr_z[\mathcal{E}[x, z]] \geq \eta/2]\}$ and notice $|A| \geq \eta q/2$.

For $x \in \mathbb{F}_q$, pick $P_x \in V$ to be a member $P \in \mathsf{List}(u_x, V, \delta)$ that maximizes

$$\Pr_{z\in\mathbb{F}_q}[P(z) = B_z(x)].$$

Let $S_x = \{z \in \mathbb{F}_q \mid P_x(z) = B_z(x)\}$ and set $\mu_x = |S_x|/q$. By definition, $|\mathsf{List}(u_x, V, \delta)| \leq L^*_\delta$, and so by the pigeonhole principle, for each $x \in A$ we have $\mu_x \geq \frac{\eta}{2L^*_\delta}$.

For x, β, γ picked uniformly from A, and z picked uniformly from $\mathbb{F}_q$, we have:

$$\begin{aligned}\mathbf{E}_{x,\beta,\gamma}[|S_x \cap S_\beta \cap S_\gamma|/q] &= \mathbf{E}_{z,x,\beta,\gamma}[1_{z\in S_x\cap S_\beta\cap S_\gamma}] = \mathbf{E}_z[\mathbf{E}_x[1_{z\in S_x}]^3] \\ &\geq \mathbf{E}_{z,x}[1_{z\in S_x}]^3 \geq \left(\frac{\eta}{2L^*_\delta}\right)^3 > \frac{d}{q} + \epsilon.\end{aligned}$$

The second equality above follows from the independence of x, β, γ. The first inequality is an application of Jensen's inequality and the last inequality is by assumption on η. Thus $\Pr_{x,\beta,\gamma}[|S_x \cap S_\beta \cap S_\gamma| > d] \geq \epsilon$. Note that $\Pr_{x,\beta,\gamma}[x, \beta, \gamma$ are not all distinct$] < 3/|A|$. Since $|A| \geq \eta q/2 \geq 2/\epsilon^2 \geq 6/\epsilon$ we have $3/|A| \leq \epsilon/2$. Thus $\Pr_{x,\beta,\gamma}[x, \beta, \gamma$ are all distinct and $|S_x \cap S_\beta \cap S_\gamma| > d] \geq \epsilon/2$.

This means that there are distinct x_0, β_0 such that $\Pr_\gamma[|S_{x_0} \cap S_{\beta_0} \cap S_\gamma| > d] \geq \epsilon/2$. Consider some γ where this happens. Let $S = S_{x_0} \cap S_{\beta_0} \cap S_\gamma$. By construction we know that for all $z \in \mathbb{F}_q$, $(x_0, B_z(x_0)), (\beta_0, B_z(\beta_0)), (\gamma, B_z(\gamma))$ are collinear. So, in particular, for $z \in S$ this holds.

By definition of S, we get that for each $z \in S$, $(x_0, P_{x_0}(z)), (\beta_0, P_{\beta_0}(z)), (\gamma, P_\gamma(z)) \in \mathbb{F}_q \times \mathbb{F}_q$ are collinear. Since $|S| > d$, we have that P_γ is uniquely determined by $P_\gamma|_S$ by a linear map. This allows us to conclude that $(x_0, P_{x_0}), (\beta_0, P_{\beta_0}), (\gamma, P_\gamma) \in \mathbb{F}_q \times \mathbb{F}_q[Y]$ are collinear in the $\mathbb{F}_q$-vector space $\mathbb{F}_q \times \mathbb{F}_q[Y]$.

Thus, an $\epsilon/2$-fraction of the $\gamma \in A$ have the "good" property that (γ, P_γ) is on the line passing through (x_0, P_{x_0}) and (β_0, P_{β_0}). Write this line as $\{P^* + xP \mid x \in \mathbb{F}\}$ and notice that for all "good" γ we have $P_\gamma = P^* + \gamma P$. Let $A' \subseteq A$ denote the set of good elements for this line, recording that $|A'| \geq |A| \cdot \epsilon/2 \geq 1/\epsilon$. By definition of $\mathsf{List}(u_x, V, \delta)$ and the assumption $P_x \in \mathsf{List}(u_x, V, \delta)$, we have that $\Delta(u_x, P_x) < \delta$ for $x \in A'$.

Consider the set $C \subset D$ defined by $C = \{y \in D \mid u^*(y) = P^*(y) \text{ AND } u(y) = P(y)\}$. For each $y \in D \setminus C$ there exists at most a single value of $x \in \mathbb{F}_q$ satisfying $u_x(y) = P_x(y)$ because $u_x(y) - P_x(y) = (u^*(y) - P^*(y)) + x \cdot (u(y) - P(y))$ has at most one value x on which it vanishes. This implies

$$\delta \geq \max_{x \in A'}\{\Delta_D(u_x, P_x)\} \geq \frac{|D \setminus C|}{|D|} \cdot \left(1 - \frac{1}{|A'|}\right) \geq \left(1 - \frac{|C|}{|D|}\right) \cdot (1 - \epsilon) \geq 1 - \frac{|C|}{|D|} - \epsilon.$$

Rearranging, we get $\frac{|C|}{|D|} \geq 1 - (\delta + \epsilon)$. Taking $v = P$ and $v^* = P^*$ completes the proof. ◀

4 First Application – DEEP-FRI

In this section we describe the new fast RS IOPP, called DEEP-FRI. We start by recalling the FRI protocol from [3], describing it nearly verbatim as in [10, Section 7].

4.1 FRI

Our starting point is a function $f^{(0)} : L^{(0)} \to \mathbb{F}$ where $\mathbb{F}$ is a finite field, the evaluation domain $L^{(0)} \subset \mathbb{F}$ is a coset of a group[3] contained in $\mathbb{F}$, and $|L^{(0)}| = 2^{k^{(0)}}$. We assume the target rate is $\rho = 2^{-\mathcal{R}}$ for some positive integer $\mathcal{R}$. The FRI protocol is a two-phase protocol (the two phases are called COMMIT and QUERY) that convinces a verifier that $f^{(0)}$ is close to the Reed-Solomon code $\mathsf{RS}[\mathbb{F}, L^{(0)}, \rho]$.

The COMMIT phase of the FRI protocol involves $\mathsf{r} = k^{(0)} - \mathcal{R}$ rounds. Before any communication, the prover and verifier agree on a sequence of (cosets of) sub-groups $L^{(i)}$, where $|L^{(i)}| = 2^{k^{(0)}-i}$. Let $\mathsf{RS}^{(i)}$ denote the Reed-Solomon code $\mathsf{RS}[\mathbb{F}, L^{(i)}, \rho|L^{(i)}|]$.

The main ingredient of the FRI protocol is a special algebraic hash function H_x, which takes a seed $x \in \mathbb{F}$, and given as input a function $f : L^{(i)} \to \mathbb{F}$, it produces as output a hash whose length is $1/2$ as long as f. More concretely, $H_x[f]$ is a function

$$H_x[f] : L^{(i+1)} \to \mathbb{F}$$

with the following properties:

1. **locality:** For any $s \in L^{(i+1)}$, $H_x[f](s)$ can be computed by querying f at just two points in its domain (these two points are $(q^{(i)})^{-1}(s)$).

[3] The group can be additive, in which case $\mathbb{F}$ is a binary field, or multiplicative, in which case it is not.

2. **completeness:** If $f \in \mathsf{RS}^{(i)}$, then for all $x \in \mathbb{F}$, we have that $H_x[f] \in \mathsf{RS}^{(i+1)}$.
3. **soundness:** If f is far from $\mathsf{RS}^{(i)}$, then with high probability over the choice of seed x, $H_x[f]$ is quite far from $\mathsf{RS}^{(i+1)}$.

These last two properties roughly show that for random x, H_x preserves distance to Reed-Solomon codes. For the precise description of H_x see Appendix E and [9].

The high-level idea of the FRI protocol can then be described as follows. First we are in the COMMIT phase of the protocol. The verifier picks a random $x^{(0)} \in \mathbb{F}$ and asks the prover to write down the hash $H_{x^{(0)}}[f^{(0)}] : L^{(1)} \to \mathbb{F}$. By Properties 2 and 3 above, our original problem of estimating the distance of $f^{(0)}$ to $\mathsf{RS}^{(0)}$ reduces to estimating the distance of $H_{x^{(0)}}[f^{(0)}]$ to $\mathsf{RS}^{(1)}$ (which is a problem of $1/2$ the size). This process is then repeated: the verifier picks a random $x^{(1)} \in \mathbb{F}$ and asks the prover to write down $H_{x^{(1)}}[H_{x^{(0)}}[f^{(0)}]]$, and so on. After r rounds of this, we are reduced to a constant sized problem which can be solved in a trivial manner. However, the verifier cannot blindly trust that the functions $f^{(1)}, \ldots$ that were written down by the prover truly are obtained by repeatedly hashing $f^{(0)}$. This has to be checked, and the verifier does this in the QUERY phase of the protocol, using Property 1 above.

We describe the phases of the protocol below.

COMMIT **Phase:**

1. For $i = 0$ to $\mathsf{r} - 1$:
   - **a.** The verifier picks uniformly random $x^{(i)} \in \mathbb{F}$ and sends it to the prover.
   - **b.** The prover writes down a function $f^{(i+1)} : L^{(i+1)} \to \mathbb{F}$. (In the case of an honest prover, $f^{(i+1)} = H_{x^{(i)}}[f^{(i)}]$.)
2. The prover writes down a value $C \in \mathbb{F}_q$. (In the case of an honest prover, $f^{(\mathsf{r})}$ is the constant function with value $= C$).

QUERY **Phase:** (executed by the Verifier)

1. Repeat ℓ times:
   - **a.** Pick $s^{(0)} \in L^{(0)}$ uniformly at random.
   - **b.** For $i = 0$ to $\mathsf{r} - 1$:
     - **i.** Define $s^{(i+1)} \in L^{(i+1)}$ by $s^{(i+1)} = q^{(i)}(s^{(i)})$.
     - **ii.** Compute $H_{x^{(i)}}[f^{(i)}](s^{(i+1)})$ by making 2 queries to $f^{(i)}$.
     - **iii.** If $f^{(i+1)}(s^{(i+1)}) \neq H_{x^{(i)}}[f^{(i)}](s^{(i+1)})$, then REJECT.
   - **c.** If $f^{(\mathsf{r})}(s^{(\mathsf{r})}) \neq C$, then REJECT.
2. ACCEPT

The previous state of the art regarding the soundness of FRI is given by the following statement from [9]. In what follows let $J_\epsilon(x) = 1 - \sqrt{1 - x(1-\epsilon)}$.

▶ **Theorem 4** (FRI soundness (informal)). *Suppose $\delta^{(0)} \triangleq \Delta(f^{(0)}, \mathsf{RS}^{(0)}) > 0$. Let $n = |L^{(0)}|$. Then for any $\epsilon > 0$ there exists $\epsilon' > 0$ so that with probability at least*

$$1 - \frac{2 \log n}{\epsilon^3 |\mathbb{F}|} \tag{8}$$

over the randomness of the verifier during the COMMIT *phase, and for any (adaptively chosen) prover oracles $f^{(1)}, \ldots, f^{(\mathsf{r})}$, the* QUERY *protocol with repetition parameter ℓ outputs* accept *with probability at most*

$$\left(1 - \min\left\{\delta^{(0)}, 1 - (\rho^{1/4} + \epsilon')\right\} + \epsilon \log n\right)^{\ell}. \tag{9}$$

▶ Remark 5. Using the improved distance preservation of Lemma 2 in the analysis of FRI from [9], one immediately improves the factor 1/4 in the exponent in Equation (11) to an exponent of 1/3 (details omitted).

4.2 DEEP-FRI

We now describe our variation of FRI, that we call DEEP-FRI, for which we can give improved soundness guarantees, at the cost of a small increase in the query complexity (but no increase in the proof length or the number of queries to committed proofs – which is important in applications).

Before we can describe our protocol we introduce the operation of "quotienting", which allows us to focus our attention on polynomials taking certain values at certain points.

4.2.1 Quotienting

Suppose we a set $L \subseteq \mathbb{F}_q$ and a function $f : L \to \mathbb{F}_q$. Suppose further that we are given a point $z \in \mathbb{F}_q$ and a value $b \in \mathbb{F}_q$.

We define the function $\mathsf{QUOTIENT}(f, z, b) : L \to \mathbb{F}_q$ as follows. Let $Z(Y) \in \mathbb{F}_q[Y]$ be the polynomial $Z(Y) = Y - z$. Then we define $\mathsf{QUOTIENT}(f, z, b)$ to be the function $g : L \to \mathbb{F}_q$ given by:

$$g(y) = \frac{f(y) - b}{Z(y)}$$

(or more succinctly, $g = \frac{f-b)}{Z}$).

▶ **Lemma 6.** *Let $L \subseteq \mathbb{F}_q$. Let $z \in \mathbb{F}_q$ with $z \notin L$. Let $d \geq 1$ be an integer.*

Let $f : L \to \mathbb{F}_q$, and $b \in \mathbb{F}_q$. Let $g = \mathsf{QUOTIENT}(f, z, b)$. Then the following are equivalent:

- *There exists a polynomial $Q(X) \in \mathbb{F}_q[X]$ of degree at most $d - 1$ such that $\Delta(g, Q) < \delta$.*
- *There exists a polynomial $R(X) \in \mathbb{F}_q[X]$ of degree at most d such that $\Delta(f, R) < \delta$ and $R(z) = b$.*

Proof. If there is such a polynomial $Q, \deg(Q) \leq d - 1$ that agrees with g on all but a δ-fraction of entries, we can take $R = QZ + b$. Notice $\deg(R) \leq d$ because $\deg(Z) = 1$.

Conversely, if there is such a polynomial R that agrees with f on all but a δ-fraction of entries, we can take $Q = (R - b)/Z$. This is indeed a polynomial because $R - b$ vanishes on z, so $Z|(R - b)$ in the ring of polynomials.

Finally, by construction R agrees with f whenever g agrees with R and this completes the proof. ◀

4.3 DEEP-FRI

Recall: We have linear spaces $L^{(0)}, L^{(1)}, \ldots, L^{(\mathsf{r})}$, with dimensions $k, k-1, \ldots, k-\mathsf{r}$. We further have 1 dimensional subspaces $L_0^{(0)}, L_0^{(1)}, \ldots, L_0^{(\mathsf{r})}$ with $L_0^{(i)} \subseteq L^{(i)}$.

For this, it will be helpful to keep in mind the case that the domain $L^{(0)}$ is much smaller than the field $\mathbb{F}_q$ (maybe $q = |L^{(0)}|^{\Theta(1)}$).

▶ **Protocol 7** (DEEP-FRI).
Input: a function $f^{(0)} : L^{(0)} \to \mathbb{F}_q$ which is supposed to be of degree $< d^{(0)}$.

COMMIT **Phase:**

1. For each $i \in [0, \mathsf{r}-1]$:
   a. The verifier picks a uniformly random $z^{(i)} \in \mathbb{F}_q$.
   b. The prover writes down a degree one polynomial $B^{(i)}_{z^{(i)}}(X) \in \mathbb{F}_q[X]$ (which is supposed to be such that $B^{(i)}_{z^{(i)}}(x)$ equals the evaluation of the low degree polynomial $H_x[f^{(i)}]$ at $z^{(i)}$).
   c. The verifier picks uniformly random $x^{(i)} \in \mathbb{F}_q$.
   d. The prover writes down a function

$$f^{(i+1)} : L^{(i+1)} \to \mathbb{F}_q.$$

   (which on input y is supposed to equal $\mathsf{QUOTIENT}(H_{x^{(i)}}[f^{(i)}], z^{(i)}, B^{(i)}_{z^{(i)}}(x))$.)
2. The prover writes down a value $C \in \mathbb{F}_q$.

QUERY **Phase:**

1. Repeat ℓ times:
   a. The verifier picks a uniformly random $s^{(0)} \in D$.
   b. For each $i \in [0, \mathsf{r}-1]$:
      i. Define $s^{(i+1)} \in L^{(i+1)}$ by $s^{(i+1)} = q^{(i)}(s^{(i)})$.
      ii. Compute $H_{x^{(i)}}[f^{(i)}](s^{(i+1)})$ by making 2 queries to $f^{(i)}$.
      iii. If $H_{x^{(i)}}[f^{(i)}](s^{(i+1)}) \neq f^{(i+1)}(s^{(i+1)}) \cdot (s^{(i+1)} - z^{(i)}) + B^{(i)}_{z^{(i)}}(x^{(i)})$, then REJECT.
   c. If $f^{(\mathsf{r})}(s^{(\mathsf{r})}) \neq C$, then REJECT.
2. ACCEPT.

4.4 Analysis

The following theorem proves the soundness of the DEEP-FRI protocol.

▶ **Theorem 8** (DEEP-FRI). *Fix degree bound* $d^{(0)} = 3 \cdot 2^{\mathsf{r}} - 2$ *and* $\mathsf{RS}^{(0)} = \mathsf{RS}[\mathbb{F}_q, L^{(0)}, d^{(0)}]$. *Let* $n = |L^{(0)}|$.

For some $\epsilon, \delta > 0$, *let*

$$\delta^* = \delta - 2\mathsf{r}\epsilon,$$

$$\mathcal{L}^* = \mathcal{L}(\mathbb{F}_q, L^{(0)}, d^{(0)}, \delta^*),$$

$$\nu^* = 2\mathcal{L}^* \left(\frac{d^{(0)}}{q} + \epsilon \right)^{1/3} + \frac{4}{\epsilon^2 q}.$$

Then the following properties hold when the DEEP-FRI *protocol is invoked on oracle* $f^{(0)} : L^{(0)} \to \mathbb{F}_q$,

1. ***Prover complexity*** *is* $O(n)$ *arithmetic operations over* $\mathbb{F}$
2. ***Verifier complexity*** *is* $O(\log n)$ *arithmetic operations over* $\mathbb{F}$ *for a single invocation of the* QUERY *phase; this also bounds communication and query complexity (measured in field elements).*
3. ***Completeness*** *If* $f^{(0)} \in \mathsf{RS}^{(0)}$ *and* $f^{(1)}, \ldots, f^{(\mathsf{r})}$ *are computed by the prover specified in the* COMMIT *phase, then the* DEEP-FRI *verifier outputs* accept *with probability* 1.

4. ***Soundness*** *Suppose* $\Delta(f^{(0)}, \mathsf{RS}^{(0)}) > \delta$. *Then with all but probability*

$$\mathsf{err}_{\mathsf{COMMIT}} \leq \mathsf{r} \cdot \nu^* \leq (\log n) \cdot \nu^*. \tag{10}$$

and for any (adaptively chosen) prover oracles $f^{(1)}, \ldots, f^{(\mathsf{r})}$, *the* QUERY *protocol with repetition parameter* ℓ *outputs* accept *with probability at most*

$$\mathsf{err}_{\mathsf{QUERY}} \leq (1 - \delta^* + (\log n) \cdot \epsilon)^{\ell} \tag{11}$$

Consequently, the soundness error of FRI *is at most*

$$\mathsf{err}\,(\delta) \leq (\log n) \cdot \nu^* + (1 - \delta^* + (\log n) \cdot \epsilon)^{\ell} \tag{12}$$

We give a consequence below with a specific setting of parameters based on the Johnson bound.

▶ **Example 9.** Continuing with the notation of Theorem 8, fix degree bound $d^{(0)} = 3 \cdot 2^{\mathsf{r}} - 2$ and assume $n = |L^{(0)}| < \sqrt{q}$. Let $\mathsf{RS}^{(0)} = \mathsf{RS}[\mathbb{F}_q, L^{(0)}, d^{(0)}]$ and let $\rho = d^{(0)}/n$ be its rate.

Let $f^{(0)} : L^{(0)} \to \mathbb{F}_q$ be a function, and let $\delta^{(0)} = \Delta(f^{(0)}, RS^{(0)})$. Then with all but probability $\mathsf{err}_{\mathsf{COMMIT}} \leq O(q^{-\Omega(1)})$, the query phase will accept with probability at most: $\mathsf{err}_{\mathsf{QUERY}} \leq (\max(1 - \delta^{(0)}, \sqrt{\rho}) + o(1))^{\ell}$ as $n \to \infty$.

Proof. Note that $d^{(0)} \leq n \leq \sqrt{q}$.

Set $\delta = \min(\delta^{(0)}, 1 - \sqrt{\rho} - q^{-1/13})$, and apply the previous theorem. Theorem 20 implies that $\mathcal{L}^* < q^{1/13}/(2\sqrt{\rho}) = O(q^{1/13})$. Set $\epsilon = q^{-6/13}$. Hence

$$\nu^* = 2\mathcal{L}^* \left(d^{(0)} q^{-1} + q^{-6/13}\right)^{1/3} + 4q^{-6/13} = O(q^{-1/13}),$$

which implies $\mathsf{err}_{\mathsf{COMMIT}} \leq \tilde{O}(q^{-1/13})$.

If $\delta = \delta^{(0)}$, then $1 - \delta^* + (\log n)\epsilon = 1 - \delta + o(1)$. Otherwise $\delta = 1 - \sqrt{\rho} - q^{-1/13}$, and so

$$1 - \delta^* + (\log n)\epsilon = \sqrt{\rho} + q^{-1/13} + (\log n)\epsilon = \sqrt{\rho} + q^{-1/13} + (\log n) q^{-6/13}.$$

Thus $\mathsf{err}_{\mathsf{QUERY}} \leq (\max(1 - \delta, \sqrt{\rho}) + o(1))^{\ell}$. ◀

We now give an example setting of DEEP-FRI under the optimistic Conjecture 21.

▶ **Example 10.** Assume Conjecture 21. Continuing with the notation of Theorem 8, fix degree bound $d^{(0)} = 3 \cdot 2^{\mathsf{r}} - 2$ and $n = |L^{(0)}|$. Let $\mathsf{RS}^{(0)} = \mathsf{RS}[\mathbb{F}_q, L^{(0)}, d^{(0)}]$ and let $\rho = d^{(0)}/n$ be its rate.

Let $C = C_\rho$ be the constant given by Conjecture 21. Suppose $q > n^{24C}$.

Let $f^{(0)} : L^{(0)} \to \mathbb{F}_q$ be a function, and let $\delta^{(0)} = \Delta(f^{(0)}, RS^{(0)})$. Then with all but probability $\mathsf{err}_{\mathsf{COMMIT}} \leq O(q^{-\Omega(1)})$, the query phase will accept with probability at most: $\mathsf{err}_{\mathsf{QUERY}} \leq (1 - \delta^{(0)} + o(1))^{\ell}$ as $n \to \infty$.

Proof. Set $\epsilon = q^{-1/(6C)}$.

Set $\delta = \min(\delta^{(0)}, 1 - \rho - q^{-1/(6C)})$. Conjecture 21 gives us:

$$\mathcal{L}^* < n^C q^{1/6}.$$

We then apply the previous theorem. We get $\nu^* \ll O(\mathcal{L}^* \cdot (d/q + \epsilon)^{1/3} + \frac{1}{\epsilon^2 q}) \ll q^{-1/12}$, and this gives us the claimed bound on $\mathsf{err}_{\mathsf{COMMIT}}$.

For the bound on $\mathsf{err}_{\mathsf{QUERY}}$, we note that $\delta = \delta^{(0)} + o(1)$. This is because *every* function is within distance $1 - \rho$ of $\mathsf{RS}^{(0)}$ (this follows easily from polynomial interpolation). Thus

$$1 - \delta^* + (\log n)\epsilon = \delta + o(1),$$

and we get the desired bound on $\mathsf{err}_{\mathsf{COMMIT}}$. ◀

Verifier complexity and completeness follow by construction (see, e.g., [3] for detailed analysis of these aspects). We start by analyzing prover time and showing it is linear. In the rest of the section we prove the soundness bound of Theorem 8.

4.5 Prover complexity

The prover is involved in two sub-steps of the COMMIT phase, Items 1b and 1d. Inspection reveals that Item 1d requires $O(|L^{(i)}|)$ arithmetic operations. In what follows we show that Item 1b can also be computed in similar asymptotic complexity. To show this it suffices to prove the following claim.

▷ Claim 11 (Fast DEEP evaluation). Let $f : L \to \mathbb{F}$ be a function and L be a group (additive or multiplicative) of size 2^k for integer k. Let $P_f(X)$ be the interpolant of f as defined in Appendix A. There exists an algorithm that, given $z \in \mathbb{F}$, computes $P_f(z)$ using $O(|L|)$ arithmetic operations.

Proof. We compute within $O(|L|)$ arithmetic operations a pair (f', z') satisfying

- $f' : L' \to \mathbb{F}$ where L' is a group of size 2^{k-1}.
- $z' \in \mathbb{F}$.
- $P_f(z) = P_{f'}(z')$ where $P_{f'}$ is the interpolant of f'.

This suffices to solve the problem by induction, noticing the total sum of arithmetic operations is a geometric sum (the base case, in which $|L| = O(1)$, is trvially solvable using $O(1)$ arithmetic operations).

To construct f' let L_0 be a subgroup of L of size 2. The quadratic polynomial $q(X)$ whose roots are L_0 induces a 2-to-1 map on L. Let $L' = \{q(x) \mid x \in L\}$ be the image of this map on L and notice that L' is a group of size $|L|/2$, and for each $y \in L'$ there exists a unique pair, denoted $x_y, x'_y \in L$, such that $y = q(x_y) = q(x'_y)$. Furthermore, there exists a unique bivariate polynomial $Q_f(X, Y)$ satisfying:

- $\deg_X(Q_f) \leq 1$
- $\deg_Y(Q_f) \leq \deg(P_f)/2$
- $P_f(X) = Q_f(X, q(X)) \mod Y - q(X)$

See [3, Claim 4.2] for a proof. We define $z' = q(z)$ and for $y \in L'$ define $f'(y) = Q_f(z, y)$. By the third item above we have $f'(z') = Q_f(z, z') = Q_f(z, q(z)) = P_f(z)$. The second item above shows that $P_{f'}(Y)$ is the interpolant of f' because $\deg(P_{f'}) < |L'|$ and both f' and $P_{f'}$ agree on all of L'. This implies $P_{f'}(z') = f'(z')$ which we showed equals $P_f(z)$, so $P_f(z) = P_{f'}(z')$ as required. All that is left is to argue that f' can be computed from f using $O(|L|)$ arithmetic operations. This follows from the first bullet because each entry $f'(y)$ can be computed from $f(x_y)$ and $f(x'_y)$ by interpolating the degree-1 polynomial $Q_f(X, y)$ and evaluating it at z to obtain $Q_f(z, y) = f'(y)$. This completes the proof. ◁

4.6 Preparations

We do the analysis below for the case $\ell = 1$. The generalization to arbitrary ℓ easily follows.

Define $d^{(0)} = 3 \cdot 2^{\mathsf{r}} - 2$, and $d^{(i+1)} = d^{(i)}/2 - 1$. It is easy to check that $d^{(\mathsf{r})} = 1$. Define $\mathsf{RS}^{(i)} = \mathsf{RS}[\mathbb{F}_q, L^{(i)}, d^{(i)}]$. In the case of the honest prover (when $f^{(0)} \in \mathsf{RS}^{(0)}$), we will have $f^{(i)} \in \mathsf{RS}^{(i)}$ for all i.

Our analysis of the above protocol will track the agreement of $f^{(i)}$ with $\mathsf{RS}^{(i)}$. This agreement will be measured in a certain weighted way, which we define next.

4.6.1 The success probability at $s \in L^{(i)}$

There is a natural directed forest that one can draw on the vertex set

$$L^{(0)} \cup L^{(1)} \cup \ldots \cup L^{(\mathsf{r})},$$

namely, where $s \in L^{(i)}$ is joined to $q^{(i)}(s) \in L^{(i+1)}$ (and we say that s is a child of $q^{(i)}(s)$). Note that every vertex not in $L^{(0)}$ has two children.

Let $i \leq \mathsf{r} - 1$ and $s_0 \in L^{(i)}$. Let $s \in L^{(i+1)}$ be the parent of s_0, and let $s_1 \in L^{(i)}$ the sibling of s_0. We color s_0 GREEN if $f^{(i+1)}(s)$ is consistent with $f^{(i)}|_{\{s_0,s_1\}}$ according to the test

$$H_{x^{(i)}}[f^{(i)}](s) = f^{(i+1)}(s) \cdot (s - z^{(i)}) + B^{(i)}_{z^{(i)}}(x^{(i)})$$

and we color s_0 RED otherwise. Notice that a vertex and its sibling get the same color.

For $s \in L^{(\mathsf{r})}$, we color s GREEN if $f^{(\mathsf{r})}(s) = C$ and RED otherwise.

The QUERY phase of the protocol can be summarized as follows: we pick a uniformly random $s^{(0)} \in L^{(0)}$ and consider the path $s^{(0)}, s^{(1)}, s^{(2)} \ldots, s^{(\mathsf{r})}$ going through all its ancestors. If all these vertices are GREEN, then we ACCEPT, otherwise we REJECT.

To capture this, we define functions $\eta^{(i)} : L^{(i)} \to \mathbb{R}$ as follows. For $s \in L^{(i)}$, let $\eta^{(i)}(s)$ be the fraction of leaf-descendants $s^{(0)}$ of s for which the path from $s^{(0)}$ to s (including $s^{(0)}$ but not including s) consists exclusively of GREEN vertices. Observe that $p_{ACCEPT} = \mathbb{E}_{s \in L^{(\mathsf{r})}}[\eta^{(\mathsf{r})}(s) \cdot 1_{f^{(\mathsf{r})}(s)=C}]$ equals the probability that the QUERY phase accepts.

The exact quantity that we will track is as i increases is the weighted agreement:

$$\alpha^{(i)} = \mathsf{agree}_{\eta^{(i)}}[f^{(i)}, \mathsf{RS}^{(i)}].$$

Notice that

$$\alpha^{(0)} = 1 - \Delta(f^{(0)}, \mathsf{RS}^{(0)}),$$

and the acceptance probability, p_{ACCEPT} satisfies:

$$p_{ACCEPT} \leq \alpha^{(\mathsf{r})}.$$

Our main intermediate claim is that with high probability over the choice of $x^{(i)}, z^{(i)}, B^{(i)}_{z^{(i)}}$, we have that $\alpha^{(i+1)}$ is not much more than $\alpha^{(i)}$. This gives us that p_{ACCEPT} is not much more than $1 - \Delta(f^{(0)}, \mathsf{RS}^{(0)})$, as desired.

4.6.2 Operations AVG and ZERO

We define two important operations.

1. AVG. For a function $w : L^{(i-1)} \to \mathbb{R}$, we define the function $\mathsf{AVG}[w] : L^{(i)} \to \mathbb{R}$ as follows. Let $s \in L^{(i)}$, and let $\{s_0, s_1\} = (q^{(i-1)})^{-1}(s)$. Then define:

$$\mathsf{AVG}[w](s) = \frac{w(s_0) + w(s_1)}{2}.$$

2. ZERO. For a function $w : L^{(i)} \to \mathbb{R}$, and a set $S \subseteq L^{(i)}$, we define the function $\mathsf{ZERO}[w, S] : L^{(i)} \to \mathbb{R}$ as follows. For $s \in L^{(i)}$, we set:

$$\mathsf{ZERO}[w, S](s) = \begin{cases} 0 & s \in S \\ w(s) & s \notin S \end{cases}.$$

We can use these two operations to express $\eta^{(i+1)}$ in terms of $\eta^{(i)}$. Let $E^{(i+1)}$ denote the set of all $s \in S^{(i+1)}$ both of whose children are RED (i.e., the test

$$H_{x^{(i)}}[f^{(i)}](s) = f^{(i+1)}(s) \cdot (s - z^{(i)}) + B^{(i)}_{z^{(i)}}(x^{(i)})$$

fails at s).

Define $\theta^{(i+1)} : L^{(i+1)} \to \mathbb{R}$ by

$$\theta^{(i+1)} = \mathsf{AVG}[\eta^{(i)}].$$

Then we have:

$$\eta^{(i+1)} = \mathsf{ZERO}(\theta^{(i+1)}, E^{(i+1)}).$$

Analogous to our definition of

$$\alpha^{(i)} = \mathsf{agree}_{\eta^{(i)}}(f^{(i)}, \mathsf{RS}^{(i)}),$$

we define

$$\beta^{(i+1)} = \mathsf{agree}_{\theta^{(i+1)}}(H_{x^{(i)}}[f^{(i)}], \{P(Y) \in \mathbb{F}_q[Y] \mid \deg(P) \le d^{(i+1)} \text{ and } P(z^{(i)}) = B^{(i)}_{z^{(i)}}(x^{(i)})\}).$$

The following two lemmas control the growth of $\alpha^{(i)}$ and $\beta^{(i)}$.

▶ **Lemma 12.** *For all i, with probability at least $1 - \nu^*$ over the choice of $x^{(i)}, z^{(i)}$, we have:*

$$\beta^{(i+1)} \le \max(\alpha^{(i)}, 1 - \delta^*) + \epsilon.$$

We prove this using Theorem 25 in Appendix F.

▶ **Lemma 13.** *For all i,*

$$\alpha^{(i)} \le \beta^{(i)}.$$

We prove this using Lemma 6 in Appendix F.

We can now complete the proof of Theorem 8.

Proof. As observed earlier, $\alpha^{(0)} = 1 - \Delta(f^{(0)}, \mathsf{RS}^{(0)}) < 1 - \delta$.

The two lemmas above imply that with probability at least $1 - \mathsf{r}\nu^*$,

$$\alpha^{(\mathsf{r})} \le \max(\alpha^{(0)}, 1 - \delta^*) + \mathsf{r} \cdot \epsilon < (1 - \min(\delta, \delta^*) + \mathsf{r} \cdot \epsilon).$$

Finally, we use the observation that $p_{ACCEPT} \le \alpha^{(\mathsf{r})}$ to complete the proof. ◀

5 Second Application – The DEEP Algebraic Linking IOP (DEEP-ALI) protocol

The techniques used earlier in Theorem 3 and Section 4 can also be used to improve soundness in other parts of an interactive oracle proof (IOP) protocol. We apply them here to obtain a Scalable Transparent IOP of Knowledge (STIK) [2, Definition 3.3] with better soundness than the prior state of the art, given in [2, Theorem 3.4].

Proof systems typically use a few steps of reduction to convert problems of membership in a nondeterministic language L to algebraic problems regarding proximity of a function (or a sequence of functions) to an algebraic code like Reed-Solomon (or, in earlier works,

Reed-Muller). The goal of such a reduction is to maintain a large *proximity gap* γ, meaning that for instances in L, an honest prover will provide information that leads to codewords, whereas for instances not in L, any oracles submitted by the prover will be converted by the reduction, with high probability, to functions that are γ-far from the code. Considerable effort is devoted to increasing γ because it is the input to the proximity protocols (like FRI and DEEP-FRI) and the soundness of those protocols is correlated to γ (as discussed earlier, e.g., in Theorem 8).

The STIK protocol is a special case of this paradigm. It requires the prover to provide oracle access to a function $f : D \to \mathbb{F}$ that is supposedly an RS encoding of a witness for membership of the input instance in L. A set of t-local constraints is applied to f to construct a function $g : D \to \mathbb{F}$, along with a gap-gurantee: If f is indeed an encoding of a valid witness for the instance, then the resulting function $g : D \to \mathbb{F}$ is also be a member of an RS code. One of the tests that the verifier performs is a *consistency test* between f and g, and, prior to this work, this consistency test was applied to the functions f and g *directly*. This leads to a rather small gap $\gamma \leq \frac{1}{8}$ which results in a small soundness guarantee from the RPT protocol applied to f, g later on.

In this section we apply the DEEP technique to this setting. After f and g have been provided, the verifier samples a random $z \in \mathbb{F}_q$ and asks for the values of the interpolating polynomials of f, g on all t entries needed to check the consistency test. Our verifier now applies the QUOTIENT operation to f, g, using the information obtained from the prover. Crucially, we prove that a *single* consistency test, conducted over a large domain $D' \supset D$, suffices to improve the proximity gap to roughly $1 - \sqrt{\rho}$, a value that approaches 1 as $\rho \to 0$. Assuming Conjecture 21 the proximity gap is nearly-optimal, at $\gamma \approx 1 - \rho$ (compare with with the value $\gamma \leq 1/8$ obtained by prior works). Details follow.

We focus on the the Algebraic linking IOP protocol (ALI) of [2, Theorem B.15], and present a new protocol that we call DEEP-ALI (Protocol 17) that obtains the aforementioned improved proximity gap(s).

In what follows, we will first recall (a variant of) the language (or, more accurately, binary relation) which was the input to the ALI protocol of [2] and is likewise the input to our DEEP-ALI protocol. The description of the protocol follows in Section 5.2. Its basic properties are specified in Section 5.3 and we analyze its soundness in Theorem 15 and Section 5.4.

5.1 The Algebraic Placement and Routing (APR) Relation

In what follows we use the notation $\tilde{f}$ to refer to a polynomial in $\mathbb{F}[x]$. Note that the operator $|_D$ for $D \subseteq \mathbb{F}$ takes a polynomial to a function: $\tilde{f}\,|_D \colon D \to \mathbb{F}$.

We start by defining a simplified version of the Algebraic placement and routing relation (APR). See [2, Definition B.10]. In particular, we only use one witness polynomial. This relation will be the input to the reduction used in Protocol 17.

▶ **Definition 14.** *The relation R_{APR} is the set of pairs $(\mathtt{x}, \mathtt{w})$ satisfying:*

1. ***Instance format:*** *The instance $\mathtt{x}$ is a tuple $(\mathbb{F}_q, d, \mathcal{C})$ where:*
   - *$\mathbb{F}_q$ is a finite field of size q.*
   - *d is an integer representing a bound on the degree of the witness.*
   - *$\mathcal{C}$ is a set of $|\mathcal{C}|$ tuples (M^i, P^i, Q^i) representing constraints. M^i is the* mask *which is a sequence of field elements $M^i = \{M^i_j \in \mathbb{F}_q\}_{j=1}^{|M^i|}$. P^i is the* condition *of the constraint which is a polynomial with $|M^i|$ variables. $Q^i \in \mathbb{F}_q[x]$ is the* domain polynomial *of the constraint which should vanish on the locations where the constraint should hold.*

We further introduce the following notation:
- *Let* $\mathcal{M} = \{M_j^i \mid 1 \le i \le |\mathcal{C}|$ *and* $1 \le j \le |M^i|\} \subseteq \mathbb{F}_q$ *be the* full mask.
- *Let* $d_{\mathcal{C}} = \max_i \deg(P^i)$ *be the maximal total degree of the* P^i*s.*
- *Let* $Q_{\text{lcm}} \in \mathbb{F}_q[x]$ *be the least common multiple of the* Q^i*s.*

2. ***Witness format:*** *The witness* $\mathtt{w}$ *is a polynomial* $\tilde{f} \in \mathbb{F}_q[x]$*. A constraint* (M, P, Q) *is said to hold at a location* $x \in \mathbb{F}_q$ *if* $P(\tilde{f}(x \cdot M_1), \ldots, \tilde{f}(x \cdot M_{|M|})) = 0$*. We say that* $\tilde{f}$ *satisfies the constraint if the constraint holds at every* $x \in \mathbb{F}_q$ *for which* $Q(x) = 0$*.*
We say that $\mathtt{w}$ *satisfies the instance if and only if* $\deg(\tilde{f}) < d$ *and* $\tilde{f}$ *satisfies all of the constraints.*

To see that the notion of the R_{APR} relation defined above is strong enough, we follow the ideas from [2] and show a reduction from an Algebraic Intermediate Representation (AIR, see [2, Definition B.3]) to an APR. The following uses the notation from [2, Definition B.3]. Let $\mathtt{x} = (\mathbb{F}_q, T, \mathsf{w}, \mathcal{P}, C, \mathsf{B})$ be an instance of R_{AIR}. Pick a multiplicative subgroup $\langle\gamma\rangle \subseteq \mathbb{F}_q^\times$ of size $T \cdot \mathsf{w}$ and pick $\tilde{f}$ such that $\tilde{f}(\gamma^{t\mathsf{w}+j}) = w_j(t)$ for $t \in [T]$ and $i \in [\mathsf{w}]$ (here $[n] = \{0, \ldots, n-1\}$). For all the constraints in $\mathcal{P}$, choose the mask $M = \{1, \gamma, \ldots, \gamma^{2\mathsf{w}-1}\}$ and choose the domain polynomial whose zeros are $\{\gamma^{t\mathsf{w}}\}_{t \in [T-1]}$ ($Q(x) = (x^T - 1)/(x - \gamma^{-\mathsf{w}})$). Replace each boundary constraint $(i, j, \alpha) \in \mathsf{B}$ with a regular constraint with mask $M = \{1\}$, $P(x) = x - \alpha$ and $Q(x) = x - \gamma^{i\mathsf{w}+j}$.

5.2 The DEEP-ALI protocol

We now describe our new protocol, that achieves improved soundness, as stated in the following theorem.

▶ **Theorem 15** (DEEP-ALI soundness)**.** *Fix a code rate* $0 < \rho < 1$ *and a distance parameter* $0 < \delta \le 1 - \rho$*. Let* $D, D' \subseteq \mathbb{F}_q$ *be two evaluation domains such that* $|D| = d\rho^{-1}$ *and* $|D'| = d \cdot d_{\mathcal{C}}\rho^{-1}$*. Let* RPT_D*,* $\text{RPT}_{D'}$ *be two IOPPs with perfect completeness for the codes* $\mathsf{RS}[\mathbb{F}_q, D, (d - |\mathcal{M}|)/|D|])$ *and* $\mathsf{RS}[\mathbb{F}_q, D', (dd_{\mathcal{C}} - 1)/|D'|])$ *respectively. Let* ϵ, ϵ' *be the bounds on the soundness error (acceptance probability) for words that are at least* δ*-far from the corresponding code. Denote*

$$L = \max\{\mathcal{L}(\mathbb{F}_q, D, d, \delta), \mathcal{L}(\mathbb{F}_q, D', d \cdot d_{\mathcal{C}}, \delta)\}.$$

Then, there exists an IOP for R_{APR} *with perfect completeness and soundness error* $\epsilon + \epsilon' + \frac{2L^2(d \cdot d_{\mathcal{C}} + \deg(Q_{\text{lcm}}))}{q}$*.*

▶ **Example 16.** Fix a code rate $0 < \rho < 1$. Choosing DEEP-FRI as the RPT protocol and setting $\delta = 1 - \sqrt{\rho} - q^{-1/13}$ as in Example 9, using ℓ repetitions, we obtain an IOP for R_{APR} with perfect completeness and soundness error that approaches $2\rho^{\ell/2}$ as $q \to \infty$ assuming the parameters d, $d_{\mathcal{C}}$, $\deg(Q_{\text{lcm}})$ of the APR are constant with respect to q.

Proof. Theorem 20 implies that $L \le q^{1/13}/(2\sqrt{\rho}) = O(q^{1/13})$. Hence the expression $2L^2(d \cdot d_{\mathcal{C}} + \deg(Q_{\text{lcm}}))/q$ approaches 0 as $q \to \infty$. Moreover, Example 9 implies that ϵ, ϵ' approach $\rho^{\ell/2}$. ◀

We now describe the protocol that achieves the soundness of Theorem 15.

▶ **Protocol 17** (DEEP-ALI)**.**
1. The prover sends an oracle $f : D \to \mathbb{F}$ (which should be $\tilde{f}\,|_D$).
2. The verifier sends random coefficients $\alpha = (\alpha_1, \ldots, \alpha_{|\mathcal{C}|}) \in \mathbb{F}_q^{|\mathcal{C}|}$.

3. The prover sends an oracle $g_\alpha : D' \to F$ (which should be $\tilde{g}_\alpha \mid_{D'}$, where

$$\tilde{g}_\alpha(x) = \sum_{i=1}^{|\mathcal{C}|} \alpha_i \cdot \frac{P^i(\tilde{f}(x \cdot M_1^i), \ldots, \tilde{f}(x \cdot M_{|M^i|}^i))}{Q^i(x)}. \tag{13}$$

   Note that $\deg(\tilde{g}_\alpha) < d \cdot d_{\mathcal{C}})$.
4. The verifier sends a random value $z \in \mathbb{F}_q$.
5. Denote $\mathcal{M}_z = \{z \cdot M_j^i \mid 1 \le i \le |\mathcal{C}| \text{ and } 1 \le j \le |M^i|\}$. The prover sends $a_{\alpha,z} : \mathcal{M}_z \to \mathbb{F}$ (which should be $\tilde{f}|_{\mathcal{M}_z}$). The verifier deduces $b_{\alpha,z}$, the alleged value of $\tilde{g}_\alpha(z)$, using Equation (13).
6. Let $U(x)$, $Z(x)$ as defined in Section 4.2.1 for $\mathsf{QUOTIENT}(f, a_{\alpha,z})$ and let

$$h^1(x) = h^1_{\alpha,z}(x) = \mathsf{QUOTIENT}(f, a_{\alpha,z}) = \frac{f(x) - U(x)}{Z(x)},$$

$$h^2(x) = h^2_{\alpha,z}(x) = \mathsf{QUOTIENT}(g_\alpha, \{z \mapsto b_{\alpha,z}\}) = \frac{g_\alpha(x) - b_{\alpha,z}}{x - z},$$

   and note that the verifier has oracle access to h^1 and h^2 using the oracles f and g_α.
7. They use RPT_D and $\mathrm{RPT}_{D'}$ to prove that h^1 is at most δ-far from $\mathsf{RS}[\mathbb{F}_q, D, (d-|\mathcal{M}|)/|D|]$ (in other words, it is close to a polynomial of degree $< d - |\mathcal{M}|$) and that h^2 is at most δ-far from $\mathsf{RS}[\mathbb{F}_q, D', (dd_{\mathcal{C}} - 1)/|D'|]$.

5.3 Properties of DEEP-ALI

Note that in the original ALI protocol, the equivalent to the expression $P^i(\tilde{f}(x \cdot M_1^i), \ldots, \tilde{f}(x \cdot M_{|M^i|}^i))/Q^i(x)$ is sampled at Q random locations from the evaluation domain, where Q is the number of queries.

The main idea in DEEP-ALI is to use Quotienting to allow the verifier to choose *one* random element z from the *entire* field, and check the consistency between $\tilde{f}$ and $\tilde{g}$ only at $x = z$.

The fact that DEEP-ALI allows to sample from the entire field introduces several advantages over the ALI protocol from [2]:

Soundness. As described above, the reduction in ALI has lower bound 1/8 on the distance from the code for inputs that are not in the language, even for $\rho \to 0$. In DEEP-ALI the lower bound on the distance is $1 - \sqrt{\rho}$.

Query complexity. In ALI the verifier queries $|\mathcal{M}| \cdot \mathsf{Q}$ field elements as we need $|\mathcal{M}|$ elements to evaluate $P^i(\tilde{f}(x \cdot M_1^i), \ldots, \tilde{f}(x \cdot M_{|M^i|}^i))$. In DEEP-ALI the verifier queries $O(|\mathcal{M}| + \mathsf{Q})$ field elements as the evaluation of P^i is done once.

Verifier complexity. Previously, the verifier complexity was $\Omega(\mathsf{Q} \cdot T_{\text{arith}})$ (where T_{arith} is the arithmetic complexity of evaluating all the constraints). The verifier complexity in DEEP-ALI depends on $\mathsf{Q} + T_{\text{arith}}$ as we evaluate the constraints only once.

Prover complexity. It is possible to alter Definition 14 and DEEP-ALI to work with several witness polynomials $f_1, \ldots, f_{\mathsf{w}}$ (as was done in ALI). The prover complexity in this case will depend on $(\mathsf{w}\rho^{-1} + d_{\mathcal{C}}\rho^{-1} + T_{\text{arith}}d_{\mathcal{C}}) \cdot d$ instead of $(\mathsf{w}d_{\mathcal{C}}\rho^{-1} + T_{\text{arith}}d_{\mathcal{C}}) \cdot d$ (in ALI).

5.4 Soundness analysis

The proof of Theorem 15 will follow from the following lemma:

▶ **Lemma 18.** *Let $\mathcal{E}$ be the event that the* DEEP-ALI *verifier accepts. If*

$$\Pr[\mathcal{E}] \geq \epsilon + \epsilon' + \frac{2L^2(d \cdot d_{\mathcal{C}} + \deg(Q_{\text{lcm}}))}{q},$$

then there exists a polynomial of degree $< d$ satisfying the constraints.

Proof. Let $L(f) \subseteq \mathsf{RS}[\mathbb{F}_q, D, \rho]$ be the set of codewords that are at most δ-far from f. Similarly define $L(g_\alpha)$. We have $|L(f)|, |L(g_\alpha)| \leq L$.

Let $\mathcal{E}_1$ be the event where the verifier accepts and h^1 and h^2 are at most δ-far from the corresponding codes. Denote $\eta = 2L^2(d \cdot d_{\mathcal{C}} + \deg(Q_{\text{lcm}}))/q$. Then, $\Pr[\mathcal{E}_1] \geq \eta$. $\mathcal{E}_1$ implies that there exists a polynomial $\tilde{h}^1 = \tilde{h}^1_{\alpha,z}$ of degree $< d - |\mathcal{M}|$ such that $\left|\{x \in D : \tilde{h}^1(x) \neq \frac{f(x)-U(x)}{Z(x)}\}\right| < \delta|D|$. Hence $Z(x) \cdot \tilde{h}^1(x) + U(x) \in L(f)$. Similarly there exists a polynomial $\tilde{h}^2 = \tilde{h}^2_{\alpha,z}$ of degree $< d \cdot d_{\mathcal{C}} - 1$ such that $(x - z)\tilde{h}^2(x) + b \in L(g_\alpha)$.

Fix $\tilde{r}^1(x)$ (independent of α and z) to be the element in $L(f)$ maximizing the probability that $\tilde{r}^1(x) = Z(x) \cdot \tilde{h}^1(x) + U(x)$ given $\mathcal{E}_1$. Let $\mathcal{E}_2 \subseteq \mathcal{E}_1$ be the event that $\tilde{r}^1(x) = Z(x) \cdot \tilde{h}^1(x) + U(x)$. It follows that $\Pr[\mathcal{E}_2] \geq \eta/L$.

Fix $\tilde{r}^2_\alpha(x) \in L(g_\alpha)$ maximizing the probability that $\tilde{r}^2_\alpha(x) = (x - z)\tilde{h}^2(x) + b$ given $\mathcal{E}_2$ (note that $\tilde{r}^2_\alpha$ depends on α as the oracle g_α was sent only after the verifier sent α), and let $\mathcal{E}_3 \subseteq \mathcal{E}_2$ be the event where $\tilde{r}^2_\alpha(x) = (x - z)\tilde{h}^2(x) + b$. We have $\Pr[\mathcal{E}_3] \geq \eta/L^2$. This implies, $\Pr_\alpha[\Pr_z[\mathcal{E}_3] \geq \eta/(2L^2)] \geq \eta/(2L^2)$.

The event $\mathcal{E}_3$ implies

$$\tilde{r}^1 \mid_{\mathcal{M}_z} = U \mid_{\mathcal{M}_z} = a_{\alpha,z},$$
$$\tilde{r}^2_\alpha(z) = b_{\alpha,z}.$$

Recall that $b_{\alpha,z}$ was defined according to (13), so

$$b_{\alpha,z} = \sum_{i=1}^{|\mathcal{C}|} \alpha_i \cdot \frac{P^i(a_{\alpha,z}(z \cdot M^i_1), \ldots, a_{\alpha,z}(z \cdot M^i_{|M^i|}))}{Q^i(z)}.$$

Substituting values for $a_{\alpha,z}$ and $b_{\alpha,z}$ and multiplying by $Q_{\text{lcm}}(z)$ we obtain:

$$Q_{\text{lcm}}(z) \cdot \tilde{r}^2_\alpha(z) = \sum_{i=1}^{|\mathcal{C}|} \alpha_i \cdot P^i(\tilde{r}^1(z \cdot M^i_1), \ldots, \tilde{r}^1(z \cdot M^i_{|M^i|})) \cdot \frac{Q_{\text{lcm}}(z)}{Q^i(z)}. \tag{14}$$

Both sides of the equation are polynomials of degree $< d \cdot d_{\mathcal{C}} + \deg(Q_{\text{lcm}})$ in z. For every α for which $\Pr_z[\mathcal{E}_3] \geq \eta/(2L^2) = (d \cdot d_{\mathcal{C}} + \deg(Q_{\text{lcm}}))/q$, we have at least $d \cdot d_{\mathcal{C}} + \deg(Q_{\text{lcm}})$ many z's satisfying (14) and thus the two polynomials in (14) are identical. Let $G_\alpha(x)$ denote the the right-hand side of (14) (replacing z with x).

So far we have:

$$\Pr_\alpha[G_\alpha(x) \text{ is divisible by } Q_{\text{lcm}}(x)] \geq \eta/(2L^2) > 1/q.$$

Note that the set of α's satisfying this event forms a vector space. If its dimension was less than $|\mathcal{C}|$ then the probability would have been $\leq 1/q$. Hence this event holds for every α. Substituting the elements of the standard basis, we get that for every $1 \leq i \leq |\mathcal{C}|$,

$$P^i(\tilde{r}^1(x \cdot M^i_1), \ldots, \tilde{r}^1(x \cdot M^i_{|M^i|})) \cdot \frac{Q_{\text{lcm}}(x)}{Q^i(x)} \text{ is divisible by } Q_{\text{lcm}}(x).$$

Substituting any x for which $Q^i(x) = 0$ gives $P^i(\tilde{r}^1(x \cdot M^i_1), \ldots, \tilde{r}^1(x \cdot M^i_{|M^i|})) = 0$ which implies that $\tilde{r}^1$ satisfies all the constraints, as required. ◀

5.5 Further optimizations for practical implementation

As we saw, it makes sense to work with several witness polynomials rather than one, as it improves the prover complexity. Another optimization is to apply the RPT only once for both h^1 and h^2 by taking a random linear combination of the two (and using Theorem 3). To make this work, the prover writes the degree $< d \cdot d_{\mathcal{C}}$ polynomial $\tilde{g}(x)$ as:

$$\tilde{g}(x) = \sum_{i=0}^{d_C-1} x^i \tilde{g}_i(x^{d_C}),$$

where the $\tilde{g}_i$s are of degree $< d$, and it sends oracles to $\tilde{g}_i \mid_D$ instead of $\tilde{g} \mid_{D'}$. In total, we will have to run RPT on $\mathsf{w} + d_{\mathcal{C}}$ polynomials of degree $< d$, so we choose only one evaluation domain $D \subseteq \mathbb{F}_q$ satisfying $|D| = d\rho^{-1}$.

References

1 Scott Ames, Carmit Hazay, Yuval Ishai, and Muthuramakrishnan Venkitasubramaniam. Ligero: Lightweight Sublinear Arguments Without a Trusted Setup. In *Proceedings of the 24th ACM Conference on Computer and Communications Security*, October 2017.

2 Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Scalable, transparent, and post-quantum secure computational integrity. Cryptology ePrint Archive, Report 2018/046, 2018. Available at `https://eprint.iacr.org/2018/046`.

3 Eli Ben-Sasson, Iddo Bentov, Ynon Horesh, and Michael Riabzev. Fast Reed-Solomon Interactive Oracle Proofs of Proximity. In *Proceedings of the 45th International Colloquium on Automata, Languages, and Programming (ICALP)*, 2018. URL: `https://eccc.weizmann.ac.il/report/2017/134`.

4 Eli Ben-Sasson, Alessandro Chiesa, Michael A. Forbes, Ariel Gabizon, Michael Riabzev, and Nicholas Spooner. On Probabilistic Checking in Perfect Zero Knowledge. *Electronic Colloquium on Computational Complexity (ECCC)*, 23:156, 2016. URL: `http://eccc.hpi-web.de/report/2016/156`.

5 Eli Ben-Sasson, Alessandro Chiesa, Michael Riabzev, Nicholas Spooner, Madars Virza, and Nicholas P. Ward. Aurora: Transparent Succinct Arguments for R1CS. *IACR Cryptology ePrint Archive*, 2018:828, 2018. URL: `https://eprint.iacr.org/2018/828`.

6 Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive Oracle Proofs. In *Theory of Cryptography - 14th International Conference, TCC 2016-B, Beijing, China, Oct. 31 – Nov. 3, 2016, Proceedings, Part II*, pages 31–60, 2016. `doi:10.1007/978-3-662-53644-5_2`.

7 Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil P. Vadhan. Robust PCPs of Proximity, Shorter PCPs, and Applications to Coding. *SIAM Journal on Computing*, 36(4):889–974, 2006.

8 Eli Ben-Sasson, Swastik Kopparty, and Jaikumar Radhakrishnan. Subspace polynomials and limits to list decoding of Reed-Solomon codes. *IEEE Trans. Information Theory*, 56(1):113–120, 2010.

9 Eli Ben-Sasson, Swastik Kopparty, and Shubhangi Saraf. Worst-Case to Average Case Reductions for the Distance to a Code. In *33rd Computational Complexity Conference, CCC 2018, June 22-24, 2018, San Diego, CA, USA*, pages 24:1–24:23, 2018. `doi:10.4230/LIPIcs.CCC.2018.24`.

10 Eli Ben-Sasson, Swastik Kopparty, and Shubhangi Saraf. Worst-case to average case reductions for the distance to a code. *Electronic Colloquium on Computational Complexity (ECCC)*, 25:90, 2018. URL: `https://eccc.weizmann.ac.il/report/2018/090`.

11 Eli Ben-Sasson and Madhu Sudan. Short PCPs with Polylog Query Complexity. *SIAM Journal on Computing*, 38(2):551–607, 2008. Preliminary version appeared in STOC '05.

12 Alessandro Chiesa, Peter Manohar, and Igor Shinkar. On Axis-Parallel Tests for Tensor Product Codes. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2017, August 16-18, 2017, Berkeley, CA, USA*, pages 39:1–39:22, 2017. doi:10.4230/LIPIcs.APPROX-RANDOM.2017.39.

13 Irit Dinur. The PCP theorem by gap amplification. *Journal of the ACM*, 54(3):12, 2007.

14 Venkatesan Guruswami. Algorithmic Results in List Decoding. *Foundations and Trends® in Theoretical Computer Science*, 2(2):107–195, 2007. doi:10.1561/0400000007.

15 Carsten Lund, Lance Fortnow, Howard J. Karloff, and Noam Nisan. Algebraic Methods for Interactive Proof Systems. *Journal of the ACM*, 39(4):859–868, 1992.

16 Alexander Polishchuk and Daniel A. Spielman. Nearly-linear size holographic proofs. In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing*, STOC '94, pages 194–203, 1994.

17 Alexander A. Razborov. Lower bounds on the size of bounded depth circuits over a complete basis with logical addition. *Mathematical notes of the Academy of Sciences of the USSR*, 41(4):333–338, 1987.

18 Omer Reingold, Guy N. Rothblum, and Ron D. Rothblum. Constant-round interactive proofs for delegating computation. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 49–62, 2016. doi:10.1145/2897518.2897652.

19 Ron M. Roth. *Introduction to coding theory*. Cambridge University Press, 2006.

20 Guy N. Rothblum, Salil Vadhan, and Avi Wigderson. Interactive proofs of proximity: delegating computation in sublinear time. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 793–802. ACM, 2013.

21 Atri Rudra and Mary Wootters. Every list-decodable code for high noise has abundant near-optimal rate puncturings. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 764–773, 2014. doi:10.1145/2591796.2591797.

A Preliminaries

Functions

For a set D, we will be working with the space of functions $u : D \to \mathbb{F}$, denoted $\mathbb{F}^D$. For $u \in \mathbb{F}^D$ we use $u(z)$ to denote the zth entry of u, for $z \in D$. For $C \subset D$ we use $f|_C$ to denote the restriction of f to C. For two functions $f, g : D \to \mathbb{F}$ we write $f = g$ when the two functions are equal as elements in $\mathbb{F}^D$ and similarly say $f|_C = g|_C$ when their restrictions are equal as elements in $\mathbb{F}^C$.

Distance

We use $\Delta_D(u, v) = \Pr_{z \in D}[u(z) \neq v(z)]$ for relative Hamming distance, and omit D when it is clear from context. For a set $S \subset \mathbb{F}^D$ we use $\Delta_D(v, S) = \min_{s \in S} \Delta_D(v, s)$ and $\Delta_D(S) = \min_{s \neq s' \in S} \Delta_D(s, s')$ denotes the minimal relative distance of S. For $u \in \mathbb{F}^D$ let $B(u, \delta)$ denote the Hamming ball in $\mathbb{F}^D$ of normalized radius δ centered at u,

$$B(u, \delta) = \left\{u' \in \mathbb{F}^D \mid \Delta_D(u, u') < \delta\right\}.$$

Linear codes

An $[n, k, d]_q$-linear error correcting code is a linear space $V \subset \mathbb{F}_q^n$ of dimension k over $\mathbb{F}_q$ with minimal Hamming distance d. A generating matrix for V is a matrix $G \in \mathbb{F}_q^{n \times k}$ of rank k such that $V = \left\{Gx \mid x \in \mathbb{F}_q^k\right\}$.

Polynomials and RS codes

The *interpolant* of $f : D \to \mathbb{F}_q$ is the unique polynomial of degree $< |D|$ whose evaluation on D is f. The degree of f, denoted $\deg(f)$, is the degree of its interpolant. The RS code evaluated over domain $D \subset \mathbb{F}$ and rate ρ is denoted $\mathsf{RS}[\mathbb{F}, D, \rho] = \{f : D \to \mathbb{F} \mid \deg(f) < \rho|D|\}$. Sometimes it will be more convenient to work with degree rather than rate, in which case we abuse notation and define $\mathsf{RS}[\mathbb{F}, D, d] = \{f : D \to \mathbb{F} \mid \deg(f) < d\}$. We use capital letters like P, Q to denote polynomials and when we say $P \in \mathsf{RS}[\mathbb{F}, D, \rho]$ we mean that $\deg(P) < \rho|D|$ and associate P with the RS codeword that is its evaluation on D. We also use $\tilde{f}$ to denote the interpolant of a function f.

A.1 List Decoding

▶ **Definition 19** (List size for Reed-Solomon Codes). *For $u \in \mathbb{F}^D$, a set $V \subset \mathbb{F}^D$, and distance parameter $\delta \in [0, 1]$, let $\mathsf{List}(u, V, \delta)$ be the set of elements in V that are at most δ-far from u in relative Hamming distance. Formally, using $B(u, \delta)$ to denote the Hamming ball of relative radius δ centered around u, we have $\mathsf{List}(u, V, \delta) = B(u, \delta) \cap V$.*

*The code V is said to be (δ, L)-*list-decodable *if $|\mathsf{List}(u, V, \delta)| \leq L$ for all $u \in \mathbb{F}_q^D$.*

For $D \subseteq \mathbb{F}_q$, let $\mathcal{L}(\mathbb{F}_q, D, d, \delta)$ be the maximum size of $\mathsf{List}(u, V, \delta)$ taken over all $u \in \mathbb{F}_q^D$ for $V = \mathsf{RS}[\mathbb{F}_q, D, \rho = d/|D|]$.

We recall the fundamental Johnson bound, which says that sets with large minimum distance have nontrivial list-decodability. The particular version below follows, e.g., from [14, Theorem 3.3] by setting $d = (1 - \rho)|D|$ and $e = (1 - \sqrt{\rho} - \varepsilon)|D|$ there.

▶ **Theorem 20** (Johnson bound). *Let $V \subset \mathbb{F}^D$ be a code with minimum relative distance $\delta \in (0, 1)$. Then V is $(1-\sqrt{1-\delta}-\varepsilon, 1/(2\varepsilon\sqrt{1-\delta}))$-list-decodable for every $\varepsilon \in (0, 1-\sqrt{1-\delta})$.*

When V is a maximum distance separable (MDS) code like a Reed-Solomon code of rate ρ we have $\rho = 1 - \delta$, so V is $(1 - \sqrt{\rho} - \varepsilon, 1/(2\varepsilon\sqrt{\rho}))$-list-decodable for every $\varepsilon \in (0, 1 - \sqrt{\rho})$.

In particular, for Reed-Solomon codes this implies the following list-decodability bound:

$$\mathcal{L}(\mathbb{F}_q, D, d = \rho|D|, 1 - \sqrt{\rho} - \varepsilon) \leq O\left(\frac{1}{\varepsilon\sqrt{\rho}}\right).$$

Extremely optimistically, we could hope that Reed-Solomon codes are list-decodable all the way up to their distance with moderate list sizes. Staying consistent with the known limitations [8], we have the following brave conjecture.

▶ **Conjecture 21** (List decodability of Reed-Solomon Codes up to Capacity). *For every $\rho > 0$, there is a constant C_ρ such that every Reed-Solomon code of length n and rate ρ is list-decodable from $1 - \rho - \varepsilon$ fraction errors with list size $\left(\frac{n}{\varepsilon}\right)^{C_\rho}$. That is:*

$$\mathcal{L}(\mathbb{F}_q, D, d = \rho|D|, 1 - \rho - \varepsilon) \leq \left(\frac{|D|}{\varepsilon}\right)^{C_\rho}.$$

B Tightness of the one-and-a-half Johnson bound

Lemma 1 says that when V is a linear code with minimum distance λ, and u^* is some element that is δ-far from V, then for any u we have with high probability

$$\Delta(u^* + xu, V) \geq \min(\delta, J^{(1.5)}(\lambda) = 1 - (1 - \lambda)^{1/3}).$$

The rightmost term seems quite strange, as the $J^{(1.5)}(\cdot)$ function is unfamiliar in other settings of coding theory. However, as we show next, in certain settings this function gives the correct bound!

▶ **Lemma 22** (Tightness of one-and-a-half Johnson bound)**.** *For every member V_n of following family of RS codes $\left\{V_n = \mathsf{RS}[\mathbb{F}_{2^n}, \mathbb{F}_{2^n}, \rho = 2^{-3}] \mid n \in \mathbb{N}\right\}$ there exist $u_n^*, u_n \in \mathbb{F}_{2^n}^{\mathbb{F}_{2^n}}$ satisfying the following:*

- $\delta_{\max} \triangleq \Delta(u_n^*, V_n) = \Delta(u_n, V_n) = \frac{3}{4} = 1 - \rho^{2/3}$
- $\forall x \neq 0, \Delta(u_n^* + xu_n, V_n) \leq \frac{1}{2} = 1 - \rho^{1/3} = J^{(1.5)}(\Delta(V_n))$

Consequently, $\mathbf{E}[\delta_x] \leq J^{(1.5)}(\Delta(V_n)) + o(1) \leq \delta_{\max} - \frac{1}{4} + o(1)$.

We shall need to following claim in our proof of the lemma.

▷ Claim 23. For every $x \in \mathbb{F}_{2^n} \setminus \{0\}$ there exists a polynomial $P_x(Y) \in \mathbb{F}_{2^n}[Y]$ of the form

$$P_x(Y) = Y^{2^{n-1}} + xY^{2^{n-2}} + \tilde{P}_x, \quad \deg(\tilde{P}_x) < 2^{n-3}.$$

that has 2^{n-1} distinct roots in $\mathbb{F}_{2^n}$.

Proof. For $x \neq 0$ let $\beta_x = 1/x^2$, noticing β_x is unique because the map $\beta \mapsto \beta^2$ is bijective on $\mathbb{F}_{2^n}$. Let $\mathsf{Tr}(Z) \triangleq \sum_{i=0}^{n-1} Z^{2^i}$ be the trace function from $\mathbb{F}_{2^n}$ to $\mathbb{F}_2$. Define

$$S_x = \{y \in \mathbb{F}_{2^n} \mid \mathsf{Tr}(\beta_x y) = 0\}.$$

It is well known that $|S_x| = 2^{n-1}$ because the trace function has 2^{n-1} roots in $\mathbb{F}_{2^n}$. So we define

$$\begin{aligned} P_x(Y) &= \frac{1}{\beta_x^{2^{n-1}}} \cdot \mathsf{Tr}(\beta_x Y) = Y^{2^{n-1}} + \frac{1}{\beta_x^{2^{n-2}}} Y^{2^{n-2}} + \tilde{P}_x \\ &= Y^{2^{n-1}} + xY^{2^{n-2}} + \tilde{P}_x, \quad \deg(\tilde{P}_x(Y)) < 2^{n-3}. \end{aligned}$$

The last equality follows because $\beta_x^{2^{n-2}} = x$. ◀

Proof of Lemma 22. Consider V_n in this family and let $\mathbb{F} = \mathbb{F}_{2^n}$. Define $u^* : \mathbb{F} \to \mathbb{F}$ to be the function $u^*(y) = y^{2^{n-1}}$ and let $u : \mathbb{F} \to \mathbb{F}$ be the function $u(y) = y^{2^{n-2}}$.

By Claim 23, for every $x \in \mathbb{F} \setminus \{0\}$ there is some $v_x \in V_n$ and P_x with 2^{n-1} roots in $\mathbb{F}$ such that

$$P_x - (u^* + xu) + v_x = 0.$$

Then

$$\Delta(u^* + xu, v_x) = \Pr_{y \in \mathbb{F}}[u^*(y) + xu(y) \neq v_x(y)] = \Pr_y[P_x(y) \neq 0] = 1/2.$$

Thus we get that for all $x \in \mathbb{F} \setminus \{0\}$

$$\Delta(u^* + xu, V) \leq 1/2.$$

On the other hand,

$$\Delta(u, V) \geq 3/4,$$

because for all $v \in V_n$, $u - v$ is a polynomial of degree at most $2^{n-2} = |\mathbb{F}|/4$. This completes the proof. ◀

▶ Remark 24. Since this example is based on Reed-Solomon codes, it also easily translates into a limitation on the soundness of FRI. In particular, it means that the improvement to the soundness of FRI given in Remark 5 is optimal.

C DEEP Theorem – weighted version and general linear spaces

In this Section we provide a weighted version for Theorem 3 and generalize it to arbitrary linear spaces.

C.1 Weighted version

For application to Reed-Solomon Proximity Testing, it is more convenient to have a weighted version of the previous result. We briefly introduce some notation for dealing with weights, and then state the new version.

Let $u, v \in \mathbb{F}_q^D$. Let $\eta \in [0,1]^D$ be a vector of weights. We define the η-agreement between u and v by:

$$\mathsf{agree}_\eta(u,v) = \frac{1}{|D|} \sum_{i \in D | u_i = v_i} \eta(i).$$

For a subspace $V \subseteq \mathbb{F}_q^n$, we define

$$\mathsf{agree}_\eta(u,V) = \max_{v \in V} \mathsf{agree}_\eta(u,v).$$

▶ **Theorem 25.** *Let $\rho > 0$ and let $V = \mathsf{RS}[\mathbb{F}_q, D, d = \rho \cdot |D|]$. For $z, b \in \mathbb{F}_q$, we let*

$$V_{z,b} = \{Q(Y) \in V \mid Q(z) = b\}.$$

For $\alpha < 1$, let $L^* = \mathcal{L}(\mathbb{F}_q, D, d = \rho|D|, 1-\alpha)$ be the list-size for list-decoding V from $(1-\alpha)$-fraction errors (without weights).

Let $u, u^* \in \mathbb{F}_q^D$. For each $z \in \mathbb{F}_q$, let $B_z(X) \in \mathbb{F}_q[X]$ be an arbitrary linear function. Suppose that

$$\Pr_{x,z \in \mathbb{F}_q}[\mathsf{agree}_\eta(u^* + xu, V_{z,B_z(x)}) > \alpha] \geq \max\left(2L^*\left(\frac{d}{q} + \epsilon\right)^{1/3}, \frac{4}{\epsilon^2 q}\right), \tag{15}$$

Then there exist $v, v^* \in V$ and $C \subset D$ such that:

- $\sum_{y \in C} \eta(y) > (\alpha - \epsilon)|D|$,
- $u|_C = v|_C$,
- $u^*|_C = v^*|_C$.

Consequently, we have $\mathsf{agree}_\eta(u,V), \mathsf{agree}_\eta(u^*,V) \geq \alpha - \epsilon$.

The proof is nearly identical to the proof of Theorem 3 so we only highlight the changes. First, we observe that if $\eta_1 : D \to [0,1]$ is the the constant function with value 1, then $\mathsf{agree}_\eta(u,v) \leq \mathsf{agree}_{\eta_1}(u,v) = 1 - \Delta(u,v)$. Thus the set

$$\{Q(Y) \in \mathbb{F}_q[Y] \mid \deg(Q) \leq d, \mathsf{agree}_\eta(u^* + xu, Q) > \alpha\}$$

is contained in

$$\{Q(Y) \in \mathbb{F}_q[Y] \mid \deg(Q) \leq d, \Delta(u^* + xu, Q) < 1 - \alpha\}.$$

The size of this latter set is bounded by L^*, and thus the size of the former set is too. The proof then proceeds as before, until the very end, where we have a set $A' \subseteq \mathbb{F}_q$, with $|A'| \geq \frac{2}{\epsilon}$, and polynomials $P, P^* \in V$ such that for each $x \in A'$, $\mathsf{agree}_\eta(u^* + xu, P^* + xP) > \alpha$.

Then we take $C = \{y \in C \mid u^*(y) = P^*(y), u(y) = P(y)\}$, and our goal is to show that $\sum_{y\in C} \eta(y) > (\alpha - \epsilon)|D|$. To this end, consider:

$$\begin{aligned}
\alpha &< \frac{1}{|A'|} \sum_{x\in A'} \mathsf{agree}_\eta(u^* + xu, P^* + xP) \\
&= \frac{1}{|D| \cdot |A'|} \sum_{x\in A'} \sum_{y\in D} (\eta(y) \cdot 1_{u^*(y)+xu(y)=P^*(y)+xP(y)}) \\
&= \frac{1}{|D|} \sum_{y\in D} \eta(y) \left(\frac{1}{|A'|} \sum_{x\in A'} 1_{u^*(y)+xu(y)=P^*(y)+xP(y)} \right) \\
&\leq \frac{1}{|D|} \sum_{y\in C} \eta(y) + \frac{1}{|D|} \sum_{y\in D\setminus C} \eta(y) \cdot \frac{1}{|A'|} \\
&\leq \frac{1}{|D|} \sum_{y\in C} \eta(y) + \epsilon/2.
\end{aligned}$$

This implies that $\sum_{y\in C} \eta(y) > (\alpha - \epsilon)|D|$, and the rest of the proof is the same as before.

C.2 DEEP Lemma for general linear codes

Theorem 3 can be generalized to apply to arbitrary linear codes, and this is the focus of this section. We explain the basic principles for an $[n, k, d]_q$-linear code V with generating matrix $G \in \mathbb{F}_q^{k\times n}$, viewing codewords as evaluations of linear forms on the columns of G.

Let $D \subset \mathbb{F}_q^k$ be the set of columns of G. A linear form $\ell \in F_q^k$ can be "evaluated" at any any element x of D. Similarly, if we fix a set of points $S \subseteq \mathbb{F}_q^k$ (thinking $|S| \gg |D|$), we may evaluate the linear form ℓ at any point of S – this corresponds to evaluation outside the original domain D.

If we are given a function $u : D \to \mathbb{F}_q$ which is supposed to be the evaluations of a linear form ℓ on D, we can ask about what the evaluation of this linear form at a point $z \in S$ is. This is the viewpoint from which the DEEP lemma generalizes to general codes.

We start with two functions $u, u^* : D \to \mathbb{F}_q$ (which are supposed to correspond to linear forms, say $\ell \in \mathbb{F}_q^k$ and $\ell^* \in \mathbb{F}_q^k$. We have a verifier who samples $z \in S$ and asking for $a = \ell(z)$ and $a^* = \ell^*(z)$. Given these answers, the verifier now samples $x \in \mathbb{F}_q$ and computes $b = a^* + xa$ which is supposedly equal to $\ell^*(z) + \ell(z)$ (if u^* and u are indeed codewords of V). The result below says that if S is the set of columns of an error correcting code with good distance, and V has small list size for list-decoding up to radius δ, then with high probability, the function $u_x = u^* + xu$ has distance at least $\approx \min\{\Delta(u^*, V), \delta\}$ from the sub-code of V corresponding to the linear forms that evaluate to b on z.

▶ **Definition 26** (Robust). *A set $S \subseteq \mathbb{F}^k$ is called σ-robust if every subset of S of size σ contains a basis for $\mathbb{F}^k$.*

The following claim is well-known in coding theory (cf. [19, Problem 2.8]).

▷ Claim 27. Fix a full-rank matrix $G \in \mathbb{F}_q^{k\times N}, N \geq k$, and let $C = \left\{x \cdot M \mid x \in \mathbb{F}_q^k\right\}$ be the linear code generated by it. Then the set of columnss of G is σ-robust if and only if the minimum distance of C is at least $N - \sigma + 1$.

▶ **Lemma 28** (DEEP method for general linear codes). *Let V be an $[n,k,d]_q$-code that is (δ, L^*_δ)-list decodable for some $\delta > 0$, and fix $G \in \mathbb{F}_q^{k\times n}$ to be its generating matrix. Let $S \subset \mathbb{F}_q^k$ be a σ-robust set of size N. For $z \in S, b \in \mathbb{F}_q$, let*

$$V_{z,b} = \{v \in V \mid v = G \cdot \ell_v \ \textit{AND}\ \langle \ell_v, z\rangle = b\}$$

where $\langle v, z\rangle = \sum_{i=1}^k v_i, z_i$.

Let $u, u^* \in \mathbb{F}_q^n$. For each $z \in S$, let $B_z(X) \in \mathbb{F}_q[X]$ be an arbitrary linear function. Suppose that for some $\epsilon > 0$ the following holds,

$$\Pr_{x\in\mathbb{F}_q, z\in S}[\Delta(u^* + xu, V_{z,B_z(x)}) < \delta] \geq \max\left(2L^*_\delta\left(\frac{\sigma}{N} + \epsilon\right)^{1/3}, \frac{4}{\epsilon^2 q}\right), \tag{16}$$

Then there exist $v, v^* \in V$ and $C \subset [n]$ such that:

- $|C| \geq (1 - \delta - \epsilon)n$,
- $u|_C = v|_C$,
- $u^*|_C = v^*|_C$.

Consequently, we have $\Delta(u, V), \Delta(u^*, V) \leq \delta + \epsilon$.

The proof is analogous to the proof in the Reed-Solomon case, and appears in Appendix D.

Discussion

For the special case of RS codes, the DEEP method can be used to locally modify the problem and reduce degree. Indeed, the subcode $V_{z,b}$ in the case of RS codes corresponds is comprised of functions $f : D \to \mathbb{F}$ that are evaluations of polynomials of degree d whose interpolating polynomial P_f satisfies $P_f(z) = b$. From such a codeword, one can construct a new codeword $f_{z,b} : D \to \mathbb{F}$ defined by $f_{z,b}(x) = \frac{f(x)-b}{z}$, which is well-defined for all $z \notin D$. Notice that the transformation from f to $f_{z,b}$ is *1-local*, meaning that each entry of $f_{z,b}$ is constructed by making a single query to f. Furthermore, this transformation maps a subset of the code $RS[\mathbb{F}, D, d]$ to the code $\mathsf{RS}[\mathbb{F}, D, d-1]$, so we may use this transformation in RS IOPPs (as will done in the following section).

In contrast, for a general k-dimensional linear code V, the subcode $V_{z,b}$, while being an affine subspace of V, has less structure. In particular, it is not clear how to locally convert this subcode to a "nice" code of dimension $k-1$. An interesting middle ground, left to future work, is the case of algebraic codes like Reed Muller codes and Algebraic Geometry codes which resemble RS codes.

D Proof of the DEEP lemma for general codes

Proof of Lemma 28. To simplify notation set $\eta = \max\left(2L^*_\delta\left(\frac{\sigma}{N} + \epsilon\right)^{1/3}, \frac{4}{\epsilon^2 q}\right)$, and let $u_x = u^* + xu$.

Let $\mathcal{E}[x,z]$ denote the event "$\exists v \in \mathsf{List}(u_x, V, \delta), \langle v, z\rangle = B_z(x)$".

The assumption of Equation (16) now reads as

$$\Pr_{x\in\mathbb{F}_q, z\in S}[\mathcal{E}[x,z]] \geq \eta.$$

Thus we get,

$$\Pr_{x\in\mathbb{F}_q}[\Pr_{z\in S}[\mathcal{E}[x,z]] \geq \eta/2] \geq \eta/2 \tag{17}$$

Let

$$A = \left\{ x \in \mathbb{F}_q \mid \Pr_{z \in S}[\mathcal{E}[x,z]] \geq \eta/2] \right\}$$

and notice $|A| \geq \eta q/2$.

For $x \in \mathbb{F}_q$, pick $v_x \in V$ to be a member of $\mathsf{List}(u_x, V, \delta)$ that maximizes $\Pr_{z \in S}[P(z) = B_z(x)]$. Let $S_x = \{z \in S \mid \langle v_x, z \rangle = B_z(x)\}$ and set $\mu_x = |S_x|/s$. By definition, $|\mathsf{List}(u_x, V, \delta)| \leq L^*_\delta$, and so by the pigeonhole principle, for each $x \in A$ we have $\mu_x \geq \frac{\eta}{2L^*_\delta}$.

For x, β, γ picked uniformly from A we have

$$\begin{aligned}
\mathbf{E}_{x,\beta,\gamma \in A}\left[\frac{|S_x \cap S_\beta \cap S_\gamma|}{s}\right] &= \mathbf{E}_{z \in S, x,\beta,\gamma \in \mathbb{F}_q}[1_{z \in S_x \cap S_\beta \cap S_\gamma}] \\
&= \mathbf{E}_{z \in S}[\mathbf{E}_{x \in \mathbb{F}_q}[1_{z \in S_x}]^3] \\
&\geq \mathbf{E}_{z \in S, x \in \mathbb{F}_q}[1_{z \in S_x}]^3 \\
&\geq \left(\frac{\eta}{2L^*_\delta}\right)^3 \\
&> \frac{\sigma}{N} + \epsilon.
\end{aligned}$$

The second equality above follows from the independence of x, β, γ. The first inequality is an application of Jensen's inequality and the last inequality is by assumption on η.

Thus

$$\Pr_{x,\beta,\gamma}[|S_x \cap S_\beta \cap S_\gamma| > \sigma] \geq \epsilon.$$

Note that $\Pr_{x,\beta,\gamma}[x, \beta, \gamma \text{ are not all distinct}] < 3/|A|$. Since $|A| \geq \eta q/2 \geq 2/\epsilon^2 \geq 6/\epsilon$ we have $3/|A| \leq \epsilon/2$. Thus $\Pr_{x,\beta,\gamma}[x, \beta, \gamma \text{ are all distinct and } |S_x \cap S_\beta \cap S_\gamma| > \sigma] \geq \epsilon/2$.

This means that there are distinct x_0, β_0 such that

$$\Pr_\gamma[|S_{x_0} \cap S_{\beta_0} \cap S_\gamma| > d] \geq \epsilon/2.$$

Consider some γ where this happens. Let $\tilde{S} = S_{x_0} \cap S_{\beta_0} \cap S_\gamma$. Extend each of u^*, u to functions over domain S by defining for all $z \in S \setminus [n]$ $u^*(z) = B_z(0)$ and $u(z) = B_z(1)$, and for $x \in \mathbb{F}_q$ let $u_x(z) = u^*(z) + xu(z)$. Since V is systematic, we define $v_x(z) = \langle v_x|_{[k]}, z \rangle$ and thus extend v_x to domain $\tilde{S}$. By construction we know

$$(x_0, u_{x_0}), (\beta_0, u_{\beta_0}), (\gamma, u_\gamma)$$

are collinear. So, in particular,

$$(x_0, u_{x_0}|_{\tilde{S}}), (\beta_0, u_{\beta_0}|_{\tilde{S}}), (\gamma, u_\gamma|_{\tilde{S}}) \in \mathbb{F}_q \times \mathbb{F}_q^{\tilde{S}}$$

are likewise collinear, as a special case. By definition of $\tilde{S}$, we get that:

$$(x_0, v_{x_0}|_{\tilde{S}}), (\beta_0, v_{\beta_0}|_{\tilde{S}}), (\gamma, v_\gamma|_{\tilde{S}}) \in \mathbb{F}_q \times \mathbb{F}_q^{\tilde{S}}$$

are also collinear. Since $|\tilde{S}| > \sigma$ and S is σ-robust we conclude that v_γ is uniquely determined by $v_\gamma|_{\tilde{S}}$. This allows us to conclude that

$$(x_0, v_{x_0}), (\beta_0, v_{\beta_0}), (\gamma, v_\gamma) \in \mathbb{F}_q \times \mathbb{F}_q^n$$

are all collinear, recalling that $v_{x_0} \in \mathsf{List}(u_{x_0}, V, \delta)$.

Thus, an $\epsilon/2$-fraction of the $\gamma \in A$ have the "good" property that (γ, v_γ) is on the line passing through (x_0, v_{x_0}) and (β_0, v_{β_0}). Write this line as $v^* + xv$ and notice that for all "good" γ we have $v_\gamma = v^* + \gamma v$. Let $A' \subseteq A$ denote the set of good elements for this line, recording that $|A'| \geq |A| \cdot \epsilon/2 \geq 1/\epsilon$. By definition of $\mathsf{List}(u_x, V, \delta)$ and the assumption $v_x \in \mathsf{List}(u_x, V, \delta)$, we have that $\Delta(u_x, v_x) < \delta$ for $x \in A'$.

Consider the set $C \subset [n]$ defined by

$$C = \{y \in [n] \mid u^*(y) = v^*(y) \text{ AND } u(y) = v(y)\}.$$

For each $y \in [n] \setminus C$ there exists at most a single value of $x \in \mathbb{F}_q$ satisfying $u_x(y) = v_x(y)$ because

$$u_x(y) - v_x(y) = (u^*(y) - v^*(y)) + x \cdot (u(y) - v(y))$$

has at most one value x on which it vanishes. This implies

$$\delta \geq \mathbf{E}_{x \in A'}[\Delta_{[n]}(u_x, v_x)] \geq \frac{|[n] \setminus C|}{n} \cdot \left(1 - \frac{1}{|A'|}\right) \geq \left(1 - \frac{|C|}{n}\right) \cdot (1 - \epsilon) \geq 1 - \frac{|C|}{n} - \epsilon.$$

Rearranging, we get $\frac{|C|}{n} \geq 1 - (\delta + \epsilon)$ and this completes the proof. ◀

E The algebraic hash function

We now describe the algebraic hash function H_x.

The description of the hash function requires fixing some choices of certain subspaces. For each $i \in [0, \mathsf{r}]$ we choose $\mathbb{F}_2$-subspaces $L_0^{(i)}$ and $L^{(i)}$, satisfying the following properties.

1. $L_0^{(i)} \subseteq L^{(i)}$ with $\dim(L_0^{(i)}) = 1$,
2. $L^{(i+1)} = q^{(i)}(L^{(i)})$, where $q^{(i)}(X)$ is the *subspace polynomial* of $L_0^{(i)}$,

$$q^{(i)}(X) = \prod_{\alpha \in L_0^{(i)}} (X - \alpha),$$

thus this is an $\mathbb{F}_2$-linear map with kernel $L_0^{(i)}$). In particular, $\dim(L^{(i+1)}) = \dim(L^{(i)}) - 1$. Let $\mathcal{S}^{(i)}$ denote the set of cosets of $L_0^{(i)}$ contained in $L^{(i)}$.

Given $x \in \mathbb{F}$ and $f : L^{(i)} \to \mathbb{F}$, the hash of f with seed x is defined to be the function $H_x[f] : L^{(i+1)} \to \mathbb{F}$ as follows. For $s \in L^{(i+1)}$, let $s_0, s_1 \in L^{(i)}$ be the two roots of $q^{(i)}(X) - s$. Let $P_{f,s}(X) \in \mathbb{F}[X]$ be the unique degree ≤ 1 polynomial satisfying

$$P_{f,s}(s_0) = f(s_0),$$

$$P_{f,s}(s_1) = f(s_1).$$

Then we define

$$H_x[f](s) = P_{f,s}(x). \tag{18}$$

Observe that $H_x[f](s)$ can be computed by querying f on the set $\{s_0, s_1\}$ (this set is a coset of $L_0^{(i)}$, and we denote it by $S_s^{(i)}$).

To understand H_x better, it is instructive to see what it does to $\mathsf{RS}^{(i)}$. Let $f \in \mathsf{RS}^{(i)}$. The underlying polynomial $f(X)$ thus has degree at most $\rho|L^{(i)}|$. We may write $f(X)$ in base $q^{(i)}(X)$ as:

$$f(X) = a_0(X) + a_1(X)q^{(i)}(X) + \ldots + a_t(X)(q^{(i)}(X))^t, \tag{19}$$

where each $a_i(X)$ has degree at most 1, and $t \le \rho|L^{(i)}|/2$. Since the polynomials $f(X)$ and $P_{f,s}(X)$ agree on the roots of $q(X) - s$, we get that $f(X) \equiv P_{f,s}(X) \mod (q^{(i)}(X) - s)$. From Equation (19), we get that

$$P_{f,s}(X) = a_0(X) + a_1(X)s + \ldots + a_t(X)s^t.$$

In particular, for all $x \in \mathbb{F}$,

$$H_x[f](s) = P_{f,s}(x) = a_0(x) + a_1(x)s + \ldots + a_t(x)s^t,$$

and thus

$$H_x[f] \in \mathsf{RS}^{(i+1)}.$$

F Proof of Lemma 12 and Lemma 13

We first prove Lemma 12.

Proof of Lemma 12. Set $\gamma = \max(\alpha^{(i)}, 1 - \delta^*)$.

For simplicity, denote $f^{(i)}$ by f.

Recall the notation $P_{f,s}$ from the definition of the algebraic hash function H_x in Section E. We have that for each $s \in L^{(i+1)}$, $H_x[f](s) = P_{f,s}(x)$ is a linear function of x. Thus we can write $H_x[f] = u^* + xu$ for $u^*, u \in \mathbb{F}_q^{L^{(i+1)}}$, and for any fixed s, we have the formal polynomial equality $P_{f,s}(X) = u^*(s) + Xu(s)$.

We are interested in bounding the probability of the event $\beta^{(i+1)} > \gamma + \epsilon$. In other words, we want to bound the probability that there exists a polynomial $Q(Y) \in \mathbb{F}_q[Y]$ with $\deg(Q) < d^{(i+1)} + 1$ such that:

- $\mathsf{agree}_{\theta^{(i+1)}}(u^* + xu, Q) > \gamma + \epsilon$,
- $Q(z^{(i)}) = B^{(i)}_{z^{(i)}}(x)$.

This is exactly the scenario of Theorem 25. That Lemma tells us that if the probability in question is larger than ν^*, then there exist polynomials $P(Y), P^*(Y)$ of degree $\le d^{(i+1)}$ and a set $T \subseteq L^{(i+1)}$ such that:

- $$\frac{1}{|L^{(i+1)}|} \sum_{s \in L^{(i+1)}} \theta^{(i+1)} > \gamma,$$
- $u|_T = P|_T$,
- $u^*|_T = P^*|_T$.

Let

$$\hat{P}(X, Y) \triangleq P^*(Y) + X \cdot P(Y)$$

and notice that $\deg_X(\hat{P}) \le 1$, $\deg_Y(\hat{P}) \le d^{(i+1)}$.

Consider the polynomial $R(X) \triangleq \hat{P}(X, q^{(i)}(X))$. We have

$$\deg(R) \le 2d^{(i+1)} + 1 = d^{(i)} - 1 < d^{(i)}.$$

We claim that R agrees with f on $\tilde{T} = \bigcup_{s \in T} S^{(i)}_s$.

Take any $s \in T$ and let $S^{(i)}_s = \{s_0, s_1\} \in \mathcal{S}^{(i)}$ be the pair of roots of the polynomial $q^{(i)}(X) - s$.

First we show that the polynomials $P_{f,s}(X)$ and $\hat{P}(X,s)$ are identical. Indeed, $\hat{P}(X,s)) = P^*(s) + XP(s) = u^*(s) + Xu(s) = P_{f,s}(X)$. It follows that

$$f(s_0) = \hat{P}(s_0,s) = \hat{P}\left(s_0, q^{(i)}(s_0)\right) = R(s_0)$$

and similarly $f(s_1) = R(s_1)$. Therefore, R and f agree on $\tilde{T}$, as claimed.

We now use the above information to show that $\alpha^{(i)} = \mathsf{agree}_{\eta^{(i)}}(f,R) > \gamma$, which contradicts the definition of γ. Indeed,

$$\begin{aligned}
\mathsf{agree}_{\eta^{(i)}}(f,R) &= \frac{1}{|L^{(i)}|} \sum_{r \in L^{(i)} | f(r) = R(r)} \eta^{(i)}(r) \\
&\geq \frac{1}{|L^{(i)}|} \sum_{r \in \tilde{T}} \eta^{(i)}(r) \\
&= \frac{1}{|L^{(i)}|} \sum_{s \in T} \sum_{r \in S_s^{(i)}} \eta^{(i)}(r) \\
&= \frac{1}{|L^{(i)}|} \sum_{s \in T} |S_s^{(i)}| \cdot \theta^{(i)}(s) \quad \text{Since } \theta(s) \text{ equals the average of } \eta(r) \mid r \in S_s^{(i)} \\
&= \frac{1}{|L^{(i+1)}|} \sum_{s \in T} \theta^{(i)}(s) \\
&> \gamma.
\end{aligned}$$

This is the desired contradiction. ◀

Next we prove Lemma 13.

Proof of Lemma 13. By definition,

$$\beta^{(i)} = \mathsf{agree}_{\theta^{(i)}}(H_{x^{(i-1)}}[f^{(i-1)}], \{P(Y) \in \mathbb{F}_q[Y] \mid \deg(P) \leq d^{(i)} \text{ and } P(z^{(i-1)}) = B^{(i-1)}_{z^{(i-1)}}(x^{(i-1)})\})$$

Next, by the properties of quotienting, Lemma 6,

$$\begin{aligned}
\beta^{(i)} &= \mathsf{agree}_{\theta^{(i)}}(H_{x^{(i-1)}}[f^{(i-1)}], \{P(Y) \in \mathbb{F}_q[Y] \mid \deg(P) \leq d^{(i)} \text{ and } P(z^{(i-1)}) = B^{(i-1)}_{z^{(i-1)}}(x^{(i-1)})\}) \\
&= \mathsf{agree}_{\theta^{(i)}}(\mathsf{QUOTIENT}(H_{x^{(i-1)}}[f^{(i-1)}], z^{(i-1)}, B^{(i-1)}_{z^{(i-1)}}(x^{(i-1)})), \\
&\qquad \{P(Y) \in \mathbb{F}_q[Y] \mid \deg(P) \leq d^{(i)} - 1\}).
\end{aligned}$$

Now observe that $\eta^{(i)}$ is obtained from $\theta^{(i)}$ by zeroing out coordinates in $E^{(i)}$, and the only coordinates where $f^{(i)}$ can differ from $\mathsf{QUOTIENT}(H_{x^{(i-1)}}[f^{(i-1)}], z^{(i-1)}, B^{(i-1)}_{z^{(i-1)}}(x^{(i-1)}))$ are in $E^{(i)}$. Thus:

$$\begin{aligned}
\beta^{(i)} &\geq \mathsf{agree}_{\theta^{(i)}}(f^{(i)}, \{P(Y) \in \mathbb{F}_q[Y] \mid \deg(P) \leq d^{(i)} - 1\}) \\
&= \mathsf{agree}_{\theta^{(i)}}(f^{(i)}, \mathsf{RS}^{(i)}) \\
&= \alpha^{(i)}.
\end{aligned}$$

This completes the proof. ◀

On the Cryptographic Hardness of Local Search

Nir Bitansky[1]
Tel Aviv University, Israel
nirbitan@tau.ac.il

Idan Gerichter
Tel Aviv University, Israel
idangerichter@gmail.com

Abstract

We show new hardness results for the class of Polynomial Local Search problems (PLS):

- Hardness of PLS based on a falsifiable assumption on bilinear groups introduced by Kalai, Paneth, and Yang (STOC 2019), and the Exponential Time Hypothesis for randomized algorithms. Previous standard model constructions relied on non-falsifiable and non-standard assumptions.
- Hardness of PLS relative to random oracles. The construction is essentially different than previous constructions, and in particular is unconditionally secure. The construction also demonstrates the hardness of parallelizing local search.

The core observation behind the results is that the *unique proofs* property of incrementally-verifiable computations previously used to demonstrate hardness in PLS can be traded with a simple *incremental completeness* property.

2012 ACM Subject Classification Theory of computation → Computational complexity and cryptography

Keywords and phrases Cryptography, PLS, Lower Bounds, Incremental Computation

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.6

Funding This work was supported in part by ISF grant 18/484, and by Len Blavatnik and the Blavatnik Family Foundation.
Nir Bitansky: Supported by the Alon Young Faculty Fellowship.

1 Introduction

Local search is a well known approach for tackling optimization problems. Local search algorithms seek solutions that are *locally optimal* – they commonly try to improve on a given solution by considering small perturbations of it, called *neighbors*, and testing whether they are better according to some value function. Indeed, many algorithms use this approach with empirical success, for instance, the Simplex linear programming algorithm, the Lin-Kernighan TSP algorithm, and various machine learning algorithms [43, 38, 31]. Nevertheless, the approach has its limitations and many natural local search problems are not known to admit polynomial-time algorithms.

Aiming to characterize the computational complexity of local search, Johnson, Papadimitriou, and Yannakakis [33] introduced the class Polynomial Local Search (PLS). The class is defined by its canonical complete problem Local-Search (LS) [33, 29]. Here the input consists of two polynomial-size circuits: a *successor circuit* $\mathcal{S} : \{0,1\}^n \to \{0,1\}^n$, which given a string x outputs a string x', and a *value circuit* $\mathcal{F}$, which given a string x outputs an integer value. The goal is to find a string x which is locally optimal in the sense that $\mathcal{F}(x) \geq \mathcal{F}(\mathcal{S}(x))$. Here strings correspond to solutions, with a value assigned by $\mathcal{F}$, and the

[1] Member of the Check Point Institute of Information Security.

successor assigned by $\mathcal{S}$ represents "the best" neighbor. Important problems that are known to be PLS complete include finding a locally optimal max-cut [49] and finding pure Nash equilibrium in congestion games [22].

As observed in [33], PLS belongs to the class TFNP of *total search problems* in NP; namely, a local optimal solution always exists (and can be efficiently verified). As a result, it is unlikely that PLS contains NP-hard problems, as this would imply that NP = coNP [33, 41]. This barrier is shared of course by other prominent subclasses of TFNP, perhaps the most famous example being the class PPAD that captures the hardness of finding Nash equilibrium in bimatrix games [18, 14]. Accordingly, researchers have turned to seek for alternative evidence of hardness.

One natural place to look for such evidence is cryptography. Indeed, cryptography typically relies on problems closer to the boundary of P than to NP hard problems. For some subclasses of TFNP (such as PPP, PPA, and Ramsey) evidence of hardness has been shown based on standard cryptographic assumptions [45, 12, 32, 36, 50]. Establishing the hardness of PLS (and PPAD), however, has been far more elusive. Hubáček and Yogev [29], building on [7, 23], demonstrated a hard problem in PLS ∩ PPAD based on *indistinguishability obfuscation* [4], a strong cryptographic primitive, yet to be constructed under standard assumptions.[2] Similar hardness was then shown by Choudhuri et al. [16] assuming hardness of #SAT and the soundness of the Fiat-Shamir transform for the sumcheck protocol, which in turn can be based on *optimally-secure fully-homomorphic encryption against quasi-polynomial attackers* [13]. More recently, the same has been shown based on the hardness of iterated squaring and soundness of Fiat-Shamir for the iterated-squaring protocol of Pietrzak [46] by both Choudhuri et al. [15] and Ephraim et al. [21].

Basing the hardness of PLS (or PPAD) on standard assumptions remains an open problem.

1.1 Our Results

We provide new results regarding the hardness of PLS. Our first result shows worst-case hardness of PLS based on *the KPY assumption* on bilinear groups and the randomized Exponential Time Hypothesis (ETH). The KPY assumption was introduced recently by Kalai, Paneth, and Yang [34] toward the construction of publicly-verifiable delegation schemes. The assumption is similar in spirit to previous standard assumptions that generalize Decisional Diffie Hellman (e.g., [8, 9]). It is polynomially falsifiable and holds in the generic group model. Randomized ETH postulates that solving SAT in the worst case requires exponential-time even for randomized algorithms [30].

▶ **Theorem 1** (Informal). *Under the KPY assumption on bilinear groups and randomized* ETH*,* PLS *is hard in the worst case.*

The result is, in fact, more general and shows a reduction of PLS hardness to a certain type of incrementally-verifiable computation schemes (IVC) [51]. We then derive the required IVC from the work of [34]. We can also show average-case hardness at the cost of assuming superpolynomial hardness of the KPY assumption and average-case randomized ETH. See further details in the technical overview.

Our second result is a construction of PLS instances that are unconditionally hard in *the random oracle model*.

[2] More accurately, they show a hard problem in the class *Continuous Local Search* (CLS), which is known to be in PLS ∩ PPAD [19].

▶ **Theorem 2** (Informal). *Relative to a random oracle,* Local-Search *is unconditionally hard-on-average.*

Previously, Choudhuri et al. [16] showed hardness of CLS $\subseteq$ PLS $\cap$ PPAD relative to a random oracle, but their result is conditioned on hardness of #SAT (the construction does not relativize with respect to this hardness and hence it cannot be derived from the random oracle itself). Indeed, our construction is quite different from theirs. Whereas their construction is based on the sum-check protocol [39] (which explains the non-black-box reliance on #SAT hardness), ours comes from recent advancements in *proofs of sequential work* motivated by blockchain applications [40, 17, 20].

The reliance on proofs of sequential work, in fact, translates to a result on the hardness of parallelizing local search. Such hardness, in the random oracle model, was recently shown for CLS (and in particular for both PLS and PPAD) by Ephraim et al. [21] further assuming that repeated squaring modulo a composite is hard to parallelize [47]. Specifically, they construct CLS instances that can be solved in some tuneable sequential polynomial time, but cannot be solved much faster by parallel algorithms. Our result gives a similar kind of hardness for PLS in the random oracle model, without relying on any unproven computational assumptions. We elaborate on this in the technical overview below.

2 Technical Overview

To motivate our approach, we start by recalling previous approaches taken toward proving PLS (or rather CLS) hardness.

2.1 Hardness via Incremental Computation

So far, the common approach toward arguing cryptographic hardness of PLS went through an intermediate problem called Sink-of-Verifiable-Line (SVL) [1, 7]. An instance of SVL consists of a successor circuit $\mathcal{S}$ and a verifier circuit $\mathcal{V}$. The successor $\mathcal{S}$ implicitly defines a step-by-step (or, incremental) computation $s_{t+1} := \mathcal{S}(s_t)$ starting from a canonical source s_0 and ending at a sink s_T, for some superpolynomial T:

$$s_0 \xrightarrow{\mathcal{S}} \ldots \xrightarrow{\mathcal{S}} s_T$$

The verifier $\mathcal{V}(s^*, t)$ is promised to accept any s^* as the t-th state if and only if it is correct, namely $s^* = \mathcal{S}^{(t)}(s_0)$. The goal is to find the sink $s_T = \mathcal{S}^{(T)}(s_0)$, given $(\mathcal{S}, \mathcal{V})$.

It is not hard to see that solving SVL can be reduced to solving a local search problem in PLS [29], and thus it is sufficient to prove the hardness of SVL.

Valiant's Incrementally-Verifiable Computation and Uniqueness. A natural way to approach SVL hardness is Valiant's notion of incrementally-verifiable computation (IVC) [51]. According to this notion, we can take a Turing machine computation given by a machine M and input x with a configuration graph:

$$M_x^0 \to \ldots \to M_x^T \ ,$$

and associate with each intermediate configuration M_x^t a short proof π_t, attesting to its correctness. The proofs are incremental – the next proof π_{t+1} can be efficiently computed from π_t.

At first glance, this general notion seems to directly yield SVL hardness. Indeed, we can take any polynomial-space machine M solving some hard search problem R on input x and consider the corresponding incrementally-verifiable computation:

$$(M_x^0, \pi_0) \xrightarrow{\mathcal{S}} \ldots \xrightarrow{\mathcal{S}} (M_x^T, \pi_T) \ .$$

Here the successor simply advances the computation of M and incrementally computes the corresponding proofs. The SVL verifier is derived directly from the IVC verifier.

The reason that the above simple transformation does not work is that *the IVC proofs may not be unique*. In particular, a correct intermediate configuration may have many different accepting proofs attesting to its correctness, whereas SVL requires that only a single string can be accepted as the t-th node (for any t). Choudhuri et al. [16] show that the SVL uniqueness requirement can be somewhat relaxed and traded with *computational uniqueness* meaning that it is computationally hard to find more than a single proof for any given statement (even if such proofs exist).

So far, however, incrementally-verifiable computation with (even computational) uniqueness has not been achieved under standard assumptions. This is, in fact, the case even for specific (hard) computations, let alone for general ones.

2.2 Incremental Completeness

We show that the hardness of PLS does not require SVL hardness nor IVC with unique proof. Rather, we observe that IVC with a conceptually simple *incremental completeness* property suffices. Then we derive such IVC for polynomially-long computations from the delegation scheme construction of Kalai, Paneth, Yang [34] and combine it with ETH to obtain PLS hardness. We next explain the notion of incremental completeness and why it suffices. In the next subsection, we explain how it is obtained.

Incremental-Complete IVC and PLS Hardness. An incremental complete IVC has the property that given *any* accepting proof π for an intermediate state M_x^t of the computation, running the prover results in an accepting proof π' for the next state M_x^{t+1}. This differs from Valiant's original formulation where completeness is only guaranteed for the prescribed proof generated by iteratively applying the prover t times.

Let us now describe how to construct hard Local-Search instances from incrementally-complete IVC. Once again, we start from some hard computation given by a polynomial-space machine M and instance x. Rather than considering a verifiable computation chain, we shall consider a DAG with nodes of the form (t, C, π), where t is a timestamp, C is an alleged configuration of $M(x)$ at time t, and π is an IVC proof. We call such a node *valid* if the corresponding proof is accepting. For simplicity, we assume for now that the IVC is also perfectly sound, meaning that valid nodes are always such that C is the correct configuration M_x^t. Note that for every time t, there may very well be multiple corresponding valid nodes. We can visualize this as multiple "parallel universes", which the verifiable computation lives in simultaneously. Incremental completeness says that in *every* such universe, as time moves forward, the computation may proceed. In particular, by induction, we always reach a *sink* of the form (T, M_x^T, π) that contains the last state of the computation.

This naturally gives an LS instance. Valid nodes (t, M_x^t, π) are given value t and are always succeeded by another valid node $(t+1, M_x^{t+1}, \pi')$ derived by advancing the computation and incrementing the IVC proof. Invalid nodes are given value -1 and are succeeded by some canonical source in the DAG $(0, M_x^0, \varepsilon)$, where ε is the empty proof, which by convention is

accepted as a proof for M_x^0. The local maximums of the corresponding instance are exactly the sinks of the DAG, which by incremental completeness are of the form (T, M_x^T, π), containing the last state. Therefore, finding a local-maximum reduce to solving the underlying hard computational task given by the computation $M(x)$.

Our actual construction does not assume perfect soundness of the IVC, but only computational soundness. There may exist so called *fooling nodes* (t, C^*, π^*) that pass verification although C^* is not the correct configuration M_x^t. These nodes may be local maximums and finding them may not reduce to solving the underlying problem. Nonetheless, finding fooling local maximum corresponds to breaking the soundness of the IVC, which is computationally infeasible.

2.3 Obtaining IVC with Incremental Completeness

Valiant [51] put forth a general approach toward constructing IVC by *recursive proof composition*. Oversimplifying, the basic idea is that to move from the t-th configuration $C = M_x^t$ and proof π_t to the next configuration $C' = M_x^{t+1}$ and proof π_{t+1}, the new proof π_{t+1} asserts that:

1. There exists a proof π that passes IVC verification as a valid proof that $C = M_x^t$ (this part is recursive in the sense that it relies on IVC verification for smaller times stamps),
2. C' is obtained from C by applying M_x's transition function.

For this to be feasible, each "single-step proof" is computed using a succinct non-interactive NP proof system where verification time (and in particular, the size of the proof) is fixed and is not affected by the complexity of the NP relation. We observe that *any IVC construction following the above blueprint is incrementally complete* by definition as long as the succinct NP proof system used has perfect completeness.

Succinct Proof Systems. Succinct proofs for NP with unconditional soundness are unlikely to exist [26, 27]. Accordingly, Valiant suggested to instantiate the blueprint using computationally sound succinct arguments. This brings about some extra complications in following the blueprint. For once, proving that a previous proof *exists* is meaningless in the case of computational soundness. Rather we need the guarantee that a proof exists and can be efficiently found. Such systems are known as *succinct non-interactive arguments of knowledge* (SNARKs) and admit an efficient knowledge extractor that can efficiently extract a witness from a convincing prover.

Using such SNARKs enables the construction of IVC for arbitrary poly-time computations (or superpolynomial ones, if the SNARK is superpolynomially secure) [51, 5], and the corresponding constructions are incrementally complete by the fact that the underlying SNARKs are perfectly complete. (The actual constructions are somewhat different from the described blueprint; in particular, to deal with issues such as blowup in knowledge extraction complexity, they aggregate proofs in a tree-like fashion rather than a chain, in order to reduce the depth of the recursion. Incremental completeness holds just the same.)

Following the above, we can obtain an incrementally-complete IVC, and thus PLS hardness from SNARKs. However, the existence of SNARKs is a strong non-falsifiable assumption. It is only known under non-standard *knowledge assumptions* and its construction from standard assumptions is subject to barriers [24, 6].

Incrementally-Complete IVC from Weaker Succinct Arguments. In a recent work, Kalai, Paneth, and Yang [34] considered a relaxation of SNARKs called *quasi-arguments* that instead of standard knowledge extraction only requires a certain weaker *no-signaling extraction*

requirement. Unlike SNARKs, such arguments are not subject to any known lower bounds, and were in fact constructed from the learning with errors assumption in their privately-verifiable form [35, 44, 11, 3] and recently also in their publicly-verifiable form based on the so called KPY assumption in bilinear groups [34].

Furthermore, Kalai, Paneth, and Yang show that similarly to SNARKs such quasi-arguments can, in fact, be recursively composed toward verifying a long computation, provided that the computation is *deterministic*. Their explicit goal was not to construct IVC but rather to bootstrap quasi-arguments with a long common reference string (CRS) to succinct arguments with a short CRS. However, this is done by implicitly constructing an IVC (this connection is also apparent in previous constructions of SNARKs with a short CRS from ones with a long CRS [5]). The resulting IVC, like other IVCs based on recursive composition, is incrementally complete.

The Class of Supported Computations and the Reliance on ETH. In its native form, the IVC derived from the KPY construction supports computations of arbitrary polynomial length $T = \lambda^{O(1)}$ in the security parameter λ, with fixed prover-verifier running time (say, λ). However, as explained earlier, to get PLS hardness, we would like to apply the IVC for a hard (and thus superpolynomial) computation.

To circumvent this difficulty, we employ a fine-grained reduction. The idea is to construct instances which are tailor-made to be hard for LS algorithms of a specific polynomial running time. Fix any constant $c > 0$ and n^c-time algorithm $\mathcal{A}$ that supposedly solve LS (here n is the size of the LS instance $(\mathcal{S}, \mathcal{F})$). Consider a search problem R and a Turing machine M that solves it in polynomial time $T(\lambda)$, while no randomized algorithm can solve it in time T^δ for constant $\delta < 1$. If we can bound the *blowup* of the IVC reduction by at most $T^{\delta/c}$, we can use $\mathcal{A}$ to construct a randomized adversary $\mathcal{A}'$ that solves R in time T^δ and get a contradiction.

The blowup of the IVC reduction is polynomially related to (a) the input size, (b) the space used by M, and (c) the efficiency of the IVC scheme. Accordingly, we require a computation where there is an arbitrarily large (polynomial) gap between the space used and the hardness of the underlying problem. To this end, we use the randomized ETH assumption. By appropriately choosing the size of the underlying SAT instance, we get computations that can be solved using fixed-space and some polynomial time via brute-force but not in much faster polynomial time, even by randomized algorithms. To bound the blowup due to the IVC efficiency, we follow the efficiency analysis by [34] and adapt it to the incremental setting. (In the body, we show the above in the non-uniform setting: assuming non-uniform ETH and using standard non-uniform derandomization techniques while taking special care to bound the associated blowup.)

We note that the above only gives worst-case hardness. That is, we do not show a single distribution that is hard for all polynomial-time algorithms. Assuming slight super-polynomial security of the KPY assumption and average-case randomized ETH, we can get average-case PLS hardness by essentially the same reduction (in fact, we can slightly weaken average-case ETH to a slightly subexponential time hypothesis).

2.4 Unconditional Hardness in the Random Oracle Model

Our second result, in the random model, is based on recent advancements in *proofs of sequential work* (PoSW) [40]. Roughly speaking, in a PoSW, the prover is given a statement χ and a time parameter T, and can generate a corresponding proof π by making T sequential steps. The soundness requirement is that provers that make $\ll T$ sequential steps, will fail to

generate a valid proof for χ, except with negligible probability (namely, proof computation is not parallelizable). The construction we present relies on the PoSW of Cohen and Pietrzak [17] and its extension by Döttling, Lai, and Malavolta [20] to so called *incremental proof of sequential work*; both are proven sound in the random oracle model. We now move to recalling how these systems work, and explain how we derive from them hard PLS instances relative to random oracles.

The CP PoSW. To construct a PoSW, Cohen and Pietrzak [17] suggest an elegant tree labeling procedure that is inherently sequential. They consider a binary tree whose edges are directed from the leaves toward the root. In addition, they add directed edges from every node ℓ having a sibling r on the right, to all the leaves in the tree rooted at r. We call this the CP graph (see Figure 1 in Section 6 for an illustration). The nodes of the CP graph are then given random labels as follows: the label of any given node is obtained by applying a random oracle $\mathsf{H}_\chi = \mathsf{H}(\chi, \cdot)$ to the labels of its incoming nodes (the oracle is seeded with χ to guarantee an independent oracle for every statement). Intuitively, the added edges enforce that in order to compute the labels of some subtree rooted at some r, it is first necessary to compute the labels of its left sibling ℓ; in this sense, labeling must be done sequentially.

To turn this into an actual proof of sequential work π for χ, the prover publishes the label of the root of the entire tree. The verifier then responds with random challenge leaves to which the prover answers by providing all the labels in the corresponding paths toward the root along with the labels of the siblings along the path, as in standard Merkle tree authentication (indeed here the tree serves both the purpose of of a commitment and of enforcing the sequential nature). Finally, they make the proof non interactive by using the Fiat-Shamir transform. Cohen and Pietrzak prove that to successfully compute an accepting proof, a prover must sequentially call the random oracle $\approx T$ times, where T is the size of the tree.

Incremental PoSW and the DLM Construction. Cohen and Piertzak further show that the standard streaming algorithm for Merkle commitments [42], can also be applied to their labeling procedure. That is, one can keep track of a small amount of nodes (about $\log T$), each a root of some tree in a gradually growing forest, and when needed, merging two nodes on the same level. In other words, the CP labeling can be done by an incremental computation with small space.

One thing that is of course missing toward getting PLS hardness is verifiability of the corresponding intermediate states. One Naïve idea toward such local verifiability is to apply the CP proof strategy for each subtree in the incremetal labeling process. Indeed, each such subtree is, in fact, CP labeled on its own, so we can use Fiat-Shamir to compute random challenges for that particular subtree. While this is a step in the right direction it is still not enough for the goal of PLS hardness, since the intermediate proofs themselves are not computed incrementally – computing each local proof may take time proportional the size of the corresponding subtree.

Döttling, Lai, and Malavolta [20] suggested a neat idea to make the CP proofs incrementally computable.[3] Roughly speaking rather than sampling fresh challenges for each subtree, they suggested to derive them at random from the previously computed challenges of this subtree. In a bit more detail, whenever merging two subtrees rooted at ℓ and r into a new

[3] They were motivated by blockhains and aimed to make PoSW a process that can be advanced in a distributed fashion by different parties.

(taller) subtree, we assume by induction that each of the two already has a set of corresponding challenges S_ℓ and S_r, where say the size of each one is some parameter k. Then, for the new subtree, rooted at their parent v, we choose a new set S_v of k challenges by choosing k challenges at random from the $2k$ challenges in $S_\ell \cup S_r$ (this is again done non-interactive using Fiat-Shamir). Each of the challenges is again just a Merkle authentication path, which can be easily augmented to an authentication path for v. They prove that this choice of challenges guarantees, for every intermediate state, essentially the same soundness as the original CP construction guarantees for the final state.

Incremental Completeness and PLS Hardness. What we get from the DLM incremental PoSW (in the random oracle) is somewhat analogous to an IVC system in the standard model. However, while in the standard model we applied general IVC over some long (hard) computation, here there is no such underlying hardness. Rather, *the difficulty is only in computing the accepting proofs themselves*. Similarly to our standard model approach, here too we can address the concept of incremental completeness. Indeed, we observe that merging two accepting proofs in the DLM construction always yields an accepting proof. From here incremental completeness follows. Formally, we need to slightly augment the DLM construction to enforce consistency among different subtrees in a forest to guarantee such incremental completeness (see Section 6 for the details).

Choosing the parameters appropriately, yields problems in PLS relative to a random oracle which are exponentially hard-on-average. This translates to sub-exponential hardness of the canonical LS probalem (due to polynomial blowup of the reduction). Also, by choosing the size of the tree to be a polynomial of our choice, we get search problems in PLS which are moderately hard but "depth-robust" in the sense that they cannot be solved much faster by parallel algorithms [21]. This follows from the sequential hardness of DLM.

2.5 More Related Work on Total Search Problems

The class TFNP was introduced by Megiddo and Papadimitriou [41]. They observed that TFNP is unlikely to include NP hard problems as this would imply that NP = coNP. Furthermore, it is strongly believed that TFNP does not have complete problems (see for instance discussion in [25]). Toward understanding of the complexity of total search problems, Papadimitriou [45] introduced several "syntactic" subclasses of TFNP that cluster problems according to the mathematical argument used to establish their totality. Recently, Goldberg and Papadimitriou [25] presented a subclass called *provable* TFNP that contains previously defined subclasses (and can be seen as generalizing them) and admits complete problems.

A long line of works have investigated the complexity of TFNP and its subclasses, searching for efficient algorithms on one hand and evidence of hardness on the other. As explained earlier in the intro, cryptography is a natural place to look for such evidence. Indeed, basing the hardness of TFNP or its subclasses, on standard cryptographic assumptions, has been successful in several cases. For example, Papadimitriou [45] observed that PPP is hard assuming one-way permutations or collision-resistant hash functions, and Jeřábek [32], building on the work of Buresh-Oppenheim [12], demonstrated the hardness of PPA assuming factoring is hard. Hubáček, Naor and Yogev [28] showed TFNP-hardness assuming any average-case hard NP language exists (in particular one-way functions) and derandomization assumptions. Komargodski, Naor, and Yogev [37] showed that hardness of the class Ramsey is equivalent to the existence of multi-collision resistant hash functions.

As discussed earlier, demonstrating hardness for PPAD and PLS has been more challenging and so far only achieved under non-standard cryptographic assumptions. Trying to explain our failure so far to base such hardness on standard cryptographic primitives, Rosen, Segev

and Shahaf [48] show that TFNP hard instances constructed in a black box way from random oracles (or some variants thereof) must have a nearly exponential number of solutions (which is indeed the case in our result in the random oracle model).

Finally, we note that in the smooth complexity setting, several PLS complete problems, with natural noise distributions, have been shown to be solvable in smooth polynomial-time. This include finding locally optimal Max-Cut [2] and finding pure Nash equilibrium in network coordination games [10]. These algorithms suggest an explanation of why local search may be empirically easy, while hard instances exist under reasonable assumptions and can be sampled.

Organization

In Section 3, we recall some preliminaries including the definition of PLS. In Section 4, we construct a computational reduction from a search problem having an incremental complete IVC to LS. In Section 5, we instantiate the computational reduction under the KPY assumption and ETH to demonstrate PLS hardness in the plain model. In Section 6, we argue the hardness of PLS relative to a random oracle.

3 Preliminaries

Notation. Throughout, λ will denote security parameters. When the input size is not the security parameter, we denote it by n. For $a \leq b$ integers we denote by $[a, b]$ the set $\{a, ..., b\}$ and by $[a]$ the set $[1, a]$.

3.1 PLS Definition

The complexity class *polynomial local search* (PLS) consists of all TFNP search problems polynomial-time reducible to the Local-Search problem [33, 29] we denote by LS.

▶ **Definition 3** (Local-Search). *The search problem* LS *is the following: given two polynomial-size circuits* $\mathcal{S} : \{0,1\}^n \to \{0,1\}^n$ *and* $\mathcal{F} : \{0,1\}^n \to \mathbb{N}$*, find a string* $v \in \{0,1\}^n$ *such that* $\mathcal{F}(\mathcal{S}(v)) \leq \mathcal{F}(v)$.

We refer to $\mathcal{S}$ as the *successor circuit* and to $\mathcal{F}$ as the *value circuit*. Intuitively, the circuits $\mathcal{S}$ and $\mathcal{F}$ could be seen as representing an implicit DAG over $\{0,1\}^n$. In this graph, a node v is connected to $u = \mathcal{S}(v)$ if and only if $\mathcal{F}(u) > \mathcal{F}(w)$. This is a DAG since $\mathcal{F}$ induce a topological ordering. Notice also that every sink corresponds to a local maximum with respect to $\mathcal{F}$. With this perspective, the totality of LS follows from the fact *every finite DAG has a sink*.

Oracle aided PLS. We denote by $\mathcal{C}^{\mathcal{O}} = \{\mathcal{C}^{\mathcal{O}}_\lambda\}$ an oracle aided circuit family. That is a circuit family with oracle gates to $\mathcal{O}_\lambda : \{0,1\}^{r(\lambda)} \to \{0,1\}^{\ell(\lambda)}$. The complexity class $\mathsf{PLS}^{\mathcal{O}}$ relative to an oracle $\mathcal{O}$, is naturally defined as all $\mathsf{TFNP}^{\mathcal{O}}$ search problem polynomial-time reducible to $\mathsf{LS}^{\mathcal{O}}$.

▶ **Definition 4** (Oracle Aided Local-Search). *The search problem* $\mathsf{LS}^{\mathcal{O}}$ *is the following: given two polynomial-size, oracle aided, circuits* $\mathcal{S}^{\mathcal{O}} : \{0,1\}^n \to \{0,1\}^n$ *and* $\mathcal{F}^{\mathcal{O}} : \{0,1\}^n \to \mathbb{N}$*, find a string* $v \in \{0,1\}^n$ *such that* $\mathcal{F}^{\mathcal{O}}(\mathcal{S}^{\mathcal{O}}(v)) \leq \mathcal{F}^{\mathcal{O}}(v)$.

4 PLS Hardness from IVC

In this section, we give a computational reduction from search problems having an IVC with incremental completeness to LS. This, in turn, establishes that any (worst/average-case) hard search problem, having an IVC with incremental completeness, implies the hardness of PLS (in the worst/average-case, respectively). We start by formally defining IVC with incremental completeness.

4.1 IVC with Incremental Completeness

Conventions. Let M be a Turing machine with $T = T(\lambda)$ and $S = S(\lambda)$ bounds on its run time and space, respectively, for input of length λ. Throughout, we assume w.l.o.g that M always makes exactly T steps and the size of a configuration is exactly S. Let $x \in \{0,1\}^\lambda$ be an input to M, we denote by $M_x^t \in \{0,1\}^S$ the configuration in the t step of the computation $M(x)$. We denote by $M_x : \{0,1\}^S \to \{0,1\}^S$ the transition circuit between configurations.

▶ **Definition 5** (IVC with Incremental Completeness)**.** *Let M be a Turing machine with $T = T(\lambda)$ and $S = S(\lambda)$ bounds on its run time and space, respectively, for input of length λ.*

An Incremental Verifiable Computation scheme (IVC) for M, with incremental completeness, consists of three algorithms $\mathsf{IVC} = (\mathsf{IVC.G}, \mathsf{IVC.P}, \mathsf{IVC.V})$:

- $\mathsf{IVC.G}(x)$ *is a randomized algorithm that given $x \in \{0,1\}^\lambda$ outputs public parameters pp.*
- $\mathsf{IVC.P}(pp, t, C, \pi)$ *is a deterministic algorithm that given public parameters pp, a natural number t, an arbitrary configuration C and arbitrary proof π, outputs a proof π'.*
- $\mathsf{IVC.V}(pp, t, C, \pi)$ *is a deterministic algorithm that given public parameters pp, a natural number t, an arbitrary configuration C and arbitrary proof π, outputs* ACC *or* REJ.

We make the following requirements:

1. ***Incremental Completeness:*** *For every security parameter $\lambda \in \mathbb{N}$:*
   a. *For every input $x \in \{0,1\}^\lambda$, time $t \in [0, T-1]$ and candidate proof $\pi \in \{0,1\}^*$:*

$$\Pr\left[\begin{array}{l|l} \mathsf{IVC.V}(pp,t,M_x^t,\pi) = \mathsf{ACC} \implies & pp \leftarrow \mathsf{IVC.G}(x) \\ \mathsf{IVC.V}(pp,t+1,M_x^{t+1},\pi') = \mathsf{ACC} & \pi' = \mathsf{IVC.P}(pp,t,M_x^t,\pi) \end{array}\right] = 1 \ ,$$

   where M_x^t, M_x^{t+1} are the configurations in the t and $t+1$ steps of the computation of $M(x)$, respectively.
   b. *For every input $x \in \{0,1\}^\lambda$:*

$$\Pr\left[\mathsf{IVC.V}(pp,0,M_x^0,\varepsilon) = \mathsf{ACC} \;\middle|\; pp \leftarrow \mathsf{IVC.G}(x)\right] = 1 \ ,$$

   where M_x^0 is the first configuration of the computation of $M(x)$, and ε is the empty proof.
2. ***Soundness:*** *For every efficient adversary $\mathcal{A}$, there exists a negligible function μ such that for every $\lambda \in \mathbb{N}$ and $x \in \{0,1\}^\lambda$:*

$$\Pr\left[\begin{array}{l|l} C^* \neq M_x^t & pp \leftarrow \mathsf{IVC.G}(x) \\ \mathsf{IVC.V}(pp,t,C^*,\pi^*) = \mathsf{ACC} & (t, C^*, \pi^*) \leftarrow \mathcal{A}(x, pp) \end{array}\right] \leq \mu(\lambda) \ ,$$

   where M_x^t is the configuration in the t step of the computation of $M(x)$.
3. ***Efficiency:*** *The efficiency of* IVC*, denoted by $\mathbb{T}_{\mathsf{IVC}}(\lambda)$, is the maximal worst-case run-time among $\mathsf{IVC.G}, \mathsf{IVC.P}, \mathsf{IVC.V}$ for input having security parameter λ. We require $\mathbb{T}_{\mathsf{IVC}}(\lambda) \leq p(\lambda)$ for a fixed polynomial p.*

4.2 Computationally Sound Karp Reduction

As discussed in the introduction, since we use IVC with computational soundness, the reduction we give is also computationally sound. In what follows, we define the notion of computationally sound Karp reduction.

For $R, R' \in \mathsf{FNP}$, a computationally sound reduction from R to R' consists of a pair of efficient algorithms $(\mathcal{X}, \mathcal{W})$. Where $\mathcal{X}(x)$ is a randomized algorithm that given an instance x of R, samples an instance $x' \leftarrow \mathcal{X}(x)$ of R'. $\mathcal{W}(w')$ is a deterministic algorithm that translates a witness w' for x', to a candidate witness w of x. We require that its infeasible to find w', such that $w = \mathcal{W}(w')$ is not a witness for x in R.

▶ **Definition 6** (Computational Karp Reduction). *For $R, R' \in \mathsf{FNP}$, a computational Karp reduction from R to R' consists of a pair $(\mathcal{X}, \mathcal{W})$, where $\mathcal{X}(x)$ is a randomized efficient algorithm and $\mathcal{W}(w)$ is a deterministic efficient algorithm. We make the following requirement:*

- ***Computational Soundness:*** *For every efficient adversary $\mathcal{A}$, there exists a negligible function μ such that for every $\lambda \in \mathbb{N}$ and $x \in \{0,1\}^\lambda$ for which R_x is non-empty:*

$$\Pr\left[\begin{array}{l|l} w' \in R'_{x'} & x' \leftarrow \mathcal{X}(x) \\ w \notin R_x & w' \leftarrow \mathcal{A}(x') \\ & w = \mathcal{W}(w') \end{array}\right] \leq \mu(\lambda) \ ,$$

where R_x and $R'_{x'}$ are the set of witnesses for x in R and x' in R' respectively.

▶ **Definition 7** (Reduction Efficiency). *The efficiency of a computational Karp reduction $(\mathcal{X}, \mathcal{W})$, denoted $\mathbb{T}_{\mathsf{Red}}(\lambda)$, is the maximum of $\mathbb{T}_{\mathcal{X}}(\lambda), \mathbb{T}_{\mathcal{W}}(\lambda)$ where:*

1. *$\mathbb{T}_{\mathcal{X}}(\lambda)$ is the worst-case run-time of $\mathcal{X}$ for input $x \in \{0,1\}^\lambda$.*
2. *$\mathbb{T}_{\mathcal{W}}(\lambda)$ is the worst-case run-time of $\mathcal{W}$ for input $w' \in \left\{ w' \;\middle|\; \begin{array}{l} x \in \{0,1\}^\lambda \\ x' \in \mathsf{Supp}(\mathcal{X}(x)) \\ w' \in R_{x'} \end{array} \right\}$.*

Note that $\mathbb{T}_{\mathsf{Red}}(\lambda)$ is a bound on instance size sampled using $\mathcal{X}$.

4.3 The Hardness Reduction

Using the notion of computational Karp reduction, our hardness result is formalized in the following theorem.

▶ **Theorem 8** (IVC Reduction). *Let $R \in \mathsf{FNP}$, solvable by a polynomial-space Turing machine M. If there exists an IVC scheme with incremental completeness for M, there exists a computationally sound Karp reduction $(\mathcal{X}, \mathcal{W})$ from R to LS. The efficiency of the reduction is*

$$\mathbb{T}_{\mathsf{Red}} = \mathrm{poly}(\mathbb{T}_{\mathsf{IVC}}, S, \lambda, |M|) \ ,$$

where $S(\lambda)$ is a bound on M space and $\mathbb{T}_{\mathsf{IVC}}(\lambda)$ is the efficiency of the IVC. The polynomial doesn't depend on the IVC scheme, M nor R.

▶ Remark 9 (Efficiency Independent of T). Jumping ahead, the fact that $\mathbb{T}_{\mathsf{Red}}$ does not depend directly on the time of the computation T, plays an important role in the fine-grained reduction for polynomial-time computations presented in Section 5.

Next, we describe the reduction and in the following section we analyze its security.

4.3.1 The Reduction

Let $\mathsf{IVC} = (\mathsf{IVC.G}, \mathsf{IVC.P}, \mathsf{IVC.V})$ be an incremental verifiable computation scheme with incremental completeness for M. Let $S = S(\lambda)$ be the polynomial bound on space on M when executed on $x \in \{0,1\}^\lambda$, guaranteed by the statement. Recall that we denote by M_x^t the configuration in the t step of the computation of $M(x)$. We assume w.l.o.g that the configuration size is exactly S. Let $T = T(\lambda) = 2^{S(\lambda)} \leq 2^{\mathrm{poly}(\lambda)}$ be an upper bound on the running time of M. We further assume w.l.o.g that M always makes exactly T steps. With the above notation, the configuration graph of M when executing x is:

$$M_x^0 \longrightarrow \cdots \longrightarrow M_x^T \ .$$

In the following, we describe the construction of $\mathcal{X}(x)$ for an input $x \in \{0,1\}^\lambda$. Recall that we denote by $M_x : \{0,1\}^S \to \{0,1\}^S$ the transition circuit of $M(x)$, taking a configuration as input and returning the next configuration as output.

Instance translation. The algorithm $\mathcal{X}(x)$ begins by sampling $pp \leftarrow \mathsf{IVC.G}(x)$. Given pp, we apply a deterministic procedure I to generate an LS instance $(\mathcal{S}, \mathcal{F}) = I(x, pp)$. The constructed LS instance corresponds to a graph on vertices of fixed polynomial length $\ell = \ell(\lambda) \geq \lambda$. Each vertex is of the form $(t, C, \pi) \in \{0,1\}^\ell$ where t is parsed as an integer in $[0, T]$, C as a configuration of length S and π as a proof for the IVC padded to length ℓ if needed. We describe the successor $\mathcal{S}$ and value circuits $\mathcal{F}$ formally in the following.

▶ Remark 10. We assume w.l.o.g that M_x always outputs something. If the transition function of M fails to generate the next configuration given C, it will simply output M_x^0, the initial state.

▶ Remark 11. We assume w.l.o.g that $\mathcal{F}$ outputs values in $[-1, T]$.

In what follows, $v_0 := (0, M_x^0, \varepsilon)$ where M_x^0 is the initial state of $M(x)$ and ε is the empty proof.

Successor Circuit $\mathcal{S}(t, C, \pi)$

Hardwired: The public parameters pp, and the input x.
Algorithm:
1. If $\mathsf{IVC.V}(pp, t, C, \pi) = \mathsf{REJ}$, output v_0.
2. If $t = T$, output (t, C, π).
3. Compute $C' = M_x(C)$, $\pi' = \mathsf{IVC.P}(pp, t, C, \pi)$. Output $(t+1, C', \pi')$.

Value Circuit $\mathcal{F}(t, C, \pi)$

Hardwired: The public parameters pp.
Algorithm:
1. If $\mathsf{IVC.V}(pp, t, C, \pi) = \mathsf{REJ}$, output -1.
2. Else, output t.

Witness translation. The algorithm $\mathcal{W}(t, C, \pi)$ simply returns the content of the output tape in the configuration C.

Efficiency. The successor circuit M_x is of size $\text{poly}(S, \lambda, |M|)$. The computations related to IVC are of size $\text{poly}(\mathbb{T}_{\mathsf{IVC}})$ by Turing machine simulation. All other computations are polynomial in the input length ℓ. We have that $\ell = \text{poly}(S, \mathbb{T}_{\mathsf{IVC}})$. It follows that $\mathbb{T}_{\mathsf{Red}} = \text{poly}(\mathbb{T}_{\mathsf{IVC}}, S, \lambda, |M|)$. All the above polynomials don't depend on M nor IVC.

By the fact M is poly-space and the efficiency requirement of IVC, that is $\mathbb{T}_{\mathsf{IVC}} \leq \text{poly}(\lambda)$, both $\mathcal{X}, \mathcal{W}$ are polynomial-time algorithms, as required. In turn, this further implies that $\mathcal{S}, \mathcal{F}$ are of size $\text{poly}(\lambda)$. Since the length is a polynomial such that $\ell(\lambda) \geq \lambda$, we have that $\mathcal{S}, \mathcal{F}$ are polynomial-size circuits.

4.4 Security Analysis

Fix $\lambda \in \mathbb{N}$ and $x \in \{0,1\}^\lambda$. Let $(\mathcal{S}, \mathcal{F})$ be an instance in the support of $\mathcal{X}(x)$ and let pp be the public parameters hardwired in $(\mathcal{S}, \mathcal{F})$. We prove that all local maximums of $(\mathcal{S}, \mathcal{F})$ are one of two types:

- **Honest:** Nodes (T, M_x^T, π) where $\mathsf{IVC.V}(pp, t, M_x^T, \pi) = \mathsf{ACC}$.
- **Fooling:** Nodes (t, C^*, π^*) for which $C^* \neq M_x^t$ and $\mathsf{IVC.V}(pp, t, C^*, \pi^*) = \mathsf{ACC}$.

The proof of the following claim use the incremental completeness of IVC.

▷ Claim 12. All local maximums of $(\mathcal{S}, \mathcal{F})$ are honest-type or fooling-type.

Proof of claim. Let $v = (t, C, \pi)$ be a local maximum of $(\mathcal{S}, \mathcal{F})$, that is $\mathcal{F}(\mathcal{S}(v)) \leq \mathcal{F}(v)$. We have that $\mathsf{IVC.V}(pp, t, C, \pi) = \mathsf{ACC}$, as otherwise it is given value -1 and is connected to $v_0 = (0, M_x^0, \varepsilon)$ which is of value 0. If $C \neq M_x^t$ then v is a fooling-type local maximum and if $C = M_x^t$ and $t = T$ then v is an honest-type local maximum.

We are left with the case $C = M_x^t$ and $t < T$. By construction, $\mathcal{S}(v) = (t+1, M_x^{t+1}, \pi')$ for $\pi' = \mathsf{IVC.P}(pp, t, M_x^t, \pi)$. By incremental completeness, $\mathsf{IVC.V}(pp, t+1, M_x^{t+1}, \pi') = \mathsf{ACC}$ and thus $\mathcal{F}(\mathcal{S}(v)) = t+1$ while $\mathcal{F}(v) = t$. Therefore it is not a local maximum, proving the claim. ◁

Note that only honest-type local maximums translate by $\mathcal{W}$ to a valid witness for R, the underlying search problem. While there exist fooling-type local maximums, by the security of the IVC, finding them efficiently is infeasible. We proceed to prove Theorem 8.

Proof of Theorem 8. Let $\mathcal{A}$ be an efficient adversary that attempts breaking the computational soundness of the reduction $(\mathcal{X}, \mathcal{W})$. Let $\lambda \in \mathbb{N}$ and $x \in \{0,1\}^\lambda$ be such that R_x is non-empty. We denote by $\varepsilon = \varepsilon(x)$ the probability of success of $\mathcal{A}$. That is

$$\varepsilon(x) := \Pr\left[\begin{array}{l|l} \mathcal{F}(\mathcal{S}(w')) \leq \mathcal{F}(w') & (\mathcal{S}, \mathcal{F}) \leftarrow \mathcal{X}(x) \\ w \notin R_x & w' \leftarrow \mathcal{A}(\mathcal{S}, \mathcal{F}) \\ & w = \mathcal{W}(w') \end{array}\right] .$$

It follows by Claim 12 that all witnesses w' for $(\mathcal{S}, \mathcal{F})$ such that $\mathcal{W}(w') = w \notin R_x$ are fooling-type local maximums. Indeed, for every honest-type local maximum, W outputs the content of the output tape of M_x^T. As M solves R and R_x is non-empty, we have that the output tape of M_x^T, the last configuration, consists of w such that $w \in R_x$.

Consider $\mathcal{A}' = \{\mathcal{A}'_\lambda\}_{\lambda \in \mathbb{N}}$ that attempts breaking the soundness of the IVC defined as follows. Recall $I(x, pp)$ is the deterministic procedure used by $\mathcal{X}$ to construct the circuits $\mathcal{S}$ and $\mathcal{F}$.

$\mathcal{A}'(x, pp)$:

1. Compute $(\mathcal{S}, \mathcal{F}) = I(x, pp)$.
2. Simulate $w' \leftarrow \mathcal{A}(\mathcal{S}, \mathcal{F})$.
3. Output $w' = (t, C, \pi)$.

Note that, every fooling-type local maximum corresponds to (t, C^*, π^*) breaking the IVC scheme. Further notice that the distribution $(\mathcal{S}, \mathcal{F}) = I(x, pp)$ for $pp \leftarrow \mathsf{IVC.G}(x)$ is, by definition, the distribution of $\mathcal{X}(x)$. Therefore, the success probability of $\mathcal{A}'(x, pp)$ is at least $\varepsilon(x)$. Using the soundness of IVC, there exists a negligible function μ such that for every $\lambda \in \mathbb{N}$ and $x \in \{0,1\}^\lambda$ such that R_x is non-empty we have the following:

$$\varepsilon(x) \leq \Pr\left[\begin{array}{l|l} C^* \neq M_x^t & pp \leftarrow \mathsf{IVC.G}(x) \\ \mathsf{IVC.V}(pp, t, C^*, \pi^*) = \mathsf{ACC} & (t, C^*, \pi^*) \leftarrow \mathcal{A}'(x, pp) \end{array}\right] \leq \mu(\lambda) \ .$$

Establishing the computational soundness of the reduction. The efficiency part of the theorem has been argued in Section 4.3. ◀

4.5 Applying the Reduction

In this section we present results that establish that a computational Karp reduction from an underlying hard (worst-case/average-case) problem R to LS implies the hardness of LS (in worst-case/average-case respectively). The proofs for the following propositions are given in the full version of this paper.

4.5.1 Worst-Case Hardness

The existence of a computational Karp reduction from R to LS and an efficient solver for LS implies, in a straightforward way, the existence of an efficient randomized solver for R, successful with high probability. Proposition 13 extends this and show the existence of a deterministic solver for R. This is by standard non-uniform derandomization techniques. We use n to denote the size of LS instances and λ the size of R instances.

▶ **Proposition 13** (Worst-Case Hardness). *Let R be an FNP search problem having a computational Karp reduction to LS. Assume there exists a adversary $\mathcal{A} = \{\mathcal{A}_n\}_{n\in\mathbb{N}}$ of polynomial-size $s(n)$ solving LS in the worst-case. Then there exists an adversary $\mathcal{A}' = \{\mathcal{A}_\lambda\}_{\lambda\in\mathbb{N}}$ solving R in the worst-case. The size of $\mathcal{A}'$ is*

$$\mathsf{size}(\mathcal{A}') = \mathrm{poly}(s(\mathbb{T}_{\mathsf{Red}}), \mathbb{T}_R, \lambda) \ ,$$

where $\mathbb{T}_{\mathsf{Red}}(\lambda)$ is the efficiency of the reduction and $\mathbb{T}_R(\lambda)$ is the efficiency of the NP verification $R(x, y)$.

4.5.2 Average-Case Hardness

▶ **Proposition 14** (Average-Case Hardness). *If there exists a hard-on-average FNP problem R, with a computationally sound Karp reduction from R to LS, then LS is hard-on-average.*

Using the computational Karp reduction constructed in Section 4.3, we get the following corollary:

▶ **Corollary 15** (IVC Average-Case Hardness). *Let R be a hard-on-average FNP problem. If there exists an IVC scheme with incremental completeness for a polynomial-space Turing machine M solving R, then LS is hard-on-average.*

5 Instantiation under the KPY Assumption and ETH

In Section 4, we were able to prove that an IVC scheme with incremental completeness for some superpolynomially hard computation suffices to show that LS is hard. It is left to instantiate the above under the KPY assumption and construct an IVC scheme for a superpolynomially hard computation. Unfortunately, under the KPY assumption we are able to construct IVC schemes only for polynomial computations[4]. In this section, we show how to circumvent this difficulty by employing a fine-grained reduction and further assuming the non-uniform exponential time hypothesis (ETH). Finally proving the worst-case hardness of LS under the KPY assumption and non-uniform ETH.

▶ Remark 16. For ease of notation, in the following section we assume w.l.o.g $\lambda > 1$.

5.1 IVC From KPY Assumption

The proof of Theorem 18 is given in the full version of this paper.

▶ Remark 17 (Dependence of Constants). In the following, we use the constant c instead of big O notation in order to emphasize that the constant is independent of α.

▶ **Theorem 18** (IVC from KPY Assumption)**.** *Fix any constants $c, \varepsilon > 0$. Let α be a large enough constant and let M be a Turing machine of run-time $T(\lambda) \leq c\lambda^{\alpha}$, configuration size $S(\lambda) \leq c\lambda$ and description size $|M| \leq c \log_2 \alpha$. Under the KPY assumption, there exists an IVC scheme with incremental completeness for M having efficiency $\mathbb{T}_{\mathsf{IVC}} = \lambda^{\varepsilon\alpha}$.*

▶ Remark 19 (Non-Uniform Reduction). The security reduction of the IVC scheme given in Theorem 18 is non-uniform and therefore we require that the IVC scheme, and accordingly the KPY assumption, to hold against non-uniform polynomial-time attackers. This is to be expected when dealing with worst-case hardness.

5.2 Fixed Space Polynomial Hardness via ETH

In this section, assuming the non-uniform ETH assumption, we show the existence of arbitrarily polynomially hard computations for circuits that can be solved by Turing machines using a fixed amount of space and slightly larger polynomial time.

▶ **Assumption 20** (Non-Uniform ETH)**.** *There exists a constant $\delta > 0$ such that no circuit family of size $O(2^{\delta n})$ solves* 3SAT*, where n is the length of the input 3CNF formula.*

▶ **Proposition 21** (Polynomial Hardness)**.** *Under the non-uniform* ETH *assumption, there exists constants $c, \delta > 0$ such that for every constant $\alpha > 1$ there exists a search problem $R_\alpha \in$ FNP satisfying the following:*

- ***Algorithm:*** *There exists a Turing machine M_α of run-time $T(\lambda) \leq c\lambda^{\alpha}$, configuration size $S(\lambda) \leq c\lambda$ and description size $|M_\alpha| \leq c \log \alpha$ solving R_α.*
- ***Lower Bound:*** *No circuit family of size $O(\lambda^{\delta\alpha})$ can solve R_α.*

Proof. Consider

$$R_\alpha := \left\{ (x 0^{2^{|x|/\alpha} - |x|}, w) \;\middle|\; (x, w) \in \mathsf{3SAT} \right\} .$$

[4] This is for polynomial security of the KPY assumption. A super-polynomial security of the KPY assumption yields an IVC scheme for computations of a corresponding superpolynomial length.

Algorithm. The naive brute force algorithm for R_α satisfy the requirement. In the following α is not suppressed by O notation. For $x \in \{0,1\}^n$ and input of the form $x0^{2^{n/\alpha}-n}$, brute force takes $O(2^n)$ time while the whole input is of size $\lambda = 2^{n/\alpha}$. The description size is $O(\log \alpha)$ and the memory used is $O(\lambda)$.

Lower Bound. Let $\delta > 0$ be the constant assumed to exist by non-uniform ETH. Every circuit family of size $O(\lambda^{\delta\alpha})$ solving R_α can be turned to a circuit of size $O(2^{\delta\alpha})$ solving 3SAT. Contradicting non-uniform ETH. ◀

5.3 Putting Things Together

In this section, we put things together to prove that under non-uniform ETH and the KPY assumption we have PLS $\not\subseteq$ FP/Poly.

▶ **Theorem 22** (PLS Hardness). *Assuming non-uniform* ETH *and the KPY assumption we have* PLS $\not\subseteq$ FP/Poly.

Toward proving the theorem, we start by bounding the blowup associated with the IVC reduction when applied to the brute-force computation solving R_α defined by Proposition 21.

▶ **Lemma 23** (Bounding the Blowup). *Fix a constant $\varepsilon > 0$. Let α be a large enough constant. Under the KPY assumption, there exists a computationally sound Karp reduction $(\mathcal{X}, \mathcal{W})$ from R_α to* LS *with efficiency $\mathbb{T}_{\mathsf{Red}} \leq \lambda^{\varepsilon\alpha}$.*

Proof. Fix $\varepsilon > 0$ and let $\varepsilon' > 0$ be a constant to be chosen later. Let c be the constant guaranteed by Proposition 21. That is, for every $\alpha > 1$ there exists a Turing machine M_α of run-time $T \leq c\lambda^\alpha$, configuration size $S \leq c\lambda$ and description size $|M_\alpha| \leq c \log_2 \alpha$ solving R_α. Let α be a large enough constant such that Theorem 18 apply to M_α with constant ε'. This gives an IVC scheme with incremental completeness for M_α having efficiency $\mathbb{T}_{\mathsf{IVC}} \leq \lambda^{\varepsilon'\alpha}$.

Applying Theorem 8 using the aforementioned IVC results in a computationally sound Karp reduction $(\mathcal{X}, \mathcal{W})$ from R_α to LS with efficiency:

$$\mathbb{T}_{\mathsf{Red}} \leq (\mathbb{T}_{\mathsf{IVC}} \cdot S \cdot \lambda \cdot |M_\alpha|)^\eta \leq \lambda^{\varepsilon'\alpha\eta} \cdot (c\lambda \log_2 \alpha)^{3\eta} \leq \lambda^{4\varepsilon'\alpha\eta} \ ,$$

where η is a constant independent of α and ε and the last inequality is true for large α. By taking $\varepsilon' < \varepsilon/4\eta$ we get $\mathbb{T}_{\mathsf{Red}} \leq \lambda^{\varepsilon\alpha}$. This completes the proof the lemma. ◀

We are now ready to prove Theorem 22.

Proof of Theorem 22. Assume toward contradiction that under non-uniform ETH and the KPY assumption we have that PLS $\subseteq$ FP/Poly. Hence, there exists a constant η and adversary $\mathcal{A} = \{\mathcal{A}_n\}_{n\in\mathbb{N}}$ of size $s(n) = n^\eta$ solving LS instances of size n.

Let $\varepsilon > 0$ be a constant to be chosen later. By bounding the blowup lemma, Lemma 23, there exists a large enough constant α such that there is a computationally sound Karp reduction $(\mathcal{X}, \mathcal{W})$ from R_α to LS having efficiency $\mathbb{T}_{\mathsf{Red}} \leq \lambda^{\varepsilon\alpha}$. Note that the efficiency of the NP verifier for R_α is $\mathbb{T}_{\mathsf{R}_\alpha} = O(\lambda)$ with a constant independent of α. By Proposition 13, there exists an adversary $\mathcal{A}' = \{\mathcal{A}'_\lambda\}_{\lambda\in\mathbb{N}}$, solving R_α instances of size λ of circuit size:

$$\mathsf{size}(\mathcal{A}'_\lambda) \leq \mathrm{poly}\left(s(\mathbb{T}_{\mathsf{Red}}), \mathbb{T}_{\mathsf{R}_\alpha}, \lambda\right) \leq \mathrm{poly}\left(\lambda^{\varepsilon\alpha\eta}, \lambda\right) \underbrace{\leq}_{\alpha > 1/\varepsilon\eta} \mathrm{poly}\left(\lambda^{\varepsilon\alpha\eta}\right) = O(\lambda^{\varepsilon\alpha\xi}) \ ,$$

where ξ is a constant independent of ε and α. Let $\delta > 0$ be the constant associated with the lower bound of R_α. Recall δ is fixed and independent of α. By choosing $\varepsilon < \delta/\xi$ we have

$$\mathsf{size}(\mathcal{A}'_\lambda) = O(\lambda^{\varepsilon\alpha\xi}) = O(\lambda^{\delta\alpha}) \ .$$

A contradiction to the lower bound. We conclude that PLS $\not\subseteq$ FP/Poly. This completes the proof. ◀

6 Unconditional PLS Hardness in the ROM

In this section we show that relative to a random oracle, there exists exponentially average-case hard problems in PLS. We further show, relative to a random oracle, the existence of depth-robust moderately hard problems in PLS. Our result is based on the incremental proofs of sequential work (iPoSW) construction by Döttling, Lai, and Malavolta [20]. We modify the DLM construction such that an incremental completeness analogous is achieved. We proceed to reduce breaking the modified DLM scheme to LS relative to a random oracle.

6.1 Graph Related Definitions

We recall some graph related definitions used in the DLM scheme.

Notation. Throughout, we consider directed acyclic graphs (DAGs) $G = (V, E)$ where $V \subseteq \{0,1\}^{\leq d}$ is the vertex set and we refer to the parameter d as the depth of G. A vertex $v \in V$ is a parent of $u \in V$ if $(v, u) \in E$. The set of parents of u is denoted by $\mathsf{parents}(u)$. A vertex v is an ancestor of u if there is a directed path from v to u in G. The set of ancestors of u is denoted by $\mathsf{ancestors}(u)$. Any vertex $v \in V \cap \{0,1\}^n$ is called a leaf. Denote by $\mathsf{leaves}(u)$ the set of all leaf nodes that are ancestors of u.

6.1.1 CP Graphs

We recall the definition of CP graphs (see Figure 1), a family of directed acyclic graphs defined first in the PoSW construction of Cohen and Pietrzak [17]. We denote the aforementioned graph family by $CP = \{CP_d\}_{d\in\mathbb{N}}$.

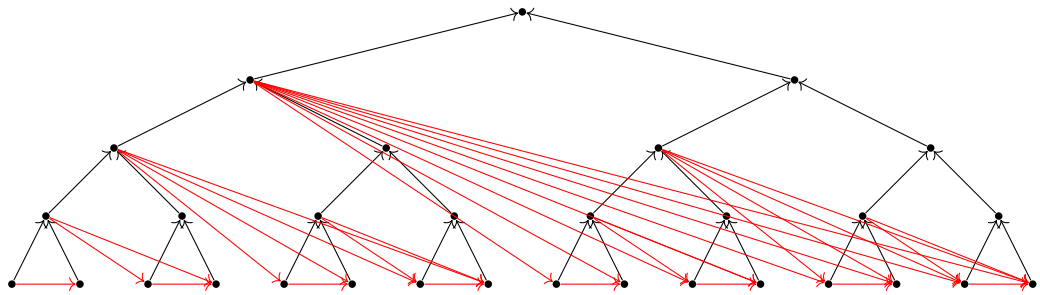


Figure 1 The CP_4 graph; red edges corresponds to E''.

▶ **Definition 24** (CP Graphs, [17]). *Let $B_d = (V, E')$ be the complete binary tree of depth d with edges pointing from the leaves to the root. The graph $CP_d = (V, E)$ is constructed from B_d with some added edges E''. Where E'' contains (v, u) for any leaf $u \in \{0,1\}^d$ and any v which is a left sibling to a node on the path form u to the root. Formally, $E = E' \cup E''$ where:*

$$E'' := \{(v, u) \mid u \in \{0,1\}^d, u = a||1||a', v = a||0, \text{ for some } a, a' \in \{0,1\}^{\leq d}\} \ .$$

The following fact on non-leaves in CP graphs is used in the construction and analysis.

▶ **Fact 25** (Left and Right Parents). *Let $CP_d = (V, E)$ and let $v \in V$ be a non-leaf node which is not the root. The node v has exactly two parents ℓ and r. We have the following:*

$$\mathsf{leaves}(v) = \mathsf{leaves}(\ell) \cup \mathsf{leaves}(r) \ .$$

6.1.2 Graph Labeling

▶ **Definition 26** (Partial Vertex Labeling). *For a graph $G = (V, E)$, a partial vertex labeling is a function $\mathcal{L} : V \to \{0,1\}^* \cup \{\perp\}$. The label for node $u \in V$ is denoted by $\mathcal{L}[u]$.*

▶ **Definition 27** (Valid DAG Labeling). *Let $G = (V, E)$ be a DAG with $V \subseteq \{0,1\}^*$ and let $f : \{0,1\}^* \to \{0,1\}^w$ be a labeling function. The labeling of G with respect to f is defined recursively, for every $v \in V$:*

$$\mathcal{L}[v] := f(v, \mathcal{L}[p_1], ..., \mathcal{L}[p_r]) \ ,$$

where $(p_1, ..., p_r) = \mathsf{parents}(v)$, ordered in lexicographic order.

Note that the above labeling is well defined since G is a acyclic and can be computed with $|V|$ queries to f in a topological order.

In the following lemma we recall the CP labeling procedure given in [17]. See Figure 2 for an illustration of the procedure.

▶ **Lemma 28** (CP Labeling Procedure, Lemma 3 in [17]). *The labeling of CP_d with respect to $f : \{0,1\}^* \to \{0,1\}^w$ can be computed in a topologicial ordering using $O(w \cdot d)$ bits of memory.*

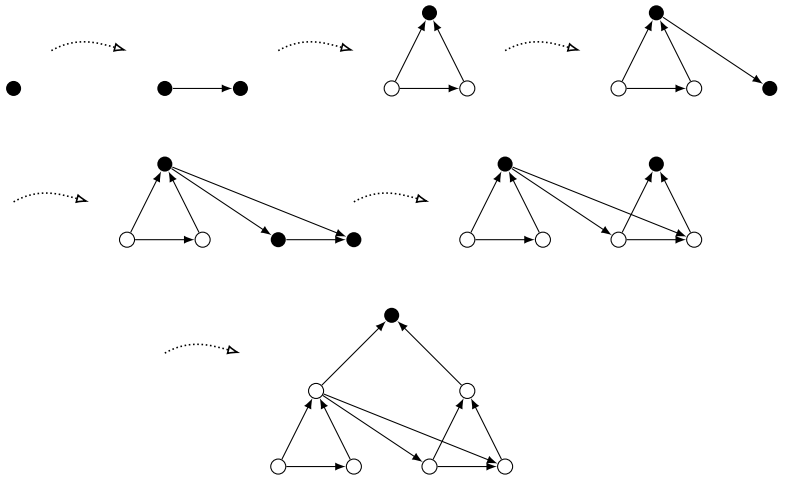


Figure 2 The CP labeling procedure for CP_2; black nodes represent labels the algorithm keep track of; denoted $\mathcal{U}_t$ when computing the t label.

We require additional properties from the labeling procedure for the modified DLM scheme. For that purpose, we give a description of the procedure bellow and prove additional lemmas regarding it.

Labeling Procedure Description. The algorithm is recursive. The base case CP_1 is easy. For $d > 1$, let v be the root of CP_d and let ℓ, r be the left and right parents of v, respectively. The subtree rooted by ℓ is isomorphic to CP_{d-1} and hence its labels can be computed recursively. Keep track of the last label $\mathcal{L}[\ell]$ and proceed to the subtree rooted by r. Apart from edges coming from ℓ to the leaves, it is isomorphic to CP_{d-1}. One can use $\mathcal{L}[\ell]$, which is in memory, and recursively compute the right subtree. Keep track of the last label $\mathcal{L}[r]$. Compute the root label $\mathcal{L}[v] = f(v, \mathcal{L}[\ell], \mathcal{L}[r])$ and forget both $\mathcal{L}[\ell]$ and $\mathcal{L}[r]$. Output the root label $\mathcal{L}[v]$.

Incremental Labeling. We consider an incremental version of the above procedure. Fix $d \in \mathbb{N}$ and let $v_1, v_2, \ldots, v_T$ be the topological ordering in which labels are being outputted by the CP labeling procedure with depth d. Let $\mathcal{U}_t \subseteq V$ be the set of nodes the procedure keep track of after outputting v_t. For the incremental version, we are interested in algorithms that efficiently compute $\mathcal{U}_t$ and v_t given t.

▶ **Lemma 29** (Computing $\mathcal{U}_t$ Efficiently). *For CP_d, computing $\mathcal{U}_t$ can be done in* poly(d) *time.*

Proof. The algorithm is recursive and follows the same post-order traversal as the labeling procedure. For CP_d, let v be the root node and let ℓ, r be its left and right parents respectively. If $t = 2^{d+1} - 1$ return the root node, $\{v\}$. If $t \leq 2^d - 1$, proceed to recursively compute $\mathcal{U}_t$ on the left subtree rooted by ℓ, which is isomorphic to CP_{d-1}. Output the resulting set. Else $t > 2^d - 1$, recursively compute $\mathcal{U}_{t-(2^d-1)}$ on the subtree rooted by r while ignoring the extra edges from ℓ. This graph is also isomorphic to CP_{d-1}. Append ℓ to the resulting set and output.

The algorithm correctness follows by induction. For efficiency, notice that the depth of the recursion is d and every level takes poly(d) time. ◀

▶ **Lemma 30** (Computing v_t Efficiently). *For CP_d, computing v_t can be done in* poly(d) *time.*

Proof. Notice that by construction of the CP labeling procedure, $v_1, \ldots, v_T$ is a post-order traversal on the binary tree B_d which is the base of CP_d. Finding the t-th node in a post-order traversal can be done by a simple recursion with efficiency poly(d). ◀

▶ **Lemma 31** (One Vertex at a Time). *For every $t \in [2, T]$ we have $\mathcal{U}_t = \{v_t\} \biguplus (\mathcal{U}_t \cap \mathcal{U}_{t-1})$.*

Proof. The theorem statement is equivalent to proving $\mathcal{U}_t \backslash \mathcal{U}_{t-1} = \{v_t\}$. By definition we have that $v_t \in \mathcal{U}_t$. Apart from v_t, the procedure only forget nodes, hence $\mathcal{U}_t \subseteq \{v_t\} \cup \mathcal{U}_{t-1}$. It is left to show that $v_t \notin \mathcal{U}_{t-1}$. Indeed, using the aforementioned, by induction $\mathcal{U}_{t-1} \subseteq \{v_1, \ldots, v_{t-1}\}$, in particular $v_t \notin \mathcal{U}_{t-1}$. ◀

6.2 Construction

In this section, we present a modified version of the incremental proof of sequential work (iPoSW) due to [20]. The main difference, apart from syntax, is that we require stricter conditions when verifying nodes.

Parameters. Let λ be a computational security parameter. Let $T(\lambda) = T = 2^{\lambda+1} - 1$ and let $s(\lambda) = s = \Theta(\lambda^3)$ that is a power of 2. Let $\mathsf{H} : \{0,1\}^* \to \{0,1\}^\lambda$ and $\mathsf{H}' : \{0,1\}^* \to \{0,1\}^{3s}$ be independently chosen random oracle ensembles depending on λ.

▶ Remark 32 (Random Coins Count). The use of $3s$ random coins allows to sample a statistically close to uniform subset. See [20] for details.

Syntax. The scheme consists of two algorithms, $\mathsf{DLM.P}^{\mathsf{H},\mathsf{H}'}(\chi, t, \pi), \mathsf{DLM.V}^{\mathsf{H},\mathsf{H}'}(\chi, t, \pi)$ with the following syntax.

1. $\mathsf{DLM.P}^{\mathsf{H},\mathsf{H}'}(\chi, t, \pi)$ is a deterministic algorithm with oracle access to H, H'. Given a statement χ, a natural number t and a proof π, it outputs a proof π'.
2. $\mathsf{DLM.V}^{\mathsf{H},\mathsf{H}'}(\chi, t, \pi)$ is a deterministic algorithm with oracle access to H, H'. Given a statement χ, a natural number t and a proof π, it outputs ACC or REJ.

Proof Structure. The proofs π are of the form $\pi = (\mathcal{L}, (u_i, S_{u_i}, \mathcal{P}_{u_i})_{i=1}^m)$ where:

- $\mathcal{L}$ is a partial vertex labeling for CP_λ.
- u_i is a node of CP_λ.
- S_{u_i} is a set of challenge leaves, indexed by the node u_i.
- $\mathcal{P}_{u_i}$ is a set of candidate authentication paths, indexed by the node u_i.
- $\mathsf{Path} \in \mathcal{P}_{u_i}$ is an ordered list of the form $\mathsf{Path} = (w_j)_j$ where each w_j is a node of CP_λ.

▶ Remark 33 (Labeling Data Structure). The partial vertex labeling $\mathcal{L}$ is given by a dictionary data structure. Where keys are nodes v and the value associated is the corresponding label $\mathcal{L}[v]$. A node which is not in the dictionary is given the default value $\perp$. Let n be the amount of nodes v for which $\mathcal{L}[v] \neq \perp$. We use a dictionary implementation where size and retrieval time are poly(n).

6.2.1 Algorithms Description

Conventions. A random oracle salted with the statement $\chi \in \{0,1\}^*$ is denoted by $\mathsf{H}_\chi := \mathsf{H}(\chi, \cdot)$. Let $v_1, v_2, \ldots, v_T$ be the topological order given by Lemma 28 for nodes of CP_λ. Recall that $\mathcal{U}_t$ is the list of nodes the CP labeling procedure keeps track of when computing the label for v_t. We extend the definition such that $\mathcal{U}_0 = \emptyset$.

$\mathsf{DLM.P}^{\mathsf{H},\mathsf{H}'}(\chi, t, \pi)$

Input:

1. String $\chi \in \{0,1\}^\lambda$.
2. Time $t \in [T]$.
3. Candidate proof $\pi = (\mathcal{L}, (u_i, S_{u_i}, \mathcal{P}_{u_i})_{i=1}^m)$.

Algorithm:

1. If $\{u_1, \ldots, u_m\} \neq \mathcal{U}_{t-1}$ output ε.
2. Set $\mathcal{L}[v_t] = \mathsf{H}_\chi(v_t, \mathcal{L}[p_1], ..., \mathcal{L}[p_r])$ where $(p_1, ..., p_r) = \mathsf{parents}(v_t)$ in lexicographic order.
3. If v_t is a leaf, let $S_{v_t} = \{v_t\}$ and $\mathcal{P}_{v_t} = \{(v_t)\}$.
4. Otherwise, v_t has two parents, $\ell, r \in \mathcal{U}_{t-1}$.
   a. Sample a random subset S_{v_t} of size $\min(s, |\mathsf{leaves}(v_t)|)$ from $S_\ell \cup S_r$ using $\mathsf{rand} \leftarrow \mathsf{H}'_\chi(v_t, \mathcal{L}[v_t])$ as random coins.
   b. Let $\mathcal{P}_{v_t} = \emptyset$. For every $\mathsf{Path} \in \mathcal{P}_\ell \cup \mathcal{P}_r$ starting from a leaf in S_{v_t}, extend with v_t and append to $\mathcal{P}_{v_t}$.
5. Remove labels from $\mathcal{L}$ for all nodes but $\left\{ v \;\middle|\; \begin{array}{l} u \in \mathcal{U}_t \\ \mathsf{Path} \in \mathcal{P}_u \\ w \in \mathsf{Path} \\ v = w \text{ or } v \in \mathsf{parents}(w) \end{array} \right\}$.
6. Output $\pi' = (\mathcal{L}, (u, S_u, \mathcal{P}_u)_{u \in \mathcal{U}_t})$.

Note that in the above we use the efficient algorithms due to computing $\mathcal{U}_t$ efficiently Lemma 29 and computing v_t efficiently Lemma 30 to compute $\mathcal{U}_{t-1}, \mathcal{U}_t$ and v_t. We also use one vertex at a time Lemma 31 in Line 5 and Line 6.

$\mathsf{DLM.V}^{\mathsf{H},\mathsf{H}'}(\chi, t, \pi)$

Input:

1. String $\chi \in \{0,1\}^\lambda$.
2. Time $t \in [T]$.
3. Candidate proof $\pi = (\mathcal{L}, (u_i, S_{u_i}, \mathcal{P}_{u_i})_{i=1}^m)$.

Algorithm:

1. Verify that $\{u_1, \ldots, u_m\} = \mathcal{U}_{t-1}$ and distinct. Otherwise, output REJ.
2. Verify that the nodes having label in $\mathcal{L}$ are $\left\{ v \;\middle|\; \begin{array}{l} u \in \mathcal{U}_{t-1} \\ \mathsf{Path} \in \mathcal{P}_u \\ w \in \mathsf{Path} \\ v = w \text{ or } v \in \mathsf{parents}(w) \end{array} \right\}$.
   Otherwise output REJ.
3. For every $i = 1 \ldots m$:
   a. Validate $S_{u_i} \subseteq \mathsf{leaves}(u_i)$ and is of size $\min(s, |\mathsf{leaves}(u_i)|)$. Otherwise output REJ.
   b. Check that every path in $\mathcal{P}_{u_i}$ is a valid path from a leaf in S_{u_i} to u_i. Check that every leaf in S_{u_i} has exactly one corresponding path in $\mathcal{P}_{u_i}$. Otherwise output REJ.
   c. Let $(\mathsf{Path}_1, \ldots, \mathsf{Path}_k) = \mathcal{P}_{u_i}$ ordered in lexicographic order of corresponding source leaves.
   d. For $j = 1, \ldots, k$:
      i. If $\mathsf{VerifyMerklePath}^{\mathsf{H}}(\mathsf{Path}_j, \mathcal{L}, \chi)$ rejects, output REJ.
      ii. If $\mathsf{VerifyRandChoice}^{\mathsf{H}'}(\mathsf{Path}_j, \mathcal{L}, j, \chi)$ rejects, output REJ.
4. Output ACC.

In the above description we used the subroutines $\mathsf{VerifyAuthPath}$ and $\mathsf{VerifyRandChoice}^{\mathsf{H}'}$ that are described in the following.

Merkle Authentication Path Verification $\mathsf{VerifyAuthPath}^{\mathsf{H}}(\mathsf{Path}, \mathcal{L}, \chi)$. Let $\mathsf{Path} = (w_j)_j$ be a candidate authentication path. For every node w_j with $(p_1, \ldots, p_r) = \mathsf{parents}(w_j)$ ordered in lexicographic order, verify

$$\mathcal{L}[w_j] = \mathsf{H}_\chi(w_j, \mathcal{L}[p_1], \ldots, \mathcal{L}[p_r]) \ .$$

If all checks pass, output ACC, otherwise output REJ.

Random Choice Verification $\mathsf{VerifyRandChoice}^{\mathsf{H}'}(\mathsf{Path}, \mathcal{L}, \mathsf{ind}, \chi)$. We give a sketch of the verification, for further details see [20]. Let $\mathsf{Path} = w_1 \to \cdots \to w_r$ be a path where $w_1 \in \mathsf{leaves}(w_r)$. In order to verify the random choice done by H'_χ, we recreate the choice going from the top down. We are given ind, the index of Path in $\mathcal{P}$ ordered by source leaves. This is also the index of the leaf that was picked in the last random choice.

Note that since s is a power of 2, the random choice procedure we use in the prover either choose an s subset from $2s$ set or take the whole set. Starting from w_r, use H'_χ to generate the random coins rand. Using rand pick an s subset from the set $[2s]$. Let i be the ind-th element in the resulting subset. If $i \le s$, we know that w_{r-1} should be a left parent to w_r. Otherwise $i > s$ which tells us that w_{r-1} should be a right parent. We continue to w_{r-1} with $\mathsf{ind} = i$ if $i \le s$ and $\mathsf{ind} = i - s$ otherwise. We continue with this procedure until we get to a node with less than s leaves. If all verifications of the path passed, output ACC and otherwise output REJ.

Efficiency. All the computations done above are polynomial in the input length that is of size $\text{poly}(\lambda, |\pi|)$. Notice that due to the restriction on the size of $\mathcal{L}$ (Line 2 of DLM.V), the size of all accepted proofs is $\text{poly}(\lambda)$. Therefore, for all accepted proofs the runtime is $\text{poly}(\lambda)$ for a fixed polynomial. Hence, we may modify DLM.V to reject and DLM.P to output ε once this $\text{poly}(\lambda)$ time limit exceeds. This way, the efficiency of DLM.V and DLM.P is $\text{poly}(\lambda)$ for every input.

▶ Remark 34 (Proofs are Not Unique). Proofs in the DLM scheme are not computationally unique. To see this, consider an adversary that attempts to find a collision for accepting proofs of time $t = \Theta(s^2) = \Theta(\lambda^6)$ that is of the form $2^{k+1} - 1$. The first proof is generated honestly. For the second proof, select randomly $u \in \mathsf{leaves}(v_t)$. Compute the proof honestly up until the label for u is computed. Alter the label for u. Continue as the honestly generated proof. With overwhelming probability, the two proofs generated are different. The second proof is accepting if there is no authentication path starting with u. Since t is of the form $2^{k+1} - 1$, there are s authentication paths starting with leaves in $\mathsf{leaves}(v_t)$. Also, all the leaves have the same probability to be chosen. There are 2^k leaves, therefore the probability u is selected is $s/2^k = \Omega(1/\lambda^3)$. The described adversary is efficient and successful with noticeable probability.

6.2.2 Incremental Completeness

The proof for the next proposition is in the full version of this paper.

▶ **Proposition 35** (Incremental Completeness of Modified DLM). *For every security parameter $\lambda \in \mathbb{N}$, time $t \in [T-1]$, candidate proof $\pi \in \{0,1\}^*$ and statement $\chi \in \{0,1\}^\lambda$:*

$$\textit{If } \mathsf{DLM.V}^{\mathsf{H},\mathsf{H}'}(\chi, t, \pi) = \mathsf{ACC} \textit{ then } \mathsf{DLM.V}^{\mathsf{H},\mathsf{H}'}(\chi, t+1, \pi') = \mathsf{ACC} \ ,$$

where $\pi' = \mathsf{DLM.P}^{\mathsf{H},\mathsf{H}'}(\chi, t, \pi)$.

6.2.3 Soundness

In this section, we state the soundness property of the modified DLM system. While we altered the DLM construction, we did so by enforcing stricter verification conditions. Accordingly, we inherit the soundness of the original scheme – every valid proof in the modified scheme corresponds to a valid proof in the original scheme without the use of the random oracle in the correspondence.

▶ **Theorem 36** (DLM Soundness, Follows From [20]). *Let $\mathcal{A}^{\mathsf{H},\mathsf{H}'} = \{\mathcal{A}_\lambda^{\mathsf{H},\mathsf{H}'}\}$ be an oracle aided adversary that makes at most $q = q(\lambda)$ total queries to both H and H'. For every $\lambda, d \in \mathbb{N}$ with $d \leq \lambda$, such that $\mathcal{A}$ makes less than 2^d sequential queries to H we have:*

$$\Pr\left[\mathsf{DLM.V}^{\mathsf{H},\mathsf{H}'}(\chi, T_d, \pi) = \mathsf{ACC} \;\middle|\; \begin{array}{l} \chi \leftarrow \{0,1\}^\lambda \\ \pi \leftarrow \mathcal{A}^{\mathsf{H},\mathsf{H}'}(\chi) \end{array}\right] \leq O(q^2) 2^{-\Omega(\lambda)} \ ,$$

where $T_d = 2^{d+1} - 1$.

▶ **Corollary 37** (Sequential Hardness). *Let $d = d(\lambda) \leq \lambda$ be a depth parameter. For every oracle aided adversary $\mathcal{A}^{\mathsf{H},\mathsf{H}'} = \{\mathcal{A}_\lambda^{\mathsf{H},\mathsf{H}'}\}_{\lambda \in \mathbb{N}}$ of depth 2^d and size $2^{o(\lambda)}$, we have for every $\lambda \in \mathbb{N}$:*

$$\Pr\left[\mathsf{DLM.V}^{\mathsf{H},\mathsf{H}'}(\chi, T_d, \pi) = \mathsf{ACC} \;\middle|\; \begin{array}{l} \chi \leftarrow \{0,1\}^\lambda \\ \pi \leftarrow \mathcal{A}^{\mathsf{H},\mathsf{H}'}(\chi) \end{array}\right] \leq 2^{-\Omega(\lambda)} \ ,$$

where $T_d = 2^{d+1} - 1$.

6.2.3.1 Single Oracle

The DLM construction is described in terms of two random oracles H, H' (which is done mostly to simplify the analysis). From hereon, it will be simpler to think of a single random oracle $\mathcal{O} : \{0,1\}^* \to \{0,1\}$. The algorithms $\mathsf{DLM.P}^{\mathcal{O}}, \mathsf{DLM.V}^{\mathcal{O}}$ simulate oracle calls to H, H' in the above construction by setting an appropriate prefix for every output bit in H and H'. The security reduction incurs a multiplicative polynomial loss in efficiency and a multiplicative logarithmic loss in depth (both in terms of λ).

Indeed, consider an adversary $\mathcal{A}$ attempting to break the scheme with $\mathcal{O}$. In order reduce this adversary to $\mathcal{A}'$ attacking the scheme using H, H' we modify the oracle gates to $\mathcal{O}$ in $\mathcal{A}$ by examining the prefix and simulating the output of the associate output bit in H or H'. In this reduction, comparing the prefix, incurs a logarithmic loss in depth and polynomial loss in size of the new adversary $\mathcal{A}'$. We conclude this in the following corollary.

▶ **Corollary 38** (Single Oracle Sequential Hardness). *Let $d = d(\lambda) \leq \lambda$ be a depth parameter. For every oracle aided adversary $\mathcal{A}^{\mathcal{O}} = \{\mathcal{A}^{\mathcal{O}}_\lambda\}_{\lambda\in\mathbb{N}}$ of depth $o(2^d/\log\lambda)$ and size $2^{o(\lambda)}$, we have for every $\lambda \in \mathbb{N}$:*

$$\Pr\left[\mathsf{DLM.V}^{\mathcal{O}}(\chi, T_d, \pi) = \mathsf{ACC} \;\middle|\; \begin{array}{c}\chi \leftarrow \{0,1\}^\lambda \\ \pi \leftarrow \mathcal{A}^{\mathcal{O}}(\chi)\end{array}\right] \leq 2^{-\Omega(\lambda)} \quad ,$$

where $T_d = 2^{d+1} - 1$.

By considering $d = \lambda$, we derive the following corollary:

▶ **Corollary 39** (Exponential Hardness). *For every oracle aided adversary $\mathcal{A}^{\mathcal{O}} = \{\mathcal{A}^{\mathcal{O}}_\lambda\}_{\lambda\in\mathbb{N}}$ of size $2^{o(\lambda)}$ and $\lambda \in \mathbb{N}$:*

$$\Pr\left[\mathsf{DLM.V}^{\mathcal{O}}(\chi, T_\lambda, \pi) = \mathsf{ACC} \;\middle|\; \begin{array}{c}\chi \leftarrow \{0,1\}^\lambda \\ \pi \leftarrow \mathcal{A}^{\mathcal{O}}(\chi)\end{array}\right] \leq 2^{-\Omega(\lambda)} \quad ,$$

where $T_\lambda = 2^{\lambda+1} - 1$.

6.3 Hard instances

In this section, we consider the problem $R^{\mathcal{O}}_d$, where instances are strings χ and a witness for χ is a corresponding proof of sequential work in the modified DLM scheme for time $T_d = 2^{d+1} - 1$. We show that $R^{\mathcal{O}}_d$ lies in $\mathsf{PLS}^{\mathcal{O}}$.

▶ Remark 40 (Efficiently Computable Depth Parameter). In the following, a depth parameter $d(\lambda)$ is a polynomial-time computable function. We require this for the efficiency of the reduction.

▶ **Definition 41.** *Let $d = d(\lambda) \leq \lambda$ be a depth parameter. The search problem $R^{\mathcal{O}}_d$ is defined as follows:*

$$R^{\mathcal{O}}_d = \left\{(\chi, \pi) \;\middle|\; \mathsf{DLM.V}^{\mathcal{O}}(\chi, T_d(|\chi|), \pi) = \mathsf{ACC}\right\} \quad ,$$

where $T_d(\lambda) = 2^{d(\lambda)+1} - 1$.

Note that $R^{\mathcal{O}}_d$ is in $\mathsf{TFNP}^{\mathcal{O}}$. It is in $\mathsf{FNP}^{\mathcal{O}}$ by the efficiency of DLM. It is total by the completeness of DLM, since for every χ, there exists a proof π that passes verification – the honestly generated proof.

6.3.1 The Reduction to Local Search

Fix a depth parameter $d = d(\lambda) \leq \lambda$. We construct a polynomial-time search reduction (f_d, g_d) from $R_d^{\mathcal{O}}$ to $\mathsf{LS}^{\mathcal{O}}$.

Instance Translation. Let $\chi \in \{0,1\}^\lambda$ be an instance for the problem $R_d^{\mathcal{O}}$. We construct an $\mathsf{LS}^{\mathcal{O}}$ instance $f_d(\chi) = (\mathcal{S}^{\mathcal{O}}, \mathcal{F}^{\mathcal{O}})$ in the following. Intuitively, we embed the computation of the modified DLM PoSW in an $\mathsf{LS}^{\mathcal{O}}$ instance such that local maximums correspond to valid proofs for time $T_d = 2^{d+1} - 1$. Concretely, every node will be of a fixed polynomial length $\ell = \ell(\lambda) \geq \lambda$. Every vertex is of the form $(t, \pi) \in \{0,1\}^\ell$ where t is parsed as an integer $t \in [T_d]$ and π as a proof of the modified DLM scheme with padding to length ℓ if needed.

Let π_1 be a valid proof for time $t = 1$ in the modified DLM scheme. For the node (t, π), if the proof π is verified with respect to time t, the node will be given value t and be connected to $(t+1, \pi')$ where π' is the proof the prover generates given π. If π is rejected, it is given value -1 and is connected to $(1, \pi_1)$. A formal definition of the circuits is given bellow.

Successor Circuit $\mathcal{S}^{\mathcal{O}}(t, \pi)$

Hardwired: The instance χ.
Algorithm:
1. If $\mathsf{DLM.V}^{\mathcal{O}}(\chi, t, \pi) = \mathsf{REJ}$, output $(1, \pi_1)$.
2. If $t = T_d$, output (t, π).
3. Compute $\pi' = \mathsf{DLM.P}^{\mathcal{O}}(\chi, t, \pi)$ and output $(t+1, \pi')$.

Value Circuit $\mathcal{F}^{\mathcal{O}}(t, \pi)$

Hardwired: The instance χ.
Algorithm:
1. If $\mathsf{DLM.V}^{\mathcal{O}}(\chi, t, \pi) = \mathsf{REJ}$, output -1.
2. Else, output t.

Witness Translation. The function $g_d(t, \pi)$ outputs π.

Efficiency. By the efficiency of the modified DLM scheme and the efficiency of the depth parameter $d(\lambda)$, the instance translation f_d is done in $\mathrm{poly}(\lambda)$ time. The witness translation g_d is also done in $\mathrm{poly}(\lambda)$ time since the length of nodes ℓ is of size $\mathrm{poly}(\lambda)$. This further implies that $\mathcal{S}^{\mathcal{O}}, \mathcal{F}^{\mathcal{O}}$ are of size $\mathrm{poly}(\lambda)$. Since the input length is a polynomial such that $\ell(\lambda) \geq \lambda$, this implies that $\mathcal{S}^{\mathcal{O}}, \mathcal{F}^{\mathcal{O}}$ are polynomial-size circuits.

6.3.2 Security Analysis

In the following claim we rely on the incremental completeness of the modified DLM Section 6.2.2.

▷ Claim 42. Let $d = d(\lambda) \leq \lambda$ be a depth parameter. Fix $\lambda \in \mathbb{N}$ and $\chi \in \{0,1\}^\lambda$. Let $w = (t, \pi)$ be a local maximum of $(\mathcal{S}^{\mathcal{O}}, \mathcal{F}^{\mathcal{O}}) = f_d(\chi)$. We have $\mathsf{DLM.V}^{\mathcal{O}}(\chi, t, \pi) = \mathsf{ACC}$ and $t = T_d$.

Proof. We have $\mathsf{DLM.V}^{\mathcal{O}}(\chi, t, \pi) = \mathsf{ACC}$ as otherwise w is given value -1 and his successor is $(1, \pi_1)$ of value 1. If $t < T_d$ then w is given value t and is connected to $(t+1, \pi')$ for $\pi' = \mathsf{DLM.P}^{\mathcal{O}}(\chi, t, \pi)$. By incremental completeness Proposition 35, we have that $\mathsf{DLM.P}(\chi, t+1, \pi') = \mathsf{ACC}$. Therefore, the value of $(t+1, \pi')$ is $t+1$, and w is not a local maximum. We are left with $t = T_d$. This concludes the claim. ◁

▶ **Theorem 43** (Hard Problems in $\mathsf{PLS}^{\mathcal{O}}$). *For every depth parameter $d = d(\lambda) \leq \lambda$, the search problem $R_d^{\mathcal{O}}$ lies in $\mathsf{PLS}^{\mathcal{O}}$.*

Proof. The class $\mathsf{PLS}^{\mathcal{O}}$ consists of all $\mathsf{TFNP}^{\mathcal{O}}$ search problems that are polynomial-time reducible to $\mathsf{LS}^{\mathcal{O}}$. The search problem $R_d^{\mathcal{O}} \in \mathsf{TFNP}^{\mathcal{O}}$ and by Claim 42, the tuple (f_d, g_d) is a valid polynomial-time reduction to $\mathsf{PLS}^{\mathcal{O}}$. Indeed, for every string χ, we have that all witnesses w' for the instance $(\mathcal{S}^{\mathcal{O}}, \mathcal{F}^{\mathcal{O}}) = f_d(\chi)$ are mapped under g_d to $w = \pi$ such that $\mathsf{DLM.V}^{\mathcal{O}}(\chi, T_d, \pi) = \mathsf{ACC}$ – a valid witness to $R_d^{\mathcal{O}}$. Hence $R_d^{\mathcal{O}} \in \mathsf{PLS}^{\mathcal{O}}$. A proof is given in the full version. ◀

▶ **Corollary 44** (Exponential Hardness in $\mathsf{PLS}^{\mathcal{O}}$). *There exists search problems in $\mathsf{PLS}^{\mathcal{O}}$ that are average-case exponentially hard.*

Proof. By Theorem 43, for every depth parameter $d = d(\lambda) \leq \lambda$, the search problem $R_d^{\mathcal{O}}$ is in $\mathsf{PLS}^{\mathcal{O}}$. By Corollary 39, for $d = \lambda$ the search problem $R_\lambda^{\mathcal{O}}$ is exponentially average-case hard. ◀

This implies that $\mathsf{LS}^{\mathcal{O}}$ is sub-exponentially hard due to the polynomial blowup the reduction incurs. We formulate it in the following corollary. In the following, n stands for the size of the $\mathsf{LS}^{\mathcal{O}}$ instance $(\mathcal{S}^{\mathcal{O}}, \mathcal{F}^{\mathcal{O}})$ and λ stands for the security parameter that is given as input to the sampler. A proof is given in the full version of this paper.

▶ **Corollary 45.** *There exists an efficient sampler HARD of $\mathsf{LS}^{\mathcal{O}}$ instances and a constant $\delta > 0$ such that for every oracle aided adversary $\mathcal{A}^{\mathcal{O}} = \{\mathcal{A}_n^{\mathcal{O}}\}_{n \in \mathbb{N}}$ of size at most 2^{n^δ} and for every $\lambda \in \mathbb{N}$:*

$$\Pr\left[\mathcal{F}^{\mathcal{O}}(\mathcal{S}^{\mathcal{O}}(w)) \leq \mathcal{F}^{\mathcal{O}}(w) \;\middle|\; \begin{array}{l}(\mathcal{S}^{\mathcal{O}}, \mathcal{F}^{\mathcal{O}}) \leftarrow \mathsf{HARD}(1^\lambda) \\ w \leftarrow \mathcal{A}(\mathcal{S}^{\mathcal{O}}, \mathcal{F}^{\mathcal{O}})\end{array}\right] \leq 2^{-\Omega(\lambda)} \quad ,$$

where $\mathcal{O} : \{0,1\}^* \to \{0,1\}$ is a random oracle.

6.4 Depth Robust Instances

In this section, we deduce the existence of depth-robust, moderately hard, search problems in $\mathsf{PLS}^{\mathcal{O}}$. That is, search problems with instances that can be solved in polynomial time yet also require high sequential time.

▶ **Proposition 46** (Depth Robust Problems in $\mathsf{PLS}^{\mathcal{O}}$). *Fix any constant $\varepsilon > 0$. For any large enough constant α, consider $R_d^{\mathcal{O}}$ for $d(\lambda) = \alpha \log_2 \lambda$ a depth parameter. The following properties hold for $\lambda > 1$:*

- ***Moderately Hard:*** *The search problem $R_d^{\mathcal{O}}$ can be solved in time $\lambda^{(1+\varepsilon)\alpha}$.*
- ***Depth Robust:*** *If $\mathcal{A}$ is an adversary of depth $\lambda^{(1-\varepsilon)\alpha}$ and size $2^{o(\lambda)}$ then:*

$$\Pr\left[(\chi, \pi) \in R_d^{\mathcal{O}} \;\middle|\; \begin{array}{l}\chi \leftarrow \{0,1\}^\lambda \\ \pi \leftarrow \mathcal{A}^{\mathcal{O}}(\chi)\end{array}\right] \leq 2^{-\Omega(\lambda)} \quad .$$

Proof. Fix $\varepsilon > 0$ throughout the proof. We start by proving that $R_d^{\mathcal{O}}$ is moderately hard.

▷ Claim 47 (Moderately Hard). The search problem $R_d^{\mathcal{O}}$, where $d(\lambda) = \alpha \log_2 \lambda$, can be solved in time $\lambda^{(1+\varepsilon)\alpha}$ when α is a large enough constant.

Proof. Fix $\lambda > 1$ and let $\chi \in \{0,1\}^\lambda$ be an instance of $R_d^{\mathcal{O}}$. We have $T_d = 2^{\alpha \log_2 \lambda + 1} - 1 \leq 2\lambda^\alpha \leq \lambda^{\alpha+1}$. By efficiency of DLM, the run-time of $\mathsf{DLM.P}^{\mathcal{O}}$ is at most λ^c for some fixed constant c, independent of ε and α. The proof π_1 for time $t = 1$ can also be generated in poly(λ) time. We assume w.l.o.g that it can be generated in time λ^c. By completeness of DLM, generating π_1 and applying the prover $T_d - 1$ times on it, results in a valid proof for time $t = T_d$. This is a valid witness for the instance χ in $R_d^{\mathcal{O}}$. The above algorithm takes at most

$$T_d \cdot \lambda^c \leq \lambda^{\alpha+1} \cdot \lambda^c \leq \lambda^{(1+\varepsilon)\alpha} \ ,$$

where the last inequality holds for large enough α, concretely $\alpha > (c+1)/\varepsilon$. ◁

We proceed to show that $R_d^{\mathcal{O}}$ is depth robust.

▷ Claim 48 (Depth Robust). Let α be a large enough constant and let $d = \alpha \log_2 \lambda$. For every adversary $\mathcal{A}$ of depth $\lambda^{(1-\varepsilon)\alpha}$ and size $2^{o(\lambda)}$ we have for every $\lambda \in \mathbb{N}$:

$$\Pr\left[(\chi,\pi) \in R_d^{\mathcal{O}} \;\middle|\; \begin{array}{c} \chi \leftarrow \{0,1\}^\lambda \\ \pi \leftarrow \mathcal{A}^{\mathcal{O}}(\chi) \end{array}\right] \leq 2^{-\Omega(\lambda)} \ .$$

Proof. This is a direct application of single oracle sequential hardness Corollary 38. We have $T_d = 2^{\alpha \log_2 \lambda + 1} - 1 \geq \lambda^\alpha$. Hence $\lambda^{(1-\varepsilon)\alpha} = o(T_d / \log \alpha)$ and the claim follows by Corollary 38. ◁

The above two claims conclude the proof of the proposition. ◀

References

1 Tim Abbot, Daniel Kane, and Paul Valiant. On algorithms for nash equilibria. *Unpublished*, 2004.

2 Omer Angel, Sébastien Bubeck, Yuval Peres, and Fan Wei. Local max-cut in smoothed polynomial time. *CoRR*, abs/1610.04807, 2016. arXiv:1610.04807.

3 Saikrishna Badrinarayanan, Yael Tauman Kalai, Dakshita Khurana, Amit Sahai, and Daniel Wichs. Succinct delegation for low-space non-deterministic computation. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 709–721, 2018. doi:10.1145/3188745.3188924.

4 Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil Vadhan, and Ke Yang. On the (Im)Possibility of Obfuscating Programs. *J. ACM*, 59(2):6:1–6:48, May 2012. doi:10.1145/2160158.2160159.

5 Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. Recursive Composition and Bootstrapping for SNARKS and Proof-carrying Data. In *Proceedings of the Forty-fifth Annual ACM Symposium on Theory of Computing*, pages 111–120, New York, NY, USA, 2013. ACM. doi:10.1145/2488608.2488623.

6 Nir Bitansky, Ran Canetti, Omer Paneth, and Alon Rosen. On the Existence of Extractable One-Way Functions. *SIAM J. Comput.*, 45(5):1910–1952, 2016. doi:10.1137/140975048.

7 Nir Bitansky, Omer Paneth, and Alon Rosen. On the Cryptographic Hardness of Finding a Nash Equilibrium. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 1480–1498, October 2015. doi:10.1109/FOCS.2015.94.

8 Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical Identity Based Encryption with Constant Size Ciphertext. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, pages 440–456, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

9 Dan Boneh, Craig Gentry, and Brent Waters. Collusion Resistant Broadcast Encryption with Short Ciphertexts and Private Keys. In Victor Shoup, editor, *Advances in Cryptology – CRYPTO 2005*, pages 258–275, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

10 Shant Boodaghians, Rucha Kulkarni, and Ruta Mehta. Nash Equilibrium in Smoothed Polynomial Time for Network Coordination Games, September 2018. arXiv:1809.02280.

11 Zvika Brakerski, Justin Holmgren, and Yael Tauman Kalai. Non-interactive delegation and batch NP verification from standard computational assumptions. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 474–482, 2017. doi:10.1145/3055399.3055497.

12 Buresh-Oppenheim. On the TFNP complexity of factoring. *Unpublished*, 2006.

13 Ran Canetti, Yilei Chen, and Leonid Reyzin. On the Correlation Intractability of Obfuscated Pseudorandom Functions. In *Proceedings, Part I, of the 13th International Conference on Theory of Cryptography - Volume 9562*, TCC 2016-A, pages 389–415, Berlin, Heidelberg, 2016. Springer-Verlag. doi:10.1007/978-3-662-49096-9_17.

14 Xi Chen, Xiaotie Deng, and Shang-Hua Teng. Settling the Complexity of Computing Two-player Nash Equilibria. *J. ACM*, 56(3):14:1–14:57, May 2009. doi:10.1145/1516512.1516516.

15 Arka Rai Choudhuri, Pavel Hubacek, Chethan Kamath, Krzysztof Pietrzak, Alon Rosen, and Guy N. Rothblum. PPAD-Hardness via Iterated Squaring Modulo a Composite. Cryptology ePrint Archive, Report 2019/667, 2019. URL: https://eprint.iacr.org/2019/667.

16 Arka Rai Choudhuri, Pavel Hubáček, Chethan Kamath, Krzysztof Pietrzak, Alon Rosen, and Guy N. Rothblum. Finding a Nash Equilibrium is No Easier Than Breaking Fiat-Shamir. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2019, pages 1103–1114, New York, NY, USA, 2019. ACM. doi:10.1145/3313276.3316400.

17 Bram Cohen and Krzysztof Pietrzak. Simple Proofs of Sequential Work. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018*, pages 451–467, Cham, 2018. Springer International Publishing.

18 Constantinos Daskalakis, Paul Goldberg, and Christos H. Papadimitriou. The Complexity of Computing a Nash Equilibrium. *SIAM J. Comput.*, 39:195–259, February 2009. doi:10.1137/070699652.

19 Constantinos Daskalakis and Christos Papadimitriou. Continuous Local Search. In *Proceedings of the Twenty-second Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '11, pages 790–804, Philadelphia, PA, USA, 2011. Society for Industrial and Applied Mathematics. URL: http://dl.acm.org/citation.cfm?id=2133036.2133098.

20 Nico Döttling, Russell W. F. Lai, and Giulio Malavolta. Incremental Proofs of Sequential Work. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2019*, pages 292–323, Cham, 2019. Springer International Publishing.

21 Naomi Ephraim, Cody Freitag, Ilan Komargodski, and Rafael Pass. Continuous Verifiable Delay Functions. Cryptology ePrint Archive, Report 2019/619, 2019. URL: https://eprint.iacr.org/2019/619.

22 Alex Fabrikant, Christos Papadimitriou, and Kunal Talwar. The Complexity of Pure Nash Equilibria. In *Proceedings of the Thirty-sixth Annual ACM Symposium on Theory of Computing*, STOC '04, pages 604–612, New York, NY, USA, 2004. ACM. doi:10.1145/1007352.1007445.

23 Sanjam Garg, Omkant Pandey, and Akshayaram Srinivasan. Revisiting the Cryptographic Hardness of Finding a Nash Equilibrium. In *Advances in Cryptology – CRYPTO 2016*, pages 579–604, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

24 Craig Gentry and Daniel Wichs. Separating Succinct Non-interactive Arguments from All Falsifiable Assumptions. In *Proceedings of the Forty-third Annual ACM Symposium on Theory of Computing*, STOC '11, pages 99–108, New York, NY, USA, 2011. ACM. doi:10.1145/1993636.1993651.

25 Paul W. Goldberg and Christos H. Papadimitriou. Towards a unified complexity theory of total functions. *J. Comput. Syst. Sci.*, 94:167–192, 2018. doi:10.1016/j.jcss.2017.12.003.

26 Oded Goldreich and Johan Håstad. On the Complexity of Interactive Proofs with Bounded Communication. *Inf. Process. Lett.*, 67(4):205–214, 1998. doi:10.1016/S0020-0190(98)00116-1.

27 Oded Goldreich, Salil P. Vadhan, and Avi Wigderson. On interactive proofs with a laconic prover. *Computational Complexity*, 11(1-2):1–53, 2002. doi:10.1007/s00037-002-0169-0.

28 Pavel Hubácek, Moni Naor, and Eylon Yogev. The Journey from NP to TFNP Hardness. In Christos H. Papadimitriou, editor, *8th Innovations in Theoretical Computer Science Conference (ITCS 2017)*, volume 67 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 60:1–60:21, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.ITCS.2017.60.

29 Pavel Hubáček and Eylon Yogev. *Hardness of Continuous Local Search: Query Complexity and Cryptographic Lower Bounds*, pages 1352–1371. ACM, 2016. doi:10.1137/1.9781611974782.88.

30 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which Problems Have Strongly Exponential Complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001. doi:10.1006/jcss.2001.1774.

31 Hisao Ishibuchi and Tadahiko Murata. A multi-objective genetic local search algorithm and its application to flowshop scheduling. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 28(3):392–403, August 1998. doi:10.1109/5326.704576.

32 Emil Jerábek. Integer factoring and modular square roots. *CoRR*, abs/1207.5220, 2012. arXiv:1207.5220.

33 David S. Johnson, Christos H. Papadimitriou, and Mihalis Yannakakis. How easy is local search? *Journal of Computer and System Sciences*, 37(1):79–100, 1988. doi:10.1016/0022-0000(88)90046-3.

34 Yael Tauman Kalai, Omer Paneth, and Lisa Yang. How to Delegate Computations Publicly. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2019, pages 1115–1124, New York, NY, USA, 2019. ACM. doi:10.1145/3313276.3316411.

35 Yael Tauman Kalai, Ran Raz, and Ron D. Rothblum. How to Delegate Computations: The Power of No-Signaling Proofs. In *In Proceedings of the 46th annual ACM symposium on Theory of computing (STOC)*, pages 485–494. ACM, January 2014. URL: https://www.microsoft.com/en-us/research/publication/delegate-computations-power-no-signaling-proofs/.

36 Ilan Komargodski, Moni Naor, and Eylon Yogev. White-Box vs. Black-Box Complexity of Search Problems: Ramsey and Graph Property Testing. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 622–632, October 2017.

37 Ilan Komargodski, Moni Naor, and Eylon Yogev. White-Box vs. Black-Box Complexity of Search Problems: Ramsey and Graph Property Testing. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 622–632, 2017. doi:10.1109/FOCS.2017.63.

38 S. Lin and Brain W. Kernighan. An Effective Heuristic Algorithm for the Traveling-Salesman Problem. *Oper. Res.*, 21(2):498–516, April 1973. doi:10.1287/opre.21.2.498.

39 Carsten Lund, Lance Fortnow, H Karloff, and Noam Nisan. Algebraic Methods for Interactive Proof Systems, November 1990. doi:10.1109/FSCS.1990.89518.

40 Mohammad Mahmoody, Tal Moran, and Salil Vadhan. Publicly Verifiable Proofs of Sequential Work. In *Proceedings of the 4th Conference on Innovations in Theoretical Computer Science*, ITCS '13, pages 373–388, New York, NY, USA, 2013. ACM. doi:10.1145/2422436.2422479.

41 Nimrod Megiddo and Christos H. Papadimitriou. On total functions, existence theorems and computational complexity. *Theoretical Computer Science*, 81(2):317–324, 1991. doi:10.1016/0304-3975(91)90200-L.

42 Ralph Merkle. *Secrecy, Authentication and Public Key Systems*. PhD thesis, Stanford University, Department of Electrical Engineering, June 1979.

43 John C. Nash. The (Dantzig) simplex method for linear programming. *Computing in Science Engineering*, 2(1):29–31, January 2000. doi:10.1109/5992.814654.

44 Omer Paneth and Guy N. Rothblum. On Zero-Testable Homomorphic Encryption and Publicly Verifiable Non-interactive Arguments. In Yael Kalai and Leonid Reyzin, editors, *Theory of Cryptography*, pages 283–315, Cham, 2017. Springer International Publishing.

45 Christos H. Papadimitriou. On the complexity of the parity argument and other inefficient proofs of existence. *Journal of Computer and System Sciences*, 48(3):498–532, 1994. doi:10.1016/S0022-0000(05)80063-7.

46 Krzysztof Pietrzak. Simple Verifiable Delay Functions. Cryptology ePrint Archive, Report 2018/627, 2018. URL: https://eprint.iacr.org/2018/627.

47 Ronald L. Rivest, Adi Shamir, and David A. Wagner. Time-lock puzzles and timed-release crypto. Technical Report MIT/LCS/TR-684, MIT, February 2000.

48 Alon Rosen, Gil Segev, and Ido Shahaf. Can PPAD Hardness be Based on Standard Cryptographic Assumptions? In Yael Kalai and Leonid Reyzin, editors, *Theory of Cryptography*, pages 747–776, Cham, 2017. Springer International Publishing.

49 Alejandro Schaffer and Mihalis Yannakakis. Simple Local Search Problems That are Hard to Solve. *SIAM J. Comput.*, 20:56–87, February 1991. doi:10.1137/0220004.

50 Katerina Sotiraki, Manolis Zampetakis, and Giorgos Zirdelis. PPP-Completeness with Connections to Cryptography. *CoRR*, abs/1808.06407, 2018. arXiv:1808.06407.

51 Paul Valiant. Incrementally Verifiable Computation or Proofs of Knowledge Imply Time/Space Efficiency. In Ran Canetti, editor, *Theory of Cryptography*, pages 1–18, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

Interactive Coding with Constant Round and Communication Blowup

Klim Efremenko
Ben Gurion University, Beersheva, Israel
http://www.cs.bgu.ac.il/~klim
klimefrem@gmail.com

Elad Haramaty
Harvard University, Cambridge, MA, USA
seladh@gmail.com

Yael Tauman Kalai
Microsoft Research, Boston, MA, USA
yael@microsoft.com

Abstract

The problem of constructing error-resilient interactive protocols was introduced in the seminal works of Schulman (FOCS 1992, STOC 1993). These works show how to convert any two-party interactive protocol into one that is resilient to constant-fraction of error, while blowing up the communication by only a constant factor. Since these seminal works, there have been many followup works which improve the error rate, the communication rate, and the computational efficiency.

All these works only consider only an increase in communication complexity and did not consider an increase in *round complexity*. This work is the first one that considers the blowup of round complexity in noisy setting. While techniques from other papers can be easily adapted encode protocols with arbitrarily round complexity this coding schemes will lead to large(and usually unbounded) increase in round complexity of the protocol.

In this work, we show how to convert any protocol Π, with *no a priori* known *communication bound*, into an error-resilient protocol Π', with comparable computational efficiency, that is resilient to constant fraction of adversarial error, while blowing up both the communication complexity and the *round complexity* by at most a constant factor. We consider the model where in each round each party may send a message of *arbitrary length*, where the length of the messages and the length of the protocol may be *adaptive*, and may depend on the private inputs of the parties and on previous communication. We consider the adversarial error model, where ϵ-fraction of the communication may be corrupted, where we allow each corruption to be an *insertion* or *deletion* (in addition to toggle).

In addition, we try to minimize the blowup parameters: In particular, we construct such Π' with $(1+\tilde{O}\left(\epsilon^{1/4}\right))$ blowup in communication and $O(1)$ blowup in rounds. We also show how to reduce the blowup in rounds at the expense of increasing the blowup in communication, and construct Π' where both the blowup in rounds and communication, approaches one (i.e., no blowup) as ϵ approaches zero. We give "evidence" that our parameters are "close to" optimal.

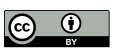
2012 ACM Subject Classification Theory of computation → Interactive computation

Keywords and phrases Interactive Coding, Round Complexity, Error Correcting Codes

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.7

Related Version A full version of the paper is available at https://eccc.weizmann.ac.il/report/2018/054/.

Funding *Klim Efremenko*: supported by the Israel Science Foundation (ISF) through grant No. 1456/18.



1 Introduction

Communication over a noisy channel is a fundamental problem in computer science, engineering and related fields. Starting from the seminal work of Shannon [30], this problem of error-resilient communication has been extensively studied. Today, we have "good" error-correcting codes – ones that achieve constant information rate as well as constant error rate. The two main error models that were considered are the *stochastic* error model, where the errors are distributed according to some distribution (such as the binary symmetric channel), and the *adversarial* error model, where errors may occur adaptively and adversarially, so long as the prescribed error rate is not exceeded. This work considers the latter (stronger) adversarial error model. In addition, we consider (adversarial) insertion and deletion errors.

In a sequence of innovative works, Schulman [27, 28, 29] initiated the study of error-resilience in the context of *interactive protocols*. Specifically, he considered the setting where two parties are interacting via a protocol over a noisy channel, where the noise could be stochastic or adversarial. Since Schulman's seminal works, there have been many followup works, that improve the error rate [9, 16, 17, 6, 1, 10], the information rate [24, 19, 12], the computational efficiency [15, 2, 4, 3], and very recently that are been beautiful works that generalize the error model of the adversary to allow insertions and deletions [8, 21, 31]. There have also been several works that consider the multi-party setting [26, 23, 7, 14]. We refer the reader to [11] for a fantastic survey on previous work on interactive coding. The focus of this work is on the 2-party setting and the adversarial error model.

All previous works consider only an increase of total communication without looking on number of rounds required to perform the task. In cryptography and in distributed computing, protocols that consist of long messages are considered, and it is desirable to keep the round complexity as low as possible. In fact, much research (in both cryptography and distributed computing) focuses on reducing the round complexity of various protocols, as often the round complexity is the bottleneck, and not the communication complexity. We argue that since we consider *interactive* protocols, we should aim for error resilient protocols, that not only blow up the communication by at most a constant factor, but also blow up the number of rounds by at most a constant factor.

Our model is the typical synchronous model used in cryptography and distributed algorithms. In our model we assume that the original protocol can be very versatile that is each party can decide how many bit he wants to send next based on previous communication and his private input. Therefore not only that number of rounds and communication of each player depends not only on the communication over the channel but also on players private input. We want to mention that more versatile original protocol is the harder it is to make a coding scheme for it. We elaborate on this model in Section 1.1.

Moreover, we emphasize that we do not assume that the communication (or round) complexity is fixed or a priori known. This is in contrast to all previous works, which assume that the communication (and round) complexity T is fixed and known in advance, and that the adversary can corrupt at most ϵT bits.[1] We note that such an assumption is often unrealistic and results in protocols where the communication complexity is always *worse-case*.

[1] We mention the work of Agrawal *et. al.* [1], which does assume that the communication complexity of the underlying errorless protocol is a priori known. However, with the goal of maximizing the error rate, the communication complexity in the error-resilient protocol is not fixed and is not a priori known. We emphasize that in our work, we do not even assume that the parties a priori know the communication complexity in the underlying errorless protocol.

In this work, we allow the communication and round complexity to differ from execution to execution, depending on the inputs, or "types" of the parties, and construct an error-resilient protocol that preserves this per-execution communication (and round) complexity. We note that the fact that we allow such adaptive (and not a priori known) communication length adds substantial technical difficulties to our work, which we elaborate on in Section 1.4.[2]

Our Results in a Nutshell. We show how to convert any protocol, where messages can be of arbitrary length, and where the communication and round complexity are not a priori known, into an error-resilient one, with comparable (computational) efficiency guarantees, that is secure against constant fraction of adversarial error, while incurring a constant blowup both to the communication complexity and to the round complexity. We allow the adversary not only to toggle with the bits of communication, but also allow the adversary to *insert* and *delete* bits. We elaborate on our communication model and error model in Section 1.1.

Moreover, we try to minimize the (constant) overhead in communication and rounds: In particular, we obtain $(1 + \tilde{O}(\epsilon^{1/4}))$ blowup in communication and $O(1)$ blowup in rounds. We also show how to reduce the blowup in rounds at the expense of slightly increasing the blowup in communication, and construct an error-resilient protocol where both the blowup in rounds and communication approaches one as ϵ approaches zero. We elaborate on our results (and on the exact parameters we obtain) in Section 1.2, we give a high-level overview of our techniques in Section 1.4, and give "evidence" that our parameters are "close to" optimal in Section 2.3 (after formally stating our main theorem in Section 2.2).

Our Technical Hurdles

The reader may at first think that dealing with short messages is the "hard case", since for long messages we can use standard error-correcting codes. We argue that this intuition is misleading. First, when considering adversarial error, applying an error-correcting code to each message separately does not help, since the entire message can be corrupted (even if the message is long), and indeed in this work we focus on adversarial error. We mention, however, that even for the case of stochastic error, dealing with messages of varying lengths, where some messages may be short while other messages may be long, is challenging.

Before explaining the difficulties that arise in this setting, we note that if we knew a priori the number of rounds and the communication complexity of the underlying protocol, then we could have "smoothed" it out perfectly, so that all the messages would have been of equal length,[3] and then we could have used a protocol (and analysis) from prior works.

Since we do not have such a bound, we cannot perfectly smooth out the underlying protocol. Nevertheless, we must somehow smooth out the protocol, since a party cannot send a long message before she is "sufficiently confident" that the transcript so far is correct, as otherwise, this long message will be wasted (even if the adversary does not corrupt it at all). Therefore, we "approximately" smoothen out the underlying protocol, by guaranteeing that each message is of length at least half and at most twice the length of the previous message. We refer the reader to Section 1.4 and Section 3 for details.

[2] We believe (though we haven't checked) that the tree-code based interactive coding schemes may easily be adapted to the setting where the communication complexity is not a priori bounded, by having each party construct an (infinitely growing) tree code. However, in tree-code based schemes the parties are computationally inefficient and there is a large blowup to the round complexity.

[3] This approach blows up the communication and round complexity by a constant factor. If the goal is to optimize this blowup (as we do in this work) then one cannot afford to perfectly smoothen out the protocol, in which case our techniques are needed.

We mention that in order to minimize the blowup, we consider two small constants $\alpha, \beta > 0$ (that depend on the error rate) and guarantee that the length of each message is at least $\alpha\ell$ and at most $\frac{\ell}{\beta}$, where ℓ is the length of the message preceding it. This is not important for the high-level overview.

We emphasize, that even after smoothing the underlying protocol, the length of the messages can still grow (or shrink) at an exponential rate, which brings rise to several challenges. For example, similar to many previous works (such as[27, 2, 19]), when a party realizes that there was an error she backtracks. In our setting we need to be extremely cautious when we backtrack. Note that the adversary can cause us to backtrack even though we are synchronized, by making us believe that we are out of sync. Previous works ensure that the adversary needs to invest enough error for such backtracking, and hence such "false" backtracking is costly for the adversary. However, in the case where messages are of varying length, this analysis becomes extremely delicate, since the adversary can corrupt a short message (by investing a small amount of his error budget), and thus falsely cause the parties to backtrack and delete a previous long message. Indeed, as opposed to previous protocols, we do not erase when we backtrack. Rather, we keep this transcript as "questionable". We refer the reader to Section 1.4 and Section 4 for details.

Moreover, when messages are of varying lengths, even if the protocol is (approximately) smooth, and even if we backtrack carefully, ensuring that the round complexity does not blow up, does not only require a careful (and significantly more complex) analysis, but also requires additional new ideas.

For example, the protocols in previous works, perform an equality test after every chunk of length d (for some parameter d), where in this equality test the parties check whether they are in sync by sending each other a hash of their transcript so far. In our setting, messages may be very long, and we cannot chop a message to chunks of d bits each, since this will blow up the round complexity. Instead, it is tempting to simply append to each message a hash of length that is proportional to the message length (e.g., append a hash of length $\lfloor \frac{\ell}{d} \rfloor$ to a message of length ℓ). However, as we show in Section 1.4 and Section 5, in order to ensure a constant blowup in round complexity, we must not only allow the length of the hash value to depend on the length of the message it is being appended to, but rather it should also depend on the length of the *entire history*. This is the case since if the protocol has messages of varying lengths, the adversary can corrupt a single long message, in a way that causes many hash collisions in future short messages. Thus, by corrupting one (long) message many rounds can be wasted.

In order to get around this problem, we allow the length of the hash to depend on the length of the entire history. Moreover, we consider randomized (i.e., seeded) hash function, where the party sends the hash value together with the hash seed, so that the adversary does not know which hash function will be used ahead of time. However, with a seed of length w one can hash messages of length at most 2^w, and the history may be longer than 2^w. Thus, in our scheme some of the seed is chosen ahead of time and some of the seed is chosen with each message. We refer the reader to Section 1.4 and Section 5 for details.

Moreover, the fact that the communication complexity is not a priori known creates an additional problem. Following previous works (such as [2, 3, 19]), we first construct a protocol in the *common random string* (CRS) model (this is done in Section 5), and then we remove the CRS (in Section 6). Removing the CRS in previous work was straightforward: First show that the CRS can be made relatively short (of size proportional to the communication complexity) by using a δ-biased source, and then argue that one of the parties can simply send the CRS using a (standard) error correcting code. In our case this cannot be done since we do not have an a priori bound on the communication complexity.

We give an overview on how we overcome the technical hurdles mentioned above in Section 1.4, but warn the reader that overcoming these challenges is quite difficult, and results in a very complex analysis.

We next explain our model in more detail.

1.1 Our Model

1.1.1 The Noiseless Model

We consider 2-party protocols, between two parties, Alice and Bob. In our model, at every round i, Alice and Bob do the following: Alice chooses $\ell_A(i) \in \mathbb{N}$ (greater than 0) and a message $m_A(i) \in \{0,1\}^{\ell_A(i)}$, based on her view of previous communication and her private input, and sends $m_A(i)$ to Bob. Similarly, Bob chooses $\ell_B(i) \in \mathbb{N}$ and a message $m_B(i) \in \{0,1\}^{\ell_B(i)}$, based on his view of previous communication and his private input, and sends $m_B(i)$ to Alice. At some round, one of the parties aborts, and both parties report an output.

More generally, we allow Bob's message in the i'th round to depend, not only on all previous communication and his private input, but also on Alice's message in the i'th round. This corresponds to the synchronous model where in each round i, Alice and Bob do not send their messages simultaneously, but rather first Alice sends her message and only then Bob sends his message (which may depend on Alice's message). This model is known as the message-passing model, and is the most common model used in cryptography (and distributed algorithms). We note that our results also apply to the synchronous simultaneous message model, and the choice of presenting our results in the synchronous message-passing model was due to the fact that we think that this model is more standard.

We denote the input of Alice by x, and we denote the input of Bob by y. Note that a pair of inputs (x, y) define ℓ_A and ℓ_B for all rounds, and also define the number of rounds. Thus, in the noiseless setting, for any protocol we can define $\mathrm{CC}(x, y)$, which is the communication complexity of the protocol for the input pair (x, y). Similarly, we can define $\mathrm{R}(x, y)$, which is the number of rounds for the input pair (x, y).

1.1.2 The Noisy Model

In this work, we consider the adversarial error model, and assume that the adversary can corrupt any ϵ-fraction of the bits, for some a priori fixed small constant $\epsilon > 0$. We allow the adversary, not only to toggle with the bits, but he can also insert and delete bits.

In our model, where messages can be of arbitrary length, protecting protocols against insertions and deletions is extremely important, since otherwise the parties can securely encode information via the *length* of the messages. Specifically, in our model, where messages can be of varying lengths, one can trivially protect protocols against (adversarial) toggle corruptions while incurring only a constant factor blowup in the communication complexity, albeit an *unbounded blowup* in the round complexity, as follows: First convert the protocol to a protocol where each party sends a single bit in each round. Then, encode this bit as follows: If the bit is zero then encode it via a single bit (zero or one), and if the bit is 1 then encode it via any two bits. Upon receiving an encoded message the parties will decode without looking at the content of the message, but rather only by the length of the message.

We note that most previous work on interactive coding do not consider insertion and deletions. Indeed, in the synchronous model, where the parties send one bit per round, insertions and deletions are not interesting, since the parties "can tell" when an insertion or deletion occurs.

An exception are the recent works of [8, 21, 31]. These works consider insertion and deletions in the *asynchronous* model. More specifically, they consider an adversary who can insert a message from one party and delete a message from the other party, and thus cause the parties to be out-of-synch with regard to which round they are on (though similarly to previous work, they are in the bit-by-bit model, thus their protocols incur a large blowup to the round complexity, and they assume an a priori bound on the communication complexity).

In our work, we consider the *synchronous* model, where the parties always agree on the number of speaking alternations (which in our case is exactly the number of rounds). We emphasize, however, that the work of [21] shows a generic method for converting any error-resilient protocol in the synchronized model into one that that is error resilient in the asynchronized model. We believe that one can use their approach (and in particular the use of a synchronization code [20]) to boost our result from the synchronous model to the asynchronous model.

In the adversarial error model, in our work and in all previous works, there is an a priori fixed constant ϵ and it is assumed that the adversary can corrupt at most ϵT bits, where T is the number of bits communicated. In most previous work, the value of T was assumed to be a priori known (and fixed). As mentioned above, in this work, we allow the length of the protocol to depend on previous communication. This models the real world setting, where we cannot a priori predict the length of our conversations, and it can depend on our private inputs (or on our "types").

In this model, care should be taken when defining the adversarial error model. One possibility is to allow the adversary to corrupt ϵT bits, where T is the number of bits that would have been transmitted assuming no error.[4]

We note, however, that with such a definition the adversary can use his ϵT bits of corruption budget, and cause the parties to abort prematurely. Namely, he can convince both parties that the other party is "boring" (i.e., that the other party has an input such that if they were executing the error-free protocol without error, the number of bits exchanged would have been less than ϵT). In such case both parties would abort prematurely and the adversary would "win".

Instead, we allow the adversary to corrupt only ϵ-fraction of the bits that were actually communicated. We note that a similar model was used in the work of Agrawal et al. [1], where their goal was to get optimal error-rate, and to that end, they considered error-resilient protocols with an adaptive speaking order and where the communication complexity may depend on the error pattern.[5] We emphasize that this adversarial model is stronger than alternative (natural) models, such as the the prefix model that allows the adversary to corrupt ϵ-fraction of any prefix of the transcript.

In this work, we also add a bound ϵ' on the number of rounds that the adversary can "fully" corrupt, where we say that a round is *fully corrupt* if the adversary corrupts more than δ-fraction of the bits, for some small constant δ (which depends on the error bound ϵ). We note that all previous works also had such a bound (implicitly), since in previous works there was no distinction between rounds and communication. In contrast, in our model, a bound on the number of bits corrupted does not imply a bound on the number of rounds that are fully corrupted. For example, consider the protocol in which there is one long message of length ℓ, followed by $\epsilon\ell$ messages, each consisting of a single bit. In such a protocol, not corrupting the long message gives the adversary the budget to corrupt all the short messages.

[4] We note that the adversary knows T since we assume (similarly to all previous work that consider the adversarial error model), that the adversary knows the private inputs of the parties.

[5] As mentioned above, the work of [1] does assume an a priori known bound on the communication complexity of the underlying (errorless) protocol.

We emphasize that bounding the number of rounds that are fully corrupted is necessary, since without such a bound, it is impossible to ensure a small blowup in round complexity. This argument is deferred to Section 2.3 where we give evidence to the optimality of our parameters.

1.2 Our Results

In what follows, we denote our error parameters by ϵ and ϵ', where ϵ corresponds to the fraction of corrupted bits, and ϵ' corresponds to the fraction of (fully) corrupted messages. We show that for any (small enough) constants $\epsilon, \epsilon' > 0$ there exist blowup parameters $\alpha, \alpha' > 0$ such that one can convert any protocol into an error resilient one (with respect to ϵ and ϵ'), with α blowup in communication, and essentially α' blowup in rounds (with an additional term that depends logarithmically on the communication complexity). We can set $\alpha' = O(1)$ we obtain a blowup of $\alpha = \tilde{O}\left(\epsilon^{1/4}\right)$ in communication complexity. Alternatively, one can set α, α' such that they both approach 0 as ϵ, ϵ' approach 0.

Our error-resilient protocol is randomized, even if the original protocol was deterministic. This is similar to all previous works that construct computationally efficient interactive coding schemes that are robust to adversarial error (starting with the work of [2]). Schulman [28] (followed by many followup works) gave a deterministic interactive coding scheme, at the price of computational inefficiency. The parties in the error resilient scheme run in exponential time in T, where T is an upper bound on the length of the underlying protocol.[6] Recently, Gelles et al. [13] gave a deterministic and efficient construction for the case of random error. However, constructing a deterministic interactive coding scheme that is resilient to adversarial error and is computationally efficient remains an interesting open problem.

We are now ready to state our main theorem. The most general theorem can be found in Section 2, and in what follows we present our theorem in a regime of parameters that we think is of particular interest.

In what follows, we let $t_{\min}$ denote the minimum value for which the underlying error-free protocol Π transmits at least $t_{\min}$ bits.

▶ **Theorem 1** (Main Theorem (informal)). *For any sufficiently small $\epsilon \geq 0$ and for any $\epsilon' \leq \epsilon^{1/4}$, there exist blowup parameters α and α', and a polynomial time probabilistic oracle machine S, such that the following holds. For any adversary $\mathcal{A}$ that corrupts at most ϵ-fraction of the bits of the simulated protocol $\Pi'_{\mathcal{A}}$ (which is the protocol Π' executed with the adversary $\mathcal{A}$), and "fully" corrupts at most ϵ'-fraction of the messages of $\Pi'_{\mathcal{A}}$, where $\mathcal{A}$ "fully" corrupt a message if he corrupts at least α^2-fraction of the bits of the message, we have the following guarantees.*

1. $\mathrm{CC}(\Pi'_{\mathcal{A}}) \geq t_{\min}$.
2. $\Pr\left[\mathrm{CC}(\Pi'_{\mathcal{A}}) > (1+\alpha)\mathrm{CC}(\Pi)\right] = \exp\left(-\mathrm{CC}(\Pi'_{\mathcal{A}})\right)$.
3. $\Pr\left[R(\Pi'_{\mathcal{A}}) > (1+\alpha')R(\Pi) + \alpha' \log \mathrm{CC}(\Pi)\right] = \exp\left(-R(\Pi'_{\mathcal{A}})\right)$.
4. $\Pr\left[(\mathsf{Output}(\Pi'_{\mathcal{A}}) \neq \mathsf{Trans}(\Pi))\right] = \exp\left(-\mathrm{CC}(\Pi'_{\mathcal{A}})\right)$.

Moreover, we can choose the parameter α, α' such that $\alpha = \tilde{O}(\epsilon^{1/4})$ and $\alpha' = O(1)$, or we can choose α, α' such that α and α' approach 0 as ϵ approaches 0.

The purpose of $t_{\min}$. In the theorem above, without adding the restriction that $CC(S^A, S^B) \geq t_{\min}$, the simulated protocol could have aborted as soon as more than ϵ-fraction of error was detected. In particular, if the first bit was noticeably corrupted, then the parties in

[6] Braverman [5] showed how to improve the parties' runtime to be sub-exponential in T.

the simulated protocol could have safely aborted. Thus, without adding the restriction that $CC(S^A, S^B) \geq t_{\min}$, this theorem does not even generalize previous works, which all assume an a priori fixed transcript size t and assume the adversary makes at most ϵt corruptions. Adding this restriction, gives the adversary an initial budget of $\epsilon t_{\min}$ corruption bits. Moreover, since the error probability is exponentially small in the actual transcript length, the requirement $CC(S^A, S^B) \geq t_{\min}$ guarantees a low error probability.

There was a long line of works that consider the case of unknown noise of

1.3 Related Works

This work is the first one that considers the blowup of round complexity in noisy setting. While techniques from other papers can be easily adapted encode protocols with arbitrarily round complexity this coding schemes will lead to large(and usually unbounded) increase in round complexity of the protocol. However there are many papers that considered models related to our model. We want to mention that in other papers consider an effect of message length(what mean also the round complexity) on the rate of the communication complexity of the coding one can see it non-explicitly in [24, 19] and more explicitly in [22]. In non-noisy setting there was a long line of works showing that reducing round complexity may cause an exponential gap in communication complexity.

The paper [1] considered a very adaptive models of the protocols where total communication is not known in advance. There was also a large line of works considering the case when to total amount of noisy in unknown see survey by Gelles [11] for more details.

1.4 Overview of Our Techniques

In this section we give a high-level overview of the main ideas behind our construction and our analysis. In this overview, we do not focus on getting "optimal" parameters, and focus on constructing an error-resilient scheme that blows up the round and communication complexity by a constant factor. We note that all the conceptual ideas in this work are needed even to achieve constant overhead.

We start with an arbitrary protocol Π.

Smoothness. We first convert Π into a smooth protocol, with the property that after a message of length ℓ comes a message of length at most 2ℓ, and before a message of length ℓ comes a message of length at least $\ell/2$. We mention that in the actual protocol, to minimize the blowup in rounds and communication, we define (α, β)-smoothness, where α and β are functions of the error rate ϵ, and the guarantee is that after a message of length ℓ comes a message of length at least $\alpha\ell$ and at most $\frac{\ell}{\beta}$, and we show how to convert any protocol Π into an (α, β)-smooth protocol.

As mentioned above, the reason we need to smoothen Π is that otherwise, if after receiving a short message a party sends a long message, then the adversary by corrupting the short message, can cause the long message to be wasted, thus effectively allowing him to corrupt the long message by only using the budget needed to corrupt the short message.

Intuitively, we smoothen Π by instructing a party who wishes to send a long message after receiving a short message, to do so "cautiously", by sending the long message over several rounds, each time increasing the message length by at most a factor of 2.

To ensure that this does not cause a blowup to the round complexity, we make sure that a party does not send a short message after receiving a long one. Otherwise, suppose Alice always sends long messages (each of length ℓ) and Bob always sends single bit messages. Then

by having Alice send her messages "cautiously", as explained above, the round complexity will blowup by a factor of $\log \ell$, which is too large. Instead, we instruct Bob to send longer messages, of length $\alpha\ell$, so that the adversary will need to invest enough budget to corrupt Bob's message; in particular, enough to allow Alice to send her length ℓ message safely.

We refer the reader to Section 3 for the formal definition of smoothness, and to Lemma 6 for how to convert a protocol into a smooth one.

From now on we assume the protocol Π is smooth, and show how to convert it into an error resilient one.

Message adversary. We first note that we can focus our attention only on adversaries, that rather than corrupting individual bits, corrupt messages, where the price of corrupting a message m is the maximum between the length of m and the length of the corrupted version of m. If the adversary chooses to corrupt a message m then he may corrupt it adversarially, and if the adversary chooses not to corrupt a message then he cannot make any changes to it.

The reason we can focus on such adversaries is that we can easily convert any protocol that is resilient to errors made by message adversaries into a protocol that is error resilient to any adversary by applying an error correcting code to each message, and hence if only a small fraction of a message is corrupted (smaller than the allowed error rate) then this corruption can be ignored, since it is immediately corrected by the error correcting code. We use the error correcting code of Guruswami and Li [18], that is resilient to insertion and deletions, and has a minimal blowup of $1 + \tilde{O}(\sqrt{\epsilon})$ to the message length.

Thus, from now on, throughout this section, we ignore the layer of error correcting code, and consider only message adversaries.

1.4.1 The Protocol in the Ideal Hash Model

We first show how to convert any protocol Π into a protocol that is error resilient in the *Ideal Hash Model*. As in previous works (starting with the original work of [27]), our starting point is the idea of using hashing to check for consistency. Namely, in the protocol Alice and Bob check equality of their partial transcripts, by sending to each other hashes of their partial transcripts.

In the Ideal Hash Model, we assume the existence of an "ideal" hash function, that is known to all parties and does not need to be communicated, and in the analysis we assume that the number of hash collisions is bounded, yet adversarially chosen (where the cost for each hash collision is proportional to the length of the hash value). We later elaborate on how we remove this ideal model assumption, by implementing this ideal hash using a real hash function.

For the sake of simplicity, throughout this overview we think of the parties appending to each message they send a hash of their transcript so far. We mention however, that in the actual protocol, since we want to optimize the communication blowup, we append a hash only to "long enough" messages, i.e., messages of length at least d, for a carefully chosen parameter $d \in \mathbb{N}$. In particular, we do not append a hash to short messages, and instead add a hash in every round that divides d (to take care of the case where all the messages are short).

Each party, upon receiving a message, first checks the consistency of the corresponding hash with its current transcript. If an inconsistency occurs, the parties enter a *correction mode*.

Correction Mode. In correction mode, the parties realize that their transcripts are inconsistent, and they need to rewind their transcript to a point where they believe they are consistent, yet without backtracking too much. Note that once an error is detected, the parties cannot simply rewind their transcript one round at a time, since the adversary can cause them to completely get out of sync. Moreover, they cannot send each other the round number they are currently simulating, as was done in [2], since this will blowup the communication by too much. Instead, we adapt the idea of backtracking to a "meeting point", an idea that was originated in [27] and used in [19]. For the sake of completeness, we explain this idea below.

Once the parties realize they are not in sync, they enter a correction mode, and once in error mode, they send two hashes of their transcript: One hash of the entire transcript, and the other of the transcript up until the second largest round. If a consistency was found they go back to the point of consistency. Otherwise, they send two hashes of their transcript until the largest, and second largest, round which is a multiple of 2. Again, if a consistency was found they go back to the point of consistency. Otherwise, in the i'th try, they send two hashes of their transcript until the largest, and second largest, round which is a multiple of 2^{i-1}.

In order to avoid the situation where the adversary invests $O(1)$ corruptions, and causes a party to go back 2^i steps, and thus lose 2^i bits of a possibly good transcript, the parties go back 2^i steps only after receiving roughly 2^i confirmations. The confirmations cannot be in a single round, since then the adversary could corrupt a single round and cause the parties to go back (possibly) 2^i rounds. Thus, instead these confirmations should span roughly 2^i rounds, and each party keeps a counter of how many confirmations it has.

One important missing piece is that they can be out of sync with respect to which are the meeting points. Thus, we also append to the message a hash of E, which denotes the number of rounds the party is in the error mode, and this length determines where the meeting points should be (which is roughly the power of 2 closest to E).

In previous works, once the parties backtrack, they erase the (seemingly) inconsistent transcript and continue to simulate the actual protocol. One important point where our protocol differs from all previous work, is that in our protocol the parties cannot afford to erase their (seemingly) inconsistent transcripts. This is due to the fact that the messages in the (seemingly) inconsistent part may be very long. For example, consider the case where the last message added to the transcript is of length 1, the one prior is of length 2, the one prior is of length 4, then length 8, and so on. Suppose no errors occurred and everything is consistent. The adversary can corrupt the hash appended to the short (1 bit) message, making the parties believe that their transcripts are inconsistent. The parties will backtrack, but the adversary will continue to make them believe that they are inconsistent, so that they erase i messages. This means erasing 2^i bits of communication, which is way more than the parties can afford to erase. Therefore, in our protocol, rather than erasing the (seemingly) inconsistent transcript, we keep it as questionable, and enter what we call a *verification mode*.

Verification Mode. In the verification mode, the parties simply test whether their questionable messages are consistent. They do this round-by-round, by sending a hash of the messages corresponding to each round. If their hashes agree, they mark the round as valid, and continue to the next round. If they arrive to a round where their messages do not agree, they don't immediately erase all the questionable transcript. Rather, they erase it only after they are "sufficiently" confident that they are inconsistent. To this end, they

send longer and longer hashes until the number of bits of hash are proportional to the (seemingly) inconsistent transcript, and if the inconsistency persists then the parties erase their questionable transcript, and continue to simulate the underlying protocol.

This protocol is formally presented in Section 4, and the formal analysis in the Ideal Hash Model can be found in Section 4.2 and proof will appear in full version.

1.4.2 Our Protocol in the Shared Randomness Model

We next show how to implement the ideal hash functions with a specific hash function. To this end, we construct a function family $\mathcal{H} = \{h_x\}$, where each hash function h_x is associated with a (possibly long) seed x.

We consider the *shared randomness model*, where the parties are allowed to share a (possibly long) random string. We later show how to eliminate the need for shared randomness. But for now, we assume that the shared randomness is as long as we need. In particular, we use a different hash function (i.e, a different seed) for each equality test, and assume that the shared random string contains all these seeds. Since the length of the protocol is adaptive and not a priori bounded, the length of the common random string is also not a priori bounded.[7]

We emphasize that the shared randomness (and in particular the seeds) are known to the adversary. Therefore the adversary, given a seed x, can try to skew the protocol and cause the parties to send many messages whose hashes collide.

Note that the adversary has $\binom{t}{\epsilon t} = 2^{\tilde{O}(\epsilon)t}$ different ways to corrupt the t bits of the communication. Thus, he can cause hash collisions in approximately $\tilde{O}(\epsilon)t$ bits. If we append each message of size ℓ with $O(\ell)$ bits of hash, the adversary will be able to cause hash collisions in messages with total volume of $\tilde{O}(\epsilon)t$, which is within the allowed error range. Indeed, our main challenge is to bound the number of *rounds* with hash collisions, a challenge that previous works did not need to deal with since in their setting, communication complexity and round complexity are equivalent.

If we a priori knew the length of the transcript t and the number of rounds R, then we could add $U = \frac{t}{R}$ bits of hash to each message, and since the adversary can cause only $\tilde{O}(\epsilon)t$ bits of hash collisions, the number of rounds in which the adversary can cause a hash collision, is bounded by $\tilde{O}(\epsilon)R$, which is again within our allowed error range.

Since we don't have such a bound, it is tempting to append to each message sent in round r a hash of length $U_r = \frac{t_r}{r}$, where t_r is the communication up to the round r. But the following example shows that such a padding does not suffice, and the adversary can still force too many rounds with hash collisions.

Consider a protocol that consists of $\tilde{O}(1/\epsilon)$ chunks such that chunk 0 consists of k single bit messages, and each chunk $i \neq 0$ consists of a single (long) message of length $2^i k$, followed by $\tilde{O}(\epsilon)k$ single bit messages. Note that in this case, the total number of hash bits in chunk i is $\approx 2^i$, and thus an adversary that corrupts the long message of this chunk can cause hash collisions in all the rounds of the chunk, resulting with a total of $O(R)$ rounds with hash collisions.[8]

[7] We later show how we convert any such protocol in the *unbounded* shared randomness model into one that uses only private randomness.

[8] to be more precise, we need a long enough message at the end of the protocol to give the adversary enough budget to corrupt all of the long messages.

To overcome this issue, the idea is to partition the protocol to chunks (which we call regimes), and append to each message a hash of length that is proportional to the average length of a message in the chunk. To be precise, we append to each message a hash of length $U_r = \max_{r'<r} \frac{t_r - t_{r'}}{r - r'}$. Unfortunately, in this case the total amount of hash bits being added can be as large as $t \log t$, which we cannot afford.

To overcome this issue, in each round, instead of sending all the U_r bits of hash, we send only a hash of these bits, where the seed of this (outer) hash is chosen using private randomness. Specifically, rather than sending $H_x(T)$ (which consists of U_r bits), the party chooses a random seed S, and sends $H_S(H_x(T))$, together with S. This reduces the number of bits being communicated from U_r to $\log U_r$. Since the adversary does not a priori know the private randomness chosen by the parties, he cannot corrupt the history to cause a hash collision in H_S in too many rounds. Moreover, since we saw that he cannot cause hash collisions in $H_x(T)$ in too many rounds, these hash functions are "safe". We note that a similar idea of using a randomized hash function was used by Haeupler [19], for the sake of improving the rate of his interactive coding scheme. We refer the reader to Section 5 formal description of the hash function.

Finally, to conclude the analysis, we need to show that adding these (randomized) hash functions does not blow up the communication by too much. More precisely, one needs to show that $\sum_r \log U_r = O(t)$. This analysis is extremely delicate and requires several new ideas.

1.4.3 The Protocol in the Private Randomness Model

Finally, we show how to remove the need for shared randomness, while using only the private randomness of the parties. Namely, we show how to convert any protocol Π in the shared random string model, to one that uses only private randomness.

The basic idea is to follow the approach used in previous works (such as [2, 19]), and replace the long shared randomness with $2^{-O(T)}$-biased randomness, where T is an upper bound on the communication complexity. Such $2^{-O(T)}$-biased randomness can be generated using only $O(T)$ random bits. So, the basic idea is to send these $O(T)$ bits of randomness in advance, using an error correcting code. If we indeed had a bound T on the communication complexity, then this idea would work, and we would be done.

However, in our setting, we do not have an a priori bound on the communication complexity. In particular, if the communication complexity exceeds $O(T)$, then the adversary has the budget to corrupt more than $O(T)$ bits, and hence can completely corrupt the randomness s. We overcome this problem by sending more (and "safer") randomness as the communication complexity increases.

The protocol starts when one of the paries, say Alice, samples the shared random string $s_1 \in \{0,1\}^{O(t_{\min})}$ on her own (using her private randomness), and sends it to Bob. Then the parties execute Π with s_1 as the shared randomness. Once the communication complexity exceeds $O(t_{\min}/\epsilon)$, where ϵ is the corruption rate of the adversary, the random string s_1 is no longer "safe", and the the parties exchange a new random string s_2 of length $O(t_1)$, where t_1 is the current communication complexity. In addition to sending the new random string s_2 the parties also resend the previous random string s_1. The reason for resending previous seeds is that by resending the seeds the goal is to ensure that if one of the seeds was ever corrupted then the parties will "catch" the adversary, since the adversary does not have enough budget to continue to corrupt that seed, and the first time that he does not corrupt it, the parties will notice the inconsistency and abort, with the guarantee that the adversary performed too many errors.

In a similar way, after the communication complexity exceed $t_2 = O(t_1/\epsilon)$ a party will choose at random s_3 such that $|s_1| + |s_2| + |s_3| = t_2$, and will send (s_1, s_2, s_3), where t_2 is the current communication complexity, etc. As mentioned above, we ensure that if at any point, a message encoding randomness was decoded incorrectly, then eventually the paries will abort, and "catch" the adversary with injecting too many errors. This guarantee simplifies the analysis: Either at some point a randomness message was decoded incorrectly, in which case the adversary is "caught" with injecting too many errors, or all the parties always agree on the randomness, in which case correctness follows from the correctness of the underlying protocol in the shared randomness model.

A minor problem with the above idea is the following: a randomness message (s_1, s_2) may have been corrupted and converted into a protocol message, and a few rounds later a protocol message could have been corrupted and converted into the same randomness (s_1, s_2). To ensure that the parties will notice such corruption, we add to the randomness also the rounds $r_1, \ldots, r_k$ in which randomness were sent.

However, there is still a problem with the above idea, which is that in the early stage of the protocol, the shared random string has relatively large bias since it is generated using a short seed, and yet the adversary may have the budget to corrupt many bits, since the total communication may be large. It can be shown that such a powerful adversary can make too many hash collisions in the first part of the protocol.

To overcome this problem, we "enforce" that the adversary corrupts at most $O(\epsilon)$ fraction of any prefix of the protocol. To do so, in each time t_i, in addition to sending $(s_1, ..., s_{i+1})$ (together with $(r_1, \ldots, r_i)$), the parties send the transcript they have seen so far. If the parties detect that the adversary made significantly more than the allowed ϵ fraction of error, they abort, causing him to fail by exceeding his allotted corruption budget.

The formal protocol is described in Section 6.

2 Our Results

In this section we present our main theorem, and give an intuitive argument for why our parameters seem to be optimal. We start by introducing notations and definitions that we use in our theorem, and throughout the manuscript.

2.1 Notations and Definitions

For any 2-party protocol $\Pi = (A, B)$, we denote by $\mathsf{Trans}(\Pi)$ the transcript of Π, which consists of all the messages exchanged throughout an execution of the protocol Π. We denote by $\mathsf{Output}(\Pi)$ the output of the parties after executing Π. We think of the protocol Π as being a deterministic protocol with no inputs. This is without loss of generality since we can always hard-wire the randomness and input into the protocol. We denote by $\mathrm{CC}(\Pi)$ the communication complexity of Π, and we denote by $R(\Pi)$ its communication complexity.

We consider simulators for simulating an interactive protocols. A simulator is a probabilistic oracle machine, that uses a protocol $\Pi = (A, B)$ as an oracle, and produces a new protocol $\Pi' = (S^A, S^B)$ that outputs the transcript of Π (even in the presence of error). For any adversary $\mathcal{A}$ we denote by $\Pi'_{\mathcal{A}}$ the protocol Π' executed with the adversary $\mathcal{A}$.

▶ **Definition 2.** *We say that an adversary $\mathcal{A}$ corrupts at most ϵ-fraction of the bits of a protocol Π' if the number of corruptions made by $\mathcal{A}$ is at most $\epsilon \mathrm{CC}(\Pi'_{\mathcal{A}})$, where each corruption is either a toggle, an insertion or a deletion. The adversary $\mathcal{A}$ can be computationally unbounded, and its corruptions may depend arbitrarily on states of both parties in Π'.*

▶ **Definition 3.** *We say that a message is γ-corrupted if the adversary corrupts at least γ-fraction of the bits of the message.*

We say that $f(x) = \tilde{O}(g(x))$ if there exists a $c \in \mathbb{N}$ such that $f(x) = O\left(g(x)\log^c\left(g(x)\right)\right)$ and we say that $f(x) = \tilde{\Omega}(g(x))$ if there exists $c \in \mathbb{N}$ such that $f(x) = \Omega\left(g(x)\log^{-c}\left(g(x)\right)\right)$.

2.2 Our Main Theorem

▶ **Theorem 4.** *There exists a universal constant $\alpha_0 \geq 0$ such that for any blowup parameters $\alpha \leq \alpha_0$ and $\alpha' \leq 1$, there exist parameters $\epsilon = \tilde{\Omega}\left(\alpha^{3+\frac{1}{\alpha'}}\right)$, $\epsilon' = \tilde{\Omega}(\alpha{\alpha'}^3)$, and $\delta = \alpha^{O(1/\alpha')}$, and there exists a probabilistic oracle machine S, such that for any protocol $\Pi = (A, B)$, in which the parties always transmit at least $t_{\min}$ bits (even in the presence of error), and for any adversary $\mathcal{A}$ that corrupts at most ϵ-fraction of the bits of the simulated protocol $\Pi'_{\mathcal{A}}$, the protocol $\Pi'_{\mathcal{A}}$ (which is the protocol Π' executed with the adversary $\mathcal{A}$), satisfies the following properties.*

1. $\mathrm{CC}(\Pi'_{\mathcal{A}}) \geq t_{\min}$.
2. *There exists $t_0 = (1 + \tilde{O}(\alpha))\mathrm{CC}(A, B)$ such that for all $t > t_0$*

$$\Pr[\mathrm{CC}(\Pi'_{\mathcal{A}}) > t] \leq 2 \cdot 2^{-\delta t} ,$$

where the probability over the private randomness of S.

3. *There exists $r_0 = (1 + O(\alpha'))\, R(A, B) + O\left(\frac{1}{\log\frac{2}{\alpha'}}\log \mathrm{CC}(A, B) + 1\right)$ such that for any $r \geq r_0$, if at most ϵ'-fraction of the messages are α^2-corrupted, then*

$$\Pr\left[R(\Pi'_{\mathcal{A}}) > r\right] \leq 2 \cdot 2^{-\delta r} ,$$

where the probability over the private randomness of S.

4. *For any $t > 0$,*

$$\Pr\left[(\mathsf{Output}(\Pi'_{\mathcal{A}}) \neq \mathsf{Trans}(\Pi)) \ \wedge \ (\mathrm{CC}(\Pi'_{\mathcal{A}}) > t)\right] \leq 2 \cdot 2^{-\delta t} ,$$

where the probability over the private randomness of S.

5. *S is a probabilistic polynomial time oracle machine, and hence the computational efficiency of S^A and S^B is comparable to that of A and B, respectively.*

In Section 2.3 below, we give an intuitive argument for why our parameters seem to be optimal. Then, the rest of the manuscript is devoted to proving Theorem 4. Before, explaining our choice of parameters, in what follows, we give a high-level overview of the structure of the proof of Theorem 4.

Road Map. We first convert $\Pi = (A, B)$ into a smooth protocol Π_{smooth}. We show how this can be done in Section 3. Then, in Section 4, we show how to convert any smooth protocol Π_{smooth} into a protocol Π_{ideal}, which is error-resilient in the ideal hash model. In this model, we assume that the adversary is a "message adversary", which means that if he corrupts even a single bit of a message the price he pays for such a corruption is the length of the entire message (more precisely, the maximum between the length of the original message and the length of the corrupted version of it). Moreover, we assume that the number of hash collisions is bounded and adversarially chosen. We refer the reader to Section 4 for details.

In Section 5, we show how to convert Π_{ideal} into a protocol Π_{rand_1}, which is error resilient in the common random string model, assuming the adversary is a "message adversary". Loosely speaking, this is done by instantiating the ideal hash using public (and private)

randomness. In Section 6, we show how to instantiate the common random string using private randomness, to obtain a protocol Π_{rand_2} that is error resilient against any "message adversary". Finally, we convert Π_{rand_2} into $\Pi' = (S^A, S^B)$, where Π' is the same as Π_{rand_2}, except that each message is sent encoded with the error correcting code that is resilient to insertions and deletions. In Section 7, we "put it all together" and prove that Π' is the error resilient protocol guaranteed in Theorem 4 above.

2.3 Intuition Behind our Parameters

In what follows, we give an intuitive argument for why our parameters seem to be optimal. We emphasize that this is by no means a proof of optimality, but rather an intuition for where these parameters came from.

As mentioned above, since messages can be of arbitrary length, and since we do not want to blow up the round complexity by much, we must use an error-correcting code that is resilient to (adversarial) insertions and deletions. To date, the maximal rate error-correcting code that is resilient to (adversarial) insertions and deletions is due to Guruswami and Li [18]. This code blows up the message length by $1 + \tilde{O}(\sqrt{\epsilon})$ and is resilient to ϵ fraction of errors.

Moreover, as argued in Section 1.4, in order to ensure a small blowup in communication our error-resilient protocol must be relatively "smooth". In other words, in the error-resilient protocol, after a message of length ℓ we should not send a message much longer than ℓ, since then the adversary will corrupt the length ℓ message and as a result will cause the next long message to be obsolete. Suppose for simplicity (for now) that all the messages are all of the same length ℓ.

Suppose our protocol has blowup $1 + \tilde{O}(\alpha)$ in communication complexity. Thus, we can use the error-correcting code of [18] that blows up the message length by at most $(1 + \tilde{O}(\alpha))$. This code is resilient to α^2 fraction of errors. Thus, by corrupting $\alpha^2\ell$ bits of a message the adversary can make the next round completely obsolete. Since the adversary can corrupt ϵ-fraction of the bits, he can make $\frac{\epsilon}{\alpha^2}$-fraction of the rounds obsolete, which implies that it must be the case that $\frac{\epsilon}{\alpha^2} \leq \alpha$, which in turn implies that $\alpha > \epsilon^{1/3}$.

Note, however, that we cannot assume that all the messages are of the same length since this will blow up the round complexity by too much. And yet, as mentioned above, we do need to assume that the error-resilient protocol is somewhat "smooth", since otherwise the communication complexity will blow up by too much. Thus, we let $\beta > 0$ be a parameter, such that in the error-resilient protocol after a message of length ℓ comes a message of length at most $\beta^{-1}\ell$. Now, an adversary corrupting $\tilde{O}(\alpha^2 \cdot \ell)$ bits of a message can cause $\beta^{-1}\ell$ bits to be obsolete. Thus, intuitively, the parties may waste $\alpha^{-2} \cdot \beta^{-1}$ bits of communication per each corruption. This, together with the fact that the adversary has an ϵ-fraction of corruption budget and the fact that the communication blows up by at most $1 + \tilde{O}(\alpha)$, implies that $\alpha^{-2} \cdot \beta^{-1} \cdot \epsilon < \alpha$, which in turn implies that

$$\alpha^3 \cdot \beta > \epsilon. \tag{1}$$

Therefore, on the one hand we would like to make β as large as possible, to improve the communication rate; on the other hand, increasing β blows up the round complexity. At first it seems that requiring this smoothing condition (i.e., that after a message of length ℓ comes a message of length at most $\beta^{-1}\ell$), will blow up the round complexity by too much. The reason is the following: Consider the real world example, where each message sent by Alice is of length ℓ, and each message sent by Bob is of length 1. Thus, to ensure that Alice is not wasting ℓ bits of communication due to a single error in Bob's message, we need to

make the protocol smooth and have Alice send her message slowly, first sending the first β^{-1} bits, then after getting a bit of approval from Bob, Alice will send the next β^{-2} bits of her message, and so on. Thus, the number of rounds it will take Alice to send her message is roughly $\log_{\frac{1}{\beta}}(\ell)$. This will cause a blowup of roughly $\log_{\frac{1}{\beta}}(\ell)$ to the round complexity, which is way too much.

To avoid this blowup, we want to make sure that after a long message does not come a message which is too short, since short messages may cause a blowup to the round complexity (if the following message is long). However, this should be done while adding at most an α fraction to the communication complexity. Thus, we also smooth the protocol in the "other direction" and require that after a message of length ℓ comes a message of length at least $\alpha\ell$. Thus, going back to our example above, where Alice is talkative (sends messages of length ℓ) and where Bob sends messages of length 1, we first convert this to another protocol where Bob sends messages of length $\alpha\ell$. This does not change the round complexity at all, and changes the communication complexity by at most an α-factor. Now, we smoothen out this protocol, by having Alice, rather than sending her ℓ bit message in "one shot", she will first send $\beta^{-1}\alpha\ell$ bits, then send the next $\beta^{-2}\alpha\ell$ bits, and so on. Note that this will cause a blowup of $\log_{\frac{1}{\beta}}(\alpha^{-1})$ in the round complexity. Since we allow a blowup of at most α' to the round complexity (without taking into account the blowup due to error, or the additive term), we take β so that $\log_{\frac{1}{\beta}}(\alpha^{-1}) \leq \alpha'$, and thus we must take β such that $\beta \leq \alpha^{1/\alpha'}$. This together with Equation (1), implies that

$$\alpha^{3+1/\alpha'} > \epsilon,$$

as in Theorem 4 above.

It remains to explain the additive term in the round blowup and the multiplicative term that depends on the round error-rate ϵ'. For the latter, clearly, if ϵ'-fraction of the rounds were completely corrupted, these rounds need to be redone, and this incurs a blowup of $1+\epsilon'$ to the round complexity. As to the former, suppose the original protocol consists of a short message followed by a very long message, to make this protocol error resilient we will have to blow up the round complexity by essentially $\log_{\frac{1}{\beta}} \mathrm{CC}$, where CC is the communication complexity of the original protocol. This is the reason we have the log additive term in the round complexity.

Finally, we explain why $\epsilon' = \alpha \cdot \mathsf{poly}(\alpha')$. We note that for the purpose of our application (Theorem 1) the exact power of α' is not important. Consider a protocol that consists of a single bit per round. In this case we can effort to add a hash check only every α^{-1} rounds. In this case, the adversary can corrupt the first message of each chunk of α^{-1} rounds, which would render the entire chunk useless. Thus, a corruption of ϵ'-fraction of the rounds, may result with a round blowup of $\epsilon'\alpha^{-1} \leq \alpha'$, which implies that indeed $\epsilon' \leq \alpha\alpha'$.

3 Smooth Protocols

Throughout this section, we refer to "rounds" in a protocol as a one way communication. Namely, the number of rounds in a protocol is equal to the number of messages that are sent in the protocol. We note that in Section 4 we diverge from this interpretation, and refer to "rounds" as a back-and-forth communication between Alice and Bob. This inconsistency allows us to simplify the notation and the presentation. Note that these two interpretations can be interchanged, while incurring a blowup of at most 2 in the round complexity.

Let Π be an arbitrary 2-party protocol. We denote by m_r the messages sent in the r^{th} round in Π. In this section we show how to convert any protocol Π into a *smooth* protocol S^{Π}. In what follows we denote by M_r the message sent in the r^{th} round in S^{Π}.

▶ **Definition 5.** *A protocol is (α, β)-smooth if for every round r the following holds:*

$$\alpha \cdot \max\{|M_{r-1}|, |M_{r-2}|, |M_{r-3}|\} \leq |M_r| \leq \frac{1}{\beta} \cdot \min\{|M_{r-1}|, |M_{r-2}|, |M_{r-3}|\} \quad (2)$$

▶ **Lemma 6.** *For any $\alpha < \frac{1}{4}$ and $\beta \leq \frac{\alpha}{8}$, the following holds: Any protocol Π can be efficiently converted into an (α, β)-smooth protocol S^{Π} such that*

1. $\mathrm{CC}(S^{\Pi}) \leq \mathrm{CC}(\Pi) \cdot (1 + 50\alpha)$.
2. $R(S^{\Pi}) \leq R(\Pi) \cdot (1 + 8\log_{2\beta} \alpha) + 4\log_{\frac{1}{2\beta}} \cdot \mathrm{CC}(\Pi) + 4$
3. *If Π is computationally efficient then so is S^{Π}.*

We defer the proof of Lemma 6 to full version.

▶ Remark 7. In Sections 4, 5, and 6, we show how to convert a smooth protocol into an error-resilient one. Similarly to previous error-resilient protocols in the literature, we will first pad the smooth protocol, and only then we convert the padded protocol into an error-resilient one. However, we will need to pad our protocol in a smooth way. This is done as follows: Suppose we want to pad our protocol with anywhere between L and $2L$ bits of 0's. Suppose that the last message in the smooth protocol is of length ℓ, then we add a message of length $\left\lfloor \frac{\ell}{\beta} \right\rfloor$, followed by a message of length $\left\lfloor \frac{\ell}{\beta^2} \right\rfloor$, and so on, until we add between βL and L bits, after which we add L bits (if we haven't added so already).

Note that such a padding results in a smooth protocol, where the communication complexity increases by at least L and at most $2L$ bits. The number of additional rounds required to do this padding is at most $\log_{\frac{1}{\beta}} L + 1$.

From now on, when we say that we convert a protocol Π to a smooth protocol, we assume that the resulting smooth protocol is padded appropriately.

4 Interactive Coding in the Ideal Hash Model

In this section, we show how to convert any protocol Π into an error resilient protocol, and analyze its properties in the **Ideal Hash Model**. This model assumes the existence of an ideal hash. In our protocol, Alice and Bob check equality of their partial transcripts, by sending to each other hashes of their partial transcripts. In this section, we consider the Ideal Hash Model, where when we analyze the communication complexity of the protocol we do not take into account the length of the hash values, and simply assume that the number of hash collisions is bounded, yet adversarially chosen. (We explain how we bound the number of collisions below). In Sections 5 and 6, we show how to remove this ideal model assumption, by implementing this ideal hash using a real hash function. In these sections, we use hash values that are short enough so the communication blowup is small, and yet we prove that with high probability the amount of hash collisions is bounded.

Moreover, we consider an adversary that either leaves a message (and corresponding hash) intact, or "fully" corrupts it. More precisely, we say that the hash is corrupted if and only if a collision occurs. In the analysis of this ideal error-resilient protocol, we say that a message is corrupted if the adversary corrupts any bit of the message (or if he corrupts the corresponding hash). We let the budget of corrupting a message be the maximum between the original message length, and the corrupted one. In particular, even if the adversary corrupted a single bit of a long message of length n (or if he corrupts only the hash corresponding to this message), we count it as n corruptions. We recall that the reason for this budgeting is that in our actual error-resilient protocol we will apply the error correcting code of Guruswami and Li [18] to each message (and hash) separately. Thus, in order to corrupt a message, the adversary will need to corrupt a constant fraction of the bits in the message. We refer the reader to Section 7 for details.

In what follows, we set

$$\alpha \leq 0.01 \ \ , \ \ \alpha' \leq 1 \ , \ \ d \geq \frac{1}{\alpha} \ \text{ and } \ \beta \leq \min\left\{\alpha^{\frac{1}{\alpha'}}, \frac{1}{5\alpha d^2}\right\}. \tag{3}$$

We assume for simplicity that α^{-1} and β^{-1} are integers. We assume without loss of generality, that the underlying protocol Π is (α, β)-smooth. This is without loss of generality since by Lemma 6, we can convert Π to an (α, β)-smooth protocol while increasing its communication complexity by a multiplicative factor of $(1 + O(\alpha))$, and increase the number of rounds by a multiplicative factor of $(1+O(\alpha'))$ and an additive factor of at most $\log \mathrm{CC}(\Pi)$, as desired.

4.1 The Protocol

We note that this (ideal) protocol is quite similar to the error-resilient protocol of Haeupler [19]. The main difference being that we need to first convert the protocol into a smooth one (whereas the protocol considered in [19] is perfectly smooth since in each round each party sends a single bit to the other party). Moreover, and more importantly, since our protocol is not perfectly smooth, when the parties backtrack, they do not erase the questionable transcript (since the messages in the questionable part may grow exponentially). Instead, the parties keep this transcript as questionable, and enter a "verification" state where they check consistency round-by-round. We note that in [19] the questionable transcript is simply erased.

In what follows, we present the (error-resilient) protocol only from Alice's perspective. Bob's perspective is symmetric. During the (error-resilient) protocol, Alice has a private variable T_A, which she believes to be a prefix of the transcript she is trying to reconstruct. T_A is initiated to $\emptyset$. We denote by m_A the message that Alice sends in the error resilient protocol.

In what follows, we define all the other notations (in addition to m_A and T_A) that are used in the protocol description:

$$S_A, R_A, \ell_A, \ell^+, \ell^-, w_A, R_A^{(1)}, R_A^{(2)},$$

where all of these variables are defined as functions of T_A and m_A.

From now on we think of each round as consisting of consecutive two messages: a message sent by Alice and a following message sent by Bob. We note that this diverges from the way we defined rounds in Section 3, where we thought of each round as containing a single message (sent by one party). The only reason for this inconsistency is that it is more convenient in terms of notations. It is important to note that this is only a notational convenience and does not affect our final result in any way.

For each variable used in our protocol

$$v_A \in \{T_A, m_A, S_A, R_A, \ell_A, \ell^+, \ell^-, w_A, R_A^{(1)}, R_A^{(2)}\},$$

we denote by $v_{A,r}$ the value of v_A that Alice uses when sending her round r message, and we occasionally omit r when it is clear from the context.

- During the protocol Alice has a state

  $$S_A \in \{\text{Simulation}, \text{Verification}\} \cup \mathbb{N}.$$

  Loosely speaking, Alice is in a Simulation state when she believes that the transcript T_A that she is holding is indeed a prefix of the correct transcript.

If $S_A \in \mathbb{N}$ then we say that Alice is in a Correction state. If Alice is in Correction state, then S_A is the first round (in the error-resilient protocol) that Alice has entered this state. Alice enters a Correction state when she thinks her beliefs are wrong (for example, when the hashes indicate that T_A and T_B are inconsistent). During this state, Alice tries to go back to an earlier round in the transcript (corresponding to the original protocol) which she believes to be correct. We denote this round by R_A. Alice will continue the simulation from $T_A[R_A]$, which denotes the truncated transcript of T_A to round R_A. As mentioned above, as opposed to the protocol of Haeupler [19], in our protocol, she does not delete the suffix of T_A, and rather she keeps this suffix as questionable. The reason she does not erase this questionable suffix, is that it may be the correct suffix (and the only reason it is questioned is due to an error), and in this case it may be too expensive to delete and reconstruct, since in our case the messages in the questionable suffix may grow at an exponential rate.

After a Correction state, Alice enters either another Correction state or a Verification state, where she decides whether to completely delete, partially delete, or keep, the questionable suffix. After the Verification state (assuming there were no errors), Alice enters Simulation state again.

We define $r - S_A$ to be zero, when $S_A \in \{\text{Simulation}, \text{Verification}\}$.

- As mentioned above, R_A denotes the round in T_A that Alice simulates. If $S_A = \text{Simulation}$ then R_A is equal to the number of rounds in T_A.
- Let $m_{A,r}$ be the message that Alice sends in round r (of the error resilient protocol), and let $\ell_{A,r}$ denote its size. Let $m_{B,r}$ be the message that Alice received from Bob in round r, and let $\ell_{B,r}$ denote its size. We define $\ell_{\max,r} = \max\{\ell_{A,r}, \ell_{B,r}\}$. Note that if Bob's message was corrupted then $\ell_{B,r}$ may be arbitrarily large. However, our (error-resilient) protocol has the property that if Bob's message was not corrupted then $\ell_{B,r} \leq \frac{\ell_{A,r}}{\beta}$.
  We define

$$\ell_r^+ = \min\left\{\frac{\ell_{A,r}}{\beta}, 2\ell_{\max,r}\right\}$$

  and

$$\ell_r^- = \min\left\{\frac{\ell_{A,r}}{\beta}, \max\left\{\beta^{-1}, \alpha\ell_{\max,r}\right\}\right\}.$$

- Let $w_{A,r}$ be $2^{\lfloor \log(r-S_A)\rfloor}$ if $S_{A,r} \in \mathbb{N}$, and let $w_{A,r}$ be 0 otherwise. In other words, $w_{A,r}$ is the number of rounds that the party has been in Correction state, rounded to the closest power of two.
- If $w_A = 0$ then let $R_A^{(1)} = R_A$. Otherwise, let $R_A^{(1)} < R_A$ be maximal that divided w_A.
- $R_A^{(2)} \triangleq R_A^{(1)} - w_A$.
- In the protocol, at each round r, Alice sends hashes to Bob if and only if $r = 0 \pmod d$ or $\ell_{A,r} \geq d$, in which case she sends five hashes, one hash corresponding to each of the following strings:

$$\left(T_A[R_A], T_A[R_A+1], T_A[R_A^{(1)}], T_A[R_A^{(2)}], S_A\right).$$

  We note that if R_A is equal to the number of rounds in T_A, then Alice will not know the partial transcript $T_A[R_A+1]$. In this case we define $T_A[R_A+1] = T_A[R]$.

Alice in round r. Upon receiving a message from Bob, parse the message as

$$(m_{B,r-1}, H(T_{B,r-1}[R_{B,r-1}]), H(T_{B,r-1}[R_{B,r-1}+1]),$$
$$H(T_{B,r-1}[R^{(1)}_{B,r-1}]), H(T_{B,r-1}[R^{(2)}_{B,r-1}]), H(S_{B,r-1})).$$

We assume that in this ideal model, parsing is easy. When we implement this ideal hash function in Section 5, we will make sure that indeed Alice will be able to parse correctly (assuming the message was not corrupted). Denote the size of $m_{B,r-1}$ by $\ell_{B,r-1}$.

1. If $S_{A,r-1} =$ Simulation then do the following:

   a. If a hash was sent by Bob (i.e., if $\ell_{B,r-1} \geq d$ or d divides $r-1$) then check that

   $$H(S_{B,r-1}) = \text{Simulation} \quad \text{and} \quad H(T_{A,r-1}[R_{A,r-1}]) = H(T_{B,r-1}[R_{B,r-1}]).$$

   b. Check that the (partial) transcript $(T_{A,r-1}[R_{A,r-1}], m_{A,r-1}, m_{B,r-1})$ satisfies the (α, β)-smoothness condition.
   c. If one of these conditions does not hold then let
      - $m_{A,r} = 0^{\ell^-_{r-1}}$
      - $S_{A,r} = r$
      - $(T_{A,r}, R_{A,r}) = (T_{A,r-1}, R_{A,r-1})$.
   d. Else, set
      - $T_{A,r} = (T_{A,r-1}, m_{A,r-1}, m_{B,r-1})$
      - $R_{A,r} = R_{A,r-1} + 1$
      - $m_{A,r} = \Pi(T_{A,r})$
      - $S_{A,r} = S_{A,r-1} =$ Simulation.

2. If $S_{A,r-1} =$ Verification then check if all the following conditions hold:

   a. $|m_{B,r-1}| \geq \beta^{-1}$.
   b. $H(S_{A,r-1}) = H(S_{B,r-1})$
   c. $H(T_{A,r-1}[R_{A,r-1}]) = H(T_{B,r-1}[R_{B,r-1}])$.

   If one of these conditions does not hold then let
   - $m_{A,r} = 0^{\ell^-_{r-1}}$
   - $S_{A,r} = r$
   - $(T_{A,r}, R_{A,r}) = (T_{A,r-1}, R_{A,r-1})$.

   Else, do the following:

   a. If number of rounds in $T_{A,r-1}$ is greater than $R_{A,r-1}+1$, and

   $$H(T_{A,r-1}[R_{A,r-1}+1]) = H(T_{B,r-1}[R_{B,r-1}+1]),$$

   then let
   - $m_{A,r} = 0^{\ell^-_{r-1}}$
   - $R_{A,r} = R_{A,r-1} + 1$
   - $(S_{A,r}, T_{A,r}) = (S_{A,r-1}, T_{A,r-1})$.

   b. Else, if $m_{A,r-1} = m_{B,r-1} = 1^\ell$ for some $\ell > |T_{A,r-1}| - |T_{A,r-1}[R_{A,r-1}]|$, then set
   - $T_{A,r} = T_{A,r-1}[R_{A,r-1}]$
   - $R_{A,r}$ be the number of rounds in $T_{A,r}$
   - $m_{A,r} = \Pi(T_{A,r})$
   - $S_{A,r} =$ Simulation.

c. Else, if $\ell^+_{r-1} > |T_{A,r-1}| - |T_{A,r-1}[R_{A,r-1}]|$ then let
- $m_{A,r} = 1^{\ell^+_{r-1}}$
- $(T_{A,r}, R_{A,r}, S_{A,r}) = (T_{A,r-1}, R_{A,r-1}, S_{A,r-1})$.

d. Else, let
- $m_A = 0^{\ell^+_{r-1}}$
- $(T_{A,r}, R_{A,r}, S_{A,r}) = (T_{A,r-1}, R_{A,r-1}, S_{A,r-1})$.

3. Else, do the following:

a. Compute the values v^0_r, v^1_r, v^2_r as follows:
If $H(S_{A,r-1}) \neq H(S_{B,r-1})$ then set $v^0_r \leftarrow v^0_{r-1} + 1$.
Else, if $H(T_{A,r-1}[R^{(1)}_{A,r-1}]) \in \left\{H(T_{B,r-1}[R^{(1)}_{B,r-1}]), H(T_{B,r-1}[R^{(2)}_{B,r-1}])\right\}$ then set $v^1_r \leftarrow v^1_{r-1} + 1$.
Else, if $H(T_{A,r-1}[R^{(2)}_{A,r-1}]) \in \left\{H(T_{B,r-1}[R^{(1)}_{B,r-1}]), H(T_{B,r-1}[R^{(2)}_{B,r-1}])\right\}$ then set $v^2_r \leftarrow v^2_{r-1} + 1$.

b. If $r - S_{A,r-1}$ is not a power of 2,[9] then set
- $m_{A,r} = 0^{\ell^-_{r-1}}$
- $(T_{A,r}, R_{A,r}, S_{A,r}) = (T_{A,r-1}, R_{A,r-1}, S_{A,r-1})$.

c. Else, if $v^0_{r-1} > \frac{1}{2}(r - S_{A,r-1})$ then let
- $S_{A,r} = r$
- $m_{A,r} = 0^{\ell^-_{r-1}}$
- Set $v^0_r = v^1_r = v^2_r = 0$.

d. Else, if $v^1_r > \frac{1}{4}(r - S_{A,r-1})$ then let
- $S_{A,r} =$ Verification
- $R_{A,r} = R^{(1)}_{A,r-1}$
- $m_{A,r} = 0^{\ell^-_{r-1}}$
- Set $v^0_r = v^1_r = v^2_r = 0$.

e. Else, if $v^2_r > \frac{1}{4}(r - S_{A,r-1})$ then let
- $S_{A,r} =$ Verification
- $R_{A,r} = R^{(2)}_{A,r-1}$
- $m_{A,r} = 0^{\ell^-_{r-1}}$
- Set $v^0_r = v^1_r = v^2_r = 0$.

f. Else, set $v^1_r = v^2_r = 0$, and let
- $m_{A,r} = 0^{\ell^-_{r-1}}$
- $(v^0_r, R_{A,r}, S_{A,r}, T_{A,r}) = (v^0_{r-1}, R_{A,r-1}, S_{A,r-1}, T_{A,r-1})$.

Send $m_{A,r}$, and if $r = 0 (\text{mod } d)$ or $|m_{A,r}| \geq d$ then append to $m_{A,r}$ also

$$\left(H(T_{A,r}[R_{A,r}]), H(T_{A,r}[R_{A,r}+1]), H(T_{A,r}[R^{(1)}_{A,r}]), H(T_{A,r}[R^{(2)}_{A,r}]), H(S_{A,r})\right).$$

Remark. Bob behaves identically to Alice, except in Steps 1 and 2b, when Bob computes his next message corresponding to the underlying protocol Π, he computes it by $m_{B,r} = \Pi(T_{B,r}, m_{A,r})$, whereas recall that Alice computed it by $m_{A,r} = \Pi(T_{A,r})$.

[9] Recall that we define $r - S_A = 0$ if $S_A \in \{\text{Simulation}, \text{Verification}\}$, and we consider 0 to be power of 2.

4.2 Analysis

Terminology. In what follows, we introduce terminology that we use in the analysis.

We allow the adversary to create collisions in the ideal hash function, in which case we say that the hash was corrupted. We say that a message is corrupted if the adversary corrupts any bit of the message, or corrupts the associated hash. We define the budget of corrupting a message m to be the maximum between the length of m and the length of the corrupted version of m. Thus, even if the adversary corrupts a few bits of a long message of length n (or corrupts the associated hash), then we count it as n corruptions. On the other hand, if the adversary corrupted a single bit message by converting it into a long n-bit message, then we count it as n corruptions.

We analyze the correctness of the (error-resilient) protocol assuming a bound on these message corruptions.

▶ **Definition 8.** *We say that the corrupted messages have* volume e *if the sum of lengths of corrupted messages (where each such length is the maximum between the length of the original message and the length of the corrupted version of it) is e.*

Using this terminology we prove the following theorem.

▶ **Theorem 9.** *Let $\Pi = (A, B)$ be any (α, β)-smooth protocol, and let $\Pi' = (S^A, S^B)$ be the simulated protocol defined above. Let $\mathcal{A}$ be any adversary in Π', who corrupts at most e' messages of total volume of at most e. Then, the protocol Π', executed with the adversary $\mathcal{A}$, denoted by $\Pi'_{\mathcal{A}}$, satisfies the following.*

1. $CC(\Pi'_{\mathcal{A}}) \geq t_{\min}$*, where $t_{\min}$ is a lower bound of the communication of any instance of* Π.
2. $CC(\Pi'_{\mathcal{A}}) \leq CC(A, B) + 18\beta^{-1}e + 20d\beta^{-1}e'$.
3. $R(\Pi'_{\mathcal{A}}) \leq R(A, B) + 906d \log \frac{1}{\beta} e'$.
4. *The parties outputs transcripts of size at most* $\mathrm{CC}(\Pi'_{\mathcal{A}})$ *that agree with* Π *on the first*

$$CC(\Pi'_{\mathcal{A}}) - 18\beta^{-1}e - 20d\beta^{-1}e' \ ,$$

   many bits.
5. *S is a polynomial time oracle machine.*

▶ Remark 10. We will apply Theorem 9 with an adversary $\mathcal{A}$ that corrupts at most $e' = O(\min\{\epsilon' \cdot R(\Pi'_{\mathcal{A}}), \frac{\epsilon}{d} CC(\Pi'_{\mathcal{A}})\})$ messages of total volume at most $e = O(\epsilon \cdot CC(\Pi'_{\mathcal{A}}))$, where $\epsilon \leq O(\alpha \cdot \beta)$ and $\epsilon' \leq \frac{\alpha'}{d \cdot \log \beta^{-1}}$. Thus,

$$\begin{aligned}
&CC(\Pi'_{\mathcal{A}}) \leq \\
&CC(A, B) + 18\beta^{-1}e + 20d\beta^{-1}e' \leq \\
&CC(A, B) + 0(\beta^{-1}\epsilon \cdot CC(\Pi'_{\mathcal{A}})) + O\left(d\beta^{-1}\frac{\epsilon}{d} CC(\Pi'_{\mathcal{A}})\right) \leq \\
&CC(A, B)(1 + O(\alpha)),
\end{aligned}$$

and

$$\begin{aligned}
&R(\Pi'_{\mathcal{A}}) \leq \\
&R(A, B) + 906d \log \frac{1}{\beta} e' \leq \\
&R(A, B) + O(d \log \frac{1}{\beta} \epsilon' \cdot R(\Pi'_{\mathcal{A}})) \leq \\
&R(A, B)(1 + O(\alpha')),
\end{aligned}$$

as desired.

Moreover, we will apply this theorem with a protocol Π which is padded by $18\beta^{-1}e + 20d\beta^{-1}e' = O(\alpha \cdot CC(\Pi'_{\mathcal{A}}))$ zeros. Thus, Item 4 from Theorem 9 implies correctness.

For example, one can set $\alpha = O(\min\{\sqrt{\epsilon}, (\epsilon')^{1/3}\})$, and set $\beta = O(\alpha)$, $d = \frac{1}{\alpha}$, $\alpha' = 1$, to obtain an error resilient protocol in the ideal hash model with constant blowup in round complexity and $1 + O(\alpha)$ blowup in communication complexity.

We defer the proof of Theorem 9 to full version.

5 Hash Implementation with Shared Randomness

Recall that in Section 4, we presented an interactive coding scheme with the desired guarantees, in the ideal hash model, where we assume that the number of hash collisions is bounded, and where the budget for making a collision is proportional to the message length (where the message length is the maximum between the length of the message that was sent and the corrupted version of it). We denote this ideal protocol by Π.

In this section, we show how to implement the ideal hash with a real hash function. Loosely speaking, given a hash function h, we convert the protocol Π to the protocol Π^h which is identical to Π, where the ideal hash function is replaced by h. In order to maintain the desired efficiency and error-resilience guarantees, we need to ensure that, on the one hand, these hash values are not too long; and on the other hand there are not too many hash collisions (i.e., that these hashes form a good equality test). To ensure the latter condition holds, it is easy to see that we cannot use a single (deterministic) hash function. Instead we use a *family* of *randomized* hash functions.

We construct a function family $\mathcal{H} = \{h_x\}$, where each hash function h_x is associated with a (possibly long) seed x. In this section, we consider the *shared randomness model*, where the parties are allowed to share a (possibly long) random string. In Section 6 we show how to eliminate the need for shared randomness.

In this section we assume that the shared randomness is as long as we need. In particular, we use a different hash function (i.e, a different seed) for each equality query. Since the length of the protocol is adaptive and not a priori bounded[10], the length of the common random string is also not a priori bounded. We assume that there is a separate segment of the common random string for each round r, and each such segment contains five hash seeds, since in Π, in rounds that a party sends an ideal hash, the party sends five ideal hashes.

We emphasize that the shared randomness (and in particular the seeds) are known to the adversary. Therefore the adversary, given a seed x can try to skew the protocol and cause the parties to send many messages whose hashes collide. To get around this, we construct a hash family, where each h_x is a *randomized* hash function. When a party sends a hash of a value V, the party will choose randomness S and will send $(S, h_x(V, S))$. On the one hand, the randomness S needs to be short, since otherwise this will blowup the communication complexity by too much. On the other hand, the adversary cannot predict S, and thus will not be able to skew the messages of the parties towards ones which the hashes collide. We note that a similar idea of using a randomized hash function was used by Haeupler [19], for the sake of improving the rate of his interactive coding scheme.

Before presenting our randomized hash family, we start with some preliminaries.

[10] In Section 6, we convert any such protocol in the *unbounded* shared randomness model into one that uses only private randomness.

5.1 Preliminaries

Chernoff bounds.

▶ **Lemma 11.** *For any $N \in \mathbb{N}$, and any N independent Bernoulli random variables $X_1, \ldots, X_N$, each with mean $\leq \gamma$, it holds that*

$$\Pr[\sum_{i=1}^{N} X_i > 2\gamma N] \leq e^{-\frac{1}{3}\gamma N}.$$

▶ **Definition 12.** *A distribution D over $\mathbb{F}_2^n$ is* δ-bias *if for any $v \in \mathbb{F}_2^n \setminus \{0^n\}$, we have that*

$$\left| \Pr_{x \sim D} \left[\sum_{i=1}^{n} v_i x_i = 0 \right] - \frac{1}{2} \right| \leq \delta \ .$$

▶ **Lemma 13** ([25])**.** *There exists an absolute constant $C \in \mathbb{N}$ and an efficiently computable function $G : \{0,1\}^* \to \{0,1\}^*$ such that for any size k and a uniformly random string $S \in \{0,1\}^{Ck}$, we have that $G(S) \in \{0,1\}^{2^k}$ is a 2^{-k}-biased distribution of length 2^k.*

▶ **Lemma 14** (6.3 from [19])**.** *There exists a hash family $\mathcal{F} = \{\mathcal{F}_L\}_{L \in \mathbb{N}}$, such that for every $L \in \mathbb{N}$ it holds that $\mathcal{F}_L = \{f_x\}_{x \in \{0,1\}^{2L}}$, and for every $x \in \{0,1\}^{2L}$, $f_x : \{0,1\}^{\leq L} \to \{0,1\}$. Moreover, for any $k \in \mathbb{N}$ and for any vectors $V_1^A, ..., V_k^A, V_1^B, ..., V_k^B \in \{0,1\}^{\leq L}$ the following holds:*

1. *For uniform $x = (x_1, \ldots, x_k) \in (\{0,1\}^{2L})^k$ it holds that for each $i \in [k]$, the probability that $f_{x_i}(V_i^A) = f_{x_i}(V_i^B)$ is $\frac{1}{2}$ whenever $V_i^A \neq V_i^B$, and 1 whenever $V_i^A = V_i^B$. Moreover, for each $i \in [k]$ these probabilities are independent.*
2. *For δ-biased distribution $x = (x_1, \ldots, x_k) \in (\{0,1\}^{2L})^k$, it holds that the distribution*

$$\left(\mathbf{1}_{f_{x_1}(V_1^A) = f_{x_1}(V_1^B)}, \ldots, \mathbf{1}_{f_{x_k}(V_k^A) = f_{x_k}(V_k^B)} \right)$$

is δ-close to the case where x is uniform.

5.2 Our Hash Function

We are now ready to construct our family of randomized hash functions. We first define the randomized hash family $\mathcal{H}' = \{h'_x\}$, which uses the hash family $\mathcal{F}$ from Lemma 14. Recall that the hash values of $\mathcal{H}'$ should not be too long, since this will result in a large blowup in communication complexity.

In our construction, as opposed to previous constructions [2, 3, 19], the length of each hash value depends not only on the length of the message it is appended to, but it also depends on the length of the *entire* communication up until the point that the hash was sent. We note that if we were only concerned with the communication blowup and were not concerned with the round blowup, then we could have the length of the hash value depend only on the length of the message it is sent with (in similar spirit to prior work). However, as we argue below, in order to ensure a constant blowup in round complexity, we must allow the length of the hash value to also depend on the length of the entire history. This is illustrated in the following example: Suppose that a short message is sent, and prior to this short message were a few very long messages (in a way that satisfies the smoothness criterion). By corrupting a few long messages, the adversary can cause a hash collision in many short messages (with hash), which will result with a large blowup to the round complexity.

Hence, we allow the length of the hash value, not only to depend on the length of the message it is appended to, but also to depend on the length of the communication history. Note that the parties do not necessarily agree on the history length even if no errors occur in the current round, since the adversary may insert and delete bits throughout the protocol, in which case they will fail to parse the message and hash pair correctly.

Thus, we define the hash family $\mathcal{H} = \{h_x\}$, where the output of h_x includes the output of h'_x, the randomness used by h'_x (which is needed in order check for equality), and also the length of the hash value. Namely, we define

$$h_x(V) = (S, h'_x(V;S), 1 \cdot 0^w),$$

where S is the (private) randomness used by h'_x, and w is the length of $H'_x(V;S)$.

We next define h'_x. As we mentioned, the length of the hash values (denoted by w) may differ from one round to the next, as they depend on the communication complexity so far, and on the length of the current message sent. We will specify how w is defined below. But we first, define h'_x assuming w is known.

The seed x is random in $\left(\{0,1\}^{2L}\right)^L$, where we assume that L is greater than the input V (which is bounded by the communication complexity of the protocol up until the point where the hash is sent). For any $y = (y_1, \ldots, y_L) \in \left(\{0,1\}^{2L}\right)^L$ and for any $k \leq L$, let

$$f^k_y(V) = (f_{y_1}(V), ..., f_{y_k}(V)).$$

where $\mathcal{F} = \{f_y\}$ is the hash family from Lemma 14. The randomness for h'_x is denoted by S and is of size $2C \cdot w$. Let h'_x be the randomized hash function, that takes as input a variable V, randomness S, and outputs

$$h'_x(V;S) = f^w_{G(S)}(Z),$$

where

$$Z = \begin{cases} (f_x^{2^w}(V), 0) & \text{if } |V| \geq 2^w \\ (V, 1) & \text{if } |V| < 2^w \end{cases}$$

In what follows we show how the length w of the hash values are chosen. To this end we need to define the following variables with respect to a certain round r.

- Let Q^A be the set of all rounds r in which Alice send a hash to Bob.
  Note that these are exactly the set of rounds r such that r divides d or Alice send a message of length $\geq d$.
  We define Q^B analogously.
- Let a^A_r be the number of messages that Alice sent with a hash until (and including) round r. Namely,

  $$a^A_r = |\{r' \leq r : r' \in Q^A\}|.$$

  We define a^B_r analogously.
- Let t^A_r be all the communication received by Alice until (and including) round r. We define t^B_r analogously.
- For every r, if Alice sends the message in round r then define

  $$u_r = \max_{r' \in Q^A \cap [r-1]} \log \frac{t^A_r - t^A_{r'}}{a^A_r - a^A_{r'}}.$$

  The definition is analogous in the case that Bob sends the message in round r.

- For every round r, let ℓ_r denote the length of the message to be sent in round r, and let

$$w_r = \lceil \alpha \ell_r \rceil + \lceil u_r \rceil + 9 \left\lceil \log \frac{1}{\gamma} \right\rceil + 6,$$

where $\alpha, \gamma > 0$ are parameters of the scheme, where $\gamma < \alpha$ (it will be instructive to think of $\gamma = \epsilon$, where ϵ is the corruption budget of the adversary, and of α as the communication blowup in the error-resilient protocol).

5.3 Analysis

We denote by E the set of all messages $m_{A,r}$ or $m_{B,r}$ that were not corrupted but had a hash associated with them that formed a hash collision. Recall that for any set of messages T, we denote by $|T|$ the volume of T (i.e., the number of bits in T), and we denote by $|T|'$ the number of messages in T.

▶ **Lemma 15.** *Fix $\epsilon < 0.0005$. The protocol $\Pi^{\mathcal{H}}$ defined in Section 5.2 satisfies the following: If $\Pi^{\mathcal{H}}$ consists of $\leq t$ bits and $\leq r$ rounds, then for any adversary that corrupts messages with total volume at most ϵt, we get that*

1. *With probability $\geq 1 - 10 \cdot e^{-\frac{\alpha\gamma}{3 \log \frac{1}{\gamma}} t}$ (over the common and private randomness),*

$$|E| \leq 20\gamma t.$$

2. *With probability $\geq 1 - 10e^{-\frac{2}{3}\gamma r}$ (over the common and private randomness),*

$$|E|' \leq 70\gamma r.$$

5.4 Communication Bound

In this section we will bound the blowup of the communication of $\Pi^{\mathcal{H}}$, defined in Section 5.2. To this end, fix any adversary $\mathcal{A}$ for the protocol $\Pi^{\mathcal{H}}$, that corrupts at most e' messages of total volume at most e. We define a corresponding adversary $\mathcal{D}$ for the protocol Π, that corrupts at most e' message of total volume at most e, as follows:

The adversary $\mathcal{D}$ sends the exact same messages as $\mathcal{A}$ does, excluding the hash values. Recall that for each message in $\Pi^{\mathcal{H}}_{\mathcal{A}}$, the part that belongs to the hash value is well-defined by the suffix of the message $1 \cdot 0^w$, and hence the adversary $\mathcal{D}$ is well defined.

▶ **Lemma 16.**

$$\mathrm{CC}(\Pi^{\mathcal{H}}_{\mathcal{A}}) \leq (1 + 50C\alpha)\,\mathrm{CC}(\Pi_{\mathcal{D}}) + e + 600C \log \frac{1}{\gamma} \cdot k$$

where C is the universal constant from Lemma 13, and k is the number of rounds with hash in $\Pi_{\mathcal{D}}$.

The proof of this lemma is deferred to full version.

6 Hash Implementation with Private Randomness

In this section we show how to implement the ideal hashes in protocol Π, defined in Section 4, without resorting to shared randomness, but rather using only private randomness. To this end, we will slightly modify the protocol Π, into a new protocol Π'.

High-level overview of Π'. Lets first recall the approach used in previous works [2, 19]. In these works, the long shared randomness is replaced with δ-biased randomness, where $\delta = 2^{-\alpha t}$ and t is a bound on the communication complexity. Such δ-biased randomness can be generated using only $O(\alpha t)$ random bits. Hence, in previous works, these $O(\alpha t)$ bits of randomness are sent in advance (using an error correcting code). If we indeed had a bound t on the communication complexity, then this idea would work, as explained below.

Recall that the randomness is used for equality testing. From Lemma 14, we know that for any oblivious adversary (i.e., one that is independent of the randomness), the fraction of collisions in the case where the seed is random is δ-close to the fraction of collisions in the case where the seed is δ-biased. Denoting by N the number of possible oblivious adversaries, and by taking a union bound over all possible oblivious adversaries, we conclude that the probability that there exists an oblivious adversary that causes "too many" hash collisions in the case where the seed is δ-based is bounded by the same probability where the seed is truly random plus an additive term of δN. We note that

$$N \leq 2^{H(\epsilon)t} \cdot 4^{\epsilon t} = 2^{O(\epsilon \log \frac{1}{\epsilon} t)},$$

and thus

$$\delta N = 2^{-\alpha t} \cdot 2^{O(\epsilon \log \frac{1}{\epsilon} t)} = 2^{-\Omega(\alpha t)}.$$

Therefore, the probability that there exists an oblivious adversary that causes "too many" hash collisions is at most $2^{-\Omega(\alpha t)}$. Note that we can view any (non-oblivious) adversary as one that chooses an oblivious adversary as a function of the public randomness, and runs this oblivious adversary. Therefore, we conclude that for any (non-oblivious) adversary the probability that there are "too many" hash collisions is bounded by $2^{-\Omega(\alpha t)}$.

However, in our setting, we do not have an a priori bound on the communication complexity. In particular, if we replace the CRS with δ-biased randomness, where $\delta = 2^{-\alpha t}$ for some t, and if the adversary has a corruption budget of more than $O(\alpha t)$ bits (i.e., the communication complexity is larger than $\frac{\alpha t}{\epsilon}$), then our protocol is no longer safe. We overcome this problem by sending more randomness as the communication complexity increases.

More specifically, the parties start by assuming that the communication complexity is some small $t_{\min}$, where $t_{\min}$ is a lower bound on the communication complexity. So, the protocol starts when one of the paries, say Alice, chooses a random string $s \in \{0,1\}^{\alpha t_{\min}}$, and sends it to Bob.[11]

Once $t_1 \geq \frac{\alpha t_{\min}}{\epsilon}$ bits are sent in the protocol, the safety of this randomness could be compromised, since the adversary has enough budget to compromise $\alpha t_{\min}$ bits. Hence, each party, before sending its message, will check whether sending this message will cause the communication complexity to exceed $\frac{\alpha t_{\min}}{\epsilon}$. If so, then instead of sending the message, the party will send new randomness. This time, the party will choose at random s_2 of size $\alpha t_1 - |s_1|$ and send (s_1, s_2). If this randomness is inconsistent with the first randomness sent (s_1) then the party receiving the randomness aborts.

Once the communication complexity is $t_2 \geq \frac{\alpha t_1}{\epsilon}$, again the safety of the previous randomness could have been compromised, and hence as above, if a party is about to send a message that will cause the communication complexity to exceed $\frac{\alpha t_1}{\epsilon}$, then instead of sending the message, the party will choose at random s_3 such that $|s_1| + |s_2| + |s_3| = \alpha t_2$,

[11] As before, we ignore the error-correcting code, since we consider only message adversaries, that corrupt messages as opposed to bits, and the budget for corrupting a message is the length of the message (or the length of the corrupted message, whichever is longer).

and will send (s_1, s_2, s_3), etc. If at any point the randomness received is inconsistent with the previous random string then the party aborts. We refer to these special messages that transmit randomness by *system messages*.

There is a slight problem with this idea: How does Bob know which message sent by Alice corresponds to a message in the initial protocol Π, and which is a system message? We fix this problem by appending 1's to system messages, and appending 0's to messages corresponding to Π. However, recall, that we do not want to blowup the communication complexity. Hence we only append these bits to long messages. This is enough, since system messages are always long.

Note that according to our protocol the parties first receive randomness s_1, then they receive new randomness (s_1, s_2), and so on. We ensure that if at any point, a system message was decoded incorrectly, then eventually the paries will abort, and "catch" the adversary with injecting too many errors. This guarantee simplifies the analysis: Either at some point a system message was decoded incorrectly, in which case the adversary is "caught" with injecting too many errors, or all the parties always agree on the randomness, in which case correctness follows from the correctness of the underlying protocol in the shared randomness model.

To ensure that indeed the parties will always notice when a system message was corrupted, we add to the system message the rounds $r_1, \ldots, r_k$ in which system messages were sent. This is done to circumvent the case where the message (s_1, s_2) was corrupted and converted into a protocol message, and a few rounds later a protocol message was corrupted and converted into the same system message (s_1, s_2). If we do not include the round number then the parties may never notice that there was a point in the protocol where they did not agree on the shared randomness. In order to avoid dealing with such cases, we simply include the round numbers of the system messages.

Finally, we notice that even though we ensure that the parties always agree on the shared randomness (assuming the adversary does not inject too many errors), there is still a subtle issue. Note that the first random string s_1 is δ-biased for $\delta = 2^{-\alpha t_{\min}}$. As we saw in previous work, this suffices if the number of oblivious adversaries, restricted to the first $t_{\min}$-bits, is bounded by $2^{O(\alpha t_{\min})}$. However, in our setting, since the total communication may be significantly larger than $t_{\min}$, the number of such oblivious adversaries can be as large as $2^{t_{\min}}$, in which case the number of rounds with hash collisions can be large. To overcome this problem, we ensure that in the first $t_{\min}$ bits of communication, the adversary cannot inject too many errors (without being "caught"). This is done by re-sending the first $t_{\min}$ bits after $t_{\min}/\alpha$ bits of the protocol were transmitted, and the parties abort if these $t_{\min}$ bits are ϵ/α-far from the first $t_{\min}$ bits of the transcript. More precisely, to each system message sent after t bits of the protocol were communicated, we append the first αt bits of the transcript.

In what follows we present our protocol Π'. For the sake of simplicity, after each system message is sent, the party receiving a system message replies with an "echo" message, by simply repeating the system message. The purpose of this "echo" message is simply to allow the other party to send his protocol message (which he didn't have the budget to send in the previous round).

The protocol Π'. Let $b > 2$, and let $\alpha \leq \frac{1}{3200C}$, where C is the constant defined in Lemma 13. Fix any $d \in \mathbb{N}$ and $\gamma > 0$ such that

$$\gamma \leq \min\left\{\frac{1}{d}, 2^{-b}\right\} \quad \text{and} \quad d \geq \frac{\log \frac{1}{\gamma}}{\alpha}. \tag{4}$$

For convenience the reader can think of $b = 2$ and $\gamma = \frac{1}{d}$.

Let Π be the protocol, in the ideal hash model, defined in Section 4, instantiated with α and d as above, and with any $\alpha' > 0$. Let $t_{\min}$ be a lower bound on the communication complexity of Π, where

$$t_{\min} \geq \max\{\alpha^{-2}, 250C\alpha^{-1}\log d\}, \tag{5}$$

and let

$$W = \alpha t_{\min}.$$

Let $\mathcal{H}$ be the hash family defined in Section 5. The protocol Π' makes oracle access to the protocol $\Pi^{\mathcal{H}}$ (defined in Section 5).

In protocol Π', each party maintains a transcript T initialized to $\emptyset$, an integer k initialized to 0, k strings $s_1, ..., s_k$, k partial transcripts $P_1, \ldots, P_k$, and k rounds $r_1, ..., r_k$ that will be determined during the protocol. Intuitively, T is the transcript corresponding to Protocol $\Pi^{\mathcal{H}}$, $s_1, \ldots, s_k$ are k seeds that are used to generate the hash function implementing the ideal hash, and $r_1, \ldots, r_k$ correspond to rounds in $\Pi^{\mathcal{H}}$ where the common randomness changes. Similarly to Π (and $\Pi^{\mathcal{H}}$), in Π' we interpret the (partial) transcripts as strings.[12] .

In Π', if a party aborts, it always waits until at least $t_{\min}$ bits are sent before aborting the protocol, so as to fulfill the requirement that the communication complexity of Π' is at least $t_{\min}$.

In the first round of Π' Alice does the following:

1. Choose $s_1 \in_R \{0,1\}^W$, and let $k = 1$, $r_1 = 0$, and $P_1 = \emptyset$.
2. Send $(s_1, P_1, r_1, 1)$.

We next describe the protocol from Alice's point of view, given her private state

$$(T, k, s_1, \ldots, s_k, r_1, \ldots, r_k, P_1, \ldots, P_k).$$

Bob's view is symmetric (by switching between A and B). Upon receiving a message m_B, Alice does the following

1. If in the previous round Alice computed her message in step 4(e)ii of the protocol (or if the previous round was the first round of the protocol) then check that m_B is an echo of (i.e., equal to) the message sent by Alice in the previous round. If not then halt, and otherwise goto Step 4c.
2. Otherwise, denote $\ell = |m_B|$.
3. If $\ell \geq b^k W$ and the least significant bit of m_B is 1, then do the following:

   a. If there exists $s, P, r \in \{0,1\}^{b^k W}$, where r is a binary representation of $|T|'$, such that

   $$(s_1, ..., s_k, s, P_1, ..., P_k, P, r_1, ..., r_k, r, 1) = m_B,$$

   and such that P can be obtained from a prefix of T by corrupting messages of volume at most $\frac{|P|}{bd\log d}$, then define $s_{k+1} = s$, define $r_{k+1} = r$, $P_{k+1} = P$, update $k \leftarrow k+1$, and send (an echo message) m_B.

   b. Else, abort the protocol.

[12] This is done by standard encoding, where after each bit of the transcript we add a bit that represents whether the message ended or not. Thus, a transcript of length ℓ can be described by a string of length 2ℓ.

4. Else, do the following:
   a. If $\ell \geq b^k W$ then let m'_B be the message m_B when the least significant bit of m_B is truncated. Otherwise, let $m'_B = m_B$.
   b. Update $T \leftarrow T \cup \{m'_B\}$
   c. Define $m_A = \Pi^{\mathcal{H}}(T)$, using $x = x(s_1, \ldots, s_k, r_1, \ldots, r_k)$ as the shared randomness, where the exact function x is defined later. If there is no m_A to send then abort.
   d. If $|T \cup m_A| < \frac{b^k W}{400C\alpha}$ then do the following:
      i. Update $T \leftarrow T \cup \{m_A\}$.
      ii. Let $\ell = |m_A|$.
      iii. If $\ell < b^k W$ then send m_A, and otherwise send $(m_A, 0)$.
   e. Else,
      i. Let $s_{k+1}, P_{k+1}, r_{k+1} \in \{0,1\}^{b^k W}$ such that s_{k+1} is a uniformly chosen random string, P_{k+1} consists of the first $b^k W$ bits of T (where T is viewed as string) and r_{k+1} is a binary representation of $|T|'$.[13]
      ii. Send

$$(s_1, \ldots, s_k, s_{k+1}, P_1, \ldots, P_k, P_{k+1}, r_1, \ldots, r_k, r_{k+1}, 1).$$

      iii. $k \leftarrow k + 1$.

▶ **Theorem 17.** *Fix any adversary $\mathcal{A}$ for Π' that corrupts $\epsilon' R(\Pi'_{\mathcal{A}})$ of the messages of total volume at most $\epsilon \mathrm{CC}(\Pi'_{\mathcal{A}})$, for $\epsilon \leq \frac{\alpha}{bd \log d}$, where $\Pi'_{\mathcal{A}}$ denotes the protocol Π' executed with the adversary $\mathcal{A}$. Then there exists an adversary $\mathcal{D}$ for the protocol Π, that corrupts at most $\epsilon' R(\Pi_{\mathcal{D}}) + 2\epsilon' \log_b \mathrm{CC}(\Pi_{\mathcal{D}})$ messages of total volume at most $2\epsilon \mathrm{CC}(\Pi_{\mathcal{D}})$,[14] such that the following holds:*

1. *$\Pi'_{\mathcal{A}}$ always sends at least $t_{\min}$ bits.*
2. *$\mathrm{CC}(\Pi'_{\mathcal{A}}) \leq (1 + 2600C\alpha)\,\mathrm{CC}(\Pi_{\mathcal{D}})$.*
3. *$R(\Pi'_{\mathcal{A}}) \leq R(\Pi_{\mathcal{D}}) + 2\log_b \mathrm{CC}(\Pi_{\mathcal{D}})$.*
4. *When $\Pi'_{\mathcal{A}}$ ends, both Alice and Bob (separately) can efficiently compute their view of the transcript of $\Pi_{\mathcal{D}}$.*
5. *The adversary $\mathcal{D}$ chooses the hash collisions in a probabilistic manner such that for every t and every r, with probability $\geq 1 - 20 \cdot 2^{-\frac{\gamma}{3d}t}$, the volume of hash collisions in the first t bits of $\Pi_{\mathcal{D}}$ is at most $35\gamma t$, and with probability $\geq 1 - 80r \cdot 2^{-\frac{7\gamma^8}{d}r}$, the number of rounds with hash collisions in the first r rounds of $\Pi_{\mathcal{D}}$ is at most $100\gamma r$.*
6. *Π' is efficiently computable if Π is efficiently computable.*

The proof of Theorem 17 is deferred to full version.

7 Putting it all Together

In this section, we prove our main theorem (Theorem 4), using the theorems from previous sections. We restate our main theorem for the sake of convenience.

▶ **Theorem 18.** *There exists a universal constant $\alpha_0 \geq 0$ such that for any blowup parameters $\alpha \leq \alpha_0$ and $\alpha' \leq 1$, there exist parameters $\epsilon = \tilde{\Omega}\left(\alpha^{3+\frac{1}{\alpha'}}\right)$, $\epsilon' = \tilde{\Omega}(\alpha\alpha'^3)$, and $\delta = \alpha^{O(1/\alpha')}$, and there exists a probabilistic oracle machine S, such that for any protocol $\Pi = (A, B)$,*

[13] The binary representation of $|T|'$ has length $\leq b^k W$ since $b^k W \geq \frac{1}{\alpha}$ and so $\log |T|' \leq \log \frac{b^k W}{\alpha} \leq b^k W$.
[14] $\Pi_{\mathcal{D}}$ denotes the protocol Π executed with the adversary $\mathcal{D}$.

in which the parties always transmit at least $t_{\min}$ bits (even in the presence of error), and for any adversary $\mathcal{A}$ that corrupts at most ϵ-fraction of the bits of the simulated protocol $\Pi' = (S^A, S^B)$, the protocol $\Pi'_{\mathcal{A}}$ (which is the protocol Π' executed with the adversary $\mathcal{A}$), satisfies the following properties.

1. $\mathrm{CC}(\Pi'_{\mathcal{A}}) \geq t_{\min}$.
2. *There exists* $t_0 = (1 + \tilde{O}(\alpha))\mathrm{CC}(A, B)$ *such that for all* $t > t_0$

$$\Pr[\mathrm{CC}(\Pi'_{\mathcal{A}}) > t] \leq 2 \cdot 2^{-\delta t} ,$$

   where the probability over the private randomness of S.
3. *There exists* $r_0 = (1 + O(\alpha'))\, R(A, B) + O\left(\frac{1}{\log \frac{2}{\alpha'}} \log \mathrm{CC}(A, B) + 1\right)$ *such that for any $r \geq r_0$, if at most ϵ'-fraction of the messages are α^2-corrupted, then*

$$\Pr\left[R(\Pi'_{\mathcal{A}}) > r\right] \leq 2 \cdot 2^{-\delta r} ,$$

   where the probability over the private randomness of S.
4. *For any $t > 0$,*

$$\Pr\left[(\mathsf{Output}(\Pi'_{\mathcal{A}}) \neq \mathsf{Trans}(\Pi)) \ \wedge \ (\mathrm{CC}(\Pi'_{\mathcal{A}}) > t)\right] \leq 2 \cdot 2^{-\delta t} ,$$

   where the probability over the private randomness of S.
5. *S is a probabilistic polynomial time oracle machine, and hence the computational efficiency of S^A and S^B is comparable to that of A and B, respectively.*

In the proof of this theorem, we use an error correcting code from a recent work of Guruswami and Li [18].

▶ **Theorem 19** ([18]). *For every $\alpha > 0$ there is an explicit encoding scheme* $\mathrm{Enc}, \mathrm{Dec} : \{0,1\}^* \to \{0,1\}^*$ *with the following properties:*

1. *For any $m \in \{0,1\}^*$ we have* $|\mathrm{Enc}(m)| = (1 + \tilde{O}(\alpha))|m|$.
2. *For any $m \in \{0,1\}^*$ and any y that can be obtained from* $\mathrm{Enc}(m)$ *by* $\alpha^2 \cdot |\mathrm{Enc}(m)|$ *insertions and deletions,* $\mathrm{Dec}(y) = m$.
3. Enc *and* Dec *are computable by a polynomial time Turing machine.*

In the proof of Theorem 18 we use the following padding claim.

▷ Claim 20. Let $\alpha, \beta \leq 0.1$, and $L_0 \geq \alpha^{-1}$. Then any $(\alpha, 2\beta)$-smooth protocol Π can be efficiently converted into an (α, β)-smooth protocol Π' such that

- Π can be computed from the first $(1 - 2\alpha)$ fraction of bits of Π'.
- $\mathrm{CC}(\Pi) + L_0 \leq \mathrm{CC}(\Pi') \leq (1 + 13\alpha)\mathrm{CC}(\Pi) + 3L_0$.
- $R(\Pi') \leq R(\Pi) + \log_{\frac{1}{\beta}} \mathrm{CC}(\Pi) + \log_{\frac{1}{\beta}} L_0 + 1$.

The proof of this claim is deferred to full version.

Proof of Theorem 18. Fix any $\alpha \leq \alpha_0$ and $\alpha' \leq 1$. Let C be the constant from Lemma 13 (see Section 5). Let Enc, Dec be the encoding scheme from Theorem 19 with the parameter α. Recall that for all m, $|\mathrm{Enc}(m)| = (1 + \tilde{O}(\alpha))|m|$. Let α_1 be the maximal constant that satisfies,

$$\forall m : |\mathrm{Enc}(m)| \leq 2|m|. \tag{6}$$

We define $\alpha_0 = \min\{\alpha_1, \frac{1}{3200C}\}$. Given α, α' define,

$$\beta = \frac{\alpha^{\frac{1}{\alpha'}}}{320 \log^2 \frac{1}{\alpha}} , \ \gamma = \frac{\alpha^{\frac{4}{\alpha'}}}{2^{40}} , \ b = \frac{2}{\alpha'} , \ d = \frac{\log \frac{1}{\gamma}}{\alpha} , \ L_0 = 250C\alpha^{-2} \log d ,$$

$$\epsilon = \min\left\{\frac{\alpha^3}{2bd\log d}, \frac{\alpha^3\beta}{320}\right\} \ , \ \epsilon' = \frac{\alpha{\alpha'}^3}{\log^3\frac{1}{\alpha}}, \text{ and } \delta = \gamma^9 \ .$$

These parameters were chosen to satisfy the following claim.

▷ Claim 21. Our parameters satisfy the following:[15]

1. $\epsilon = \tilde{\Omega}(\alpha^{3+\frac{1}{\alpha'}})$, $\epsilon' = \tilde{\Omega}(\alpha{\alpha'}^3)$ and $\delta = \alpha^{O(1/\alpha')}$.
2. $\alpha < \frac{1}{4}$ and $\beta \leq \frac{\alpha}{16}$.
3. $\alpha, \beta < 0.1$ and $L_0 \geq \alpha^{-1}$.
4. $\alpha \leq 0.01$, $\alpha' \leq 1$, $d \geq \frac{1}{\alpha}$ and $\beta \leq \min\left\{\alpha^{\frac{1}{\alpha'}}, \frac{1}{5\alpha d^2}\right\}$.
5. $\gamma \leq \min\left\{\frac{1}{d}, 2^{-b}\right\}$, $d \geq \frac{\log\frac{1}{\gamma}}{\alpha}$, and $L_0 \geq \max\{\alpha^{-2}, 250C\alpha^{-1}\log d\}$.
6. $\frac{2\epsilon}{\alpha^2} \leq \frac{\alpha}{bd\log d}$.
7. $18\beta^{-1}(35\gamma + \frac{4\epsilon}{\alpha^2}) + 20d\beta^{-1}(35\gamma + 4\epsilon) \leq \alpha$.
8. $(100\gamma + \epsilon') \cdot 906d\log\frac{1}{\beta} \leq \alpha'$ and $1812d\log\frac{1}{\beta}\epsilon' \leq \frac{1}{\log\frac{2}{\alpha'}}$.
9. $\delta \leq \frac{1}{(10\alpha^{-1}+10)L_0}$, and for all $x > 0$ we have that

$$2\cdot 2^{-\delta x} \geq \min\left\{1, \frac{120d}{\gamma}\cdot 2^{-\frac{\gamma}{3d}x} + \gamma^{-17}\cdot 2^{-\frac{3}{2}\gamma^8 x}\right\} \ .$$

The protocol Π' is defined as follows:

1. Convert Π into a $(\alpha, 2\beta)$-smooth protocol Π_{smooth} by applying Lemma 6 (Section 3) to Π, with respect to parameters $(\alpha, 2\beta)$. These parameters satisfy the requirements in Lemma 6 by Item 2.
2. Convert Π_{smooth} into Π_{pad} using Claim 20 (above) with parameters α, β, L_0. These parameters satisfy the requirements in Claim 20 by Item 3.
3. Convert Π_{pad} to the error-resilient protocol Π_{ideal}, which is error-resilient in the ideal hash model, by applying the protocol from Section 4 to Π_{pad}, with parameters $\alpha, \alpha', \beta, d$. Jumping ahead, note that by Item 4, these parameters satisfy Equation (3) which is required in order to apply Theorem 9.
4. Convert Π_{ideal} to the protocol Π_{rand}, which is obtained by instantiating the ideal hash using private randomness, obtained by applying the protocol described in Section 6 to Π_{ideal}. Jumping ahead, note that by Items 5 and 6, imply that Equations (4) and (5) are satisfied and the requirements of Theorem 17 are satisfied with respect to any adversary $\mathcal{A}$ that corrupts messages of total volume $\leq \frac{2\epsilon}{\alpha^2}\mathrm{CC}((\Pi_{\mathsf{rand}})_{\mathcal{A}})$.
5. Convert Π_{rand} to $\Pi' = (S^A, S^B)$, where Π' is the same as Π_{rand}, except that each message is sent encoded with the error correcting code from Theorem 19 with parameter α.

▶ **Lemma 22.** *The protocol $\Pi' = (S^A, S^B)$ satisfies the conditions of Theorem 18.*

The proof of Lemma 22 is deferred to full version. ◀

[15] Each of the following items will later be used to apply a different theorem from previous sections.

References

1 Shweta Agrawal, Ran Gelles, and Amit Sahai. Adaptive protocols for interactive communication. In *IEEE International Symposium on Information Theory, ISIT 2016, Barcelona, Spain, July 10-15, 2016*, pages 595–599, 2016. doi:10.1109/ISIT.2016.7541368.

2 Zvika Brakerski and Yael Tauman Kalai. Efficient Interactive Coding against Adversarial Noise. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012*, pages 160–166, 2012. doi:10.1109/FOCS.2012.22.

3 Zvika Brakerski, Yael Tauman Kalai, and Moni Naor. Fast Interactive Coding against Adversarial Noise. *J. ACM*, 61(6):35:1–35:30, 2014. doi:10.1145/2661628.

4 Zvika Brakerski and Moni Naor. Fast Algorithms for Interactive Coding. In *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '13, pages 443–456, 2013.

5 Mark Braverman. Towards deterministic tree code constructions. In *Innovations in Theoretical Computer Science 2012, Cambridge, MA, USA, January 8-10, 2012*, pages 161–167, 2012. doi:10.1145/2090236.2090250.

6 Mark Braverman and Klim Efremenko. List and Unique Coding for Interactive Communication in the Presence of Adversarial Noise. In *Proceedings of the IEEE Symposium on Foundations of Computer Science*, FOCS '14, pages 236–245, 2014.

7 Mark Braverman, Klim Efremenko, Ran Gelles, and Bernhard Haeupler. Constant-rate coding for multiparty interactive communication is impossible. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 999–1010, 2016. doi:10.1145/2897518.2897563.

8 Mark Braverman, Ran Gelles, Jieming Mao, and Rafail Ostrovsky. Coding for Interactive Communication Correcting Insertions and Deletions. In *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, pages 61:1–61:14, 2016. doi:10.4230/LIPIcs.ICALP.2016.61.

9 Mark Braverman and Anup Rao. Towards coding for maximum errors in interactive communication. In *Proceedings of the 43rd annual ACM symposium on Theory of computing*, STOC '11, pages 159–166, New York, NY, USA, 2011. ACM. doi:10.1145/1993636.1993659.

10 Klim Efremenko, Ran Gelles, and Bernhard Haeupler. Maximal Noise in Interactive Communication Over Erasure Channels and Channels With Feedback. *IEEE Trans. Information Theory*, 62(8):4575–4588, 2016. doi:10.1109/TIT.2016.2582176.

11 Ran Gelles. Coding for Interactive Communication: A Survey. *Foundations and Trends in Theoretical Computer Science*, 13(1-2):1–157, 2017. doi:10.1561/0400000079.

12 Ran Gelles and Bernhard. Capacity of Interactive Communication over Erasure Channels and Channels with Feedback. *SIAM J. Comput.*, 46(4):1449–1472, 2017. doi:10.1137/15M1052202.

13 Ran Gelles, Bernhard Haeupler, Gillat Kol, Noga Ron-Zewi, and Avi Wigderson. Towards Optimal Deterministic Coding for Interactive Communication. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1922–1936, 2016. doi:10.1137/1.9781611974331.ch135.

14 Ran Gelles and Yael Tauman Kalai. Constant-Rate Interactive Coding Is Impossible, Even In Constant-Degree Networks. *Electronic Colloquium on Computational Complexity (ECCC)*, 24:95, 2017. URL: https://eccc.weizmann.ac.il/report/2017/095.

15 Ran Gelles, Ankur Moitra, and Amit Sahai. Efficient and Explicit Coding for Interactive Communication. In *Proceeding of the IEEE Symposium on Foundations of Computer Science*, FOCS '11, pages 768–777, 2011.

16 Mohsen Ghaffari and Bernhard Haeupler. Optimal Error Rates for Interactive Coding II: Efficiency and List Decoding. *CoRR*, abs/1312.1763, 2013. arXiv:1312.1763.

17 Mohsen Ghaffari, Bernhard Haeupler, and Madhu Sudan. Optimal error rates for interactive coding I: adaptivity and other settings. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 794–803, 2014. doi:10.1145/2591796.2591872.

18 Venkatesan Guruswami and Ray Li. Efficiently decodable insertion/deletion codes for high-noise and high-rate regimes. In *IEEE International Symposium on Information Theory, ISIT 2016, Barcelona, Spain, July 10-15, 2016*, pages 620–624, 2016. doi:10.1109/ISIT.2016.7541373.

19 Bernhard Haeupler. Interactive Channel Capacity Revisited. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 226–235, 2014. doi:10.1109/FOCS.2014.32.

20 Bernhard Haeupler and Amirbehshad Shahrasbi. Synchronization strings: codes for insertions and deletions approaching the Singleton bound. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 33–46, 2017. doi:10.1145/3055399.3055498.

21 Bernhard Haeupler, Amirbehshad Shahrasbi, and Ellen Vitercik. Synchronization Strings: Channel Simulations and Interactive Coding for Insertions and Deletions. *CoRR*, abs/1707.04233, 2017. arXiv:1707.04233.

22 Bernhard Haeupler and Ameya Velingker. Bridging the Capacity Gap Between Interactive and One-Way Communication. *CoRR*, abs/1605.08792, 2016. arXiv:1605.08792.

23 Abhishek Jain, Yael Tauman Kalai, and Allison Lewko. Interactive Coding for Multiparty Protocols. In *Proceedings of the 6th Conference on Innovations in Theoretical Computer Science, ITCS '15*, pages 1–10, 2015.

24 Gillat Kol and Ran Raz. Interactive channel capacity. In *STOC '13: Proceedings of the 45th annual ACM symposium on Symposium on theory of computing*, pages 715–724, New York, NY, USA, 2013. ACM. doi:10.1145/2488608.2488699.

25 Joseph Naor and Moni Naor. Small-Bias Probability Spaces: Efficient Constructions and Applications. *SIAM J. Comput.*, 22(4):838–856, 1993. doi:10.1137/0222053.

26 Sridhar Rajagopalan and Leonard Schulman. A coding theorem for distributed computation. In *STOC '94: Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, pages 790–799, New York, NY, USA, 1994. ACM. doi:10.1145/195058.195462.

27 Leonard J. Schulman. Communication on noisy channels: a coding theorem for computation. *Foundations of Computer Science, Annual IEEE Symposium on*, pages 724–733, 1992. doi:10.1109/SFCS.1992.267778.

28 Leonard J. Schulman. Deterministic coding for interactive communication. In *STOC '93: Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, pages 747–756, New York, NY, USA, 1993. ACM. doi:10.1145/167088.167279.

29 Leonard J. Schulman. Coding for interactive communication. *IEEE Transactions on Information Theory*, 42(6):1745–1756, 1996.

30 Claude E. Shannon. A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(1):3–55, 2001. Originally appeared in *Bell System Tech. J.* 27:379–423, 623–656, 1948.

31 Alexander A. Sherstov and Pei Wu. Optimal Interactive Coding for Insertions, Deletions, and Substitutions. *Electronic Colloquium on Computational Complexity (ECCC)*, 24:79, 2017. URL: https://eccc.weizmann.ac.il/report/2017/079.

From Independent Sets and Vertex Colorings to Isotropic Spaces and Isotropic Decompositions: Another Bridge Between Graphs and Alternating Matrix Spaces

Xiaohui Bei
School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore
xhbei@ntu.edu.sg

Shiteng Chen
State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing, China
chenst@ios.ac.cn

Ji Guan
State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing, China
guanji1992@gmail.com

Youming Qiao
Centre for Quantum Software and Information, University of Technology Sydney, Australia
jimmyqiao86@gmail.com

Xiaoming Sun
Institute of Computing Technology, Chinese Academy of Sciences and University of Chinese Academy of Sciences, Beijing, China
sunxiaoming@ict.ac.cn

Abstract

In the 1970's, Lovász built a bridge between graphs and alternating matrix spaces, in the context of perfect matchings (FCT 1979). A similar connection between bipartite graphs and matrix spaces plays a key role in the recent resolutions of the non-commutative rank problem (Garg-Gurvits-Oliveira-Wigderson, FOCS 2016; Ivanyos-Qiao-Subrahmanyam, ITCS 2017). In this paper, we lay the foundation for another bridge between graphs and alternating matrix spaces, in the context of independent sets and vertex colorings. The corresponding structures in alternating matrix spaces are *isotropic spaces* and *isotropic decompositions*, both useful structures in group theory and manifold theory.

We first show that the maximum independent set problem and the vertex c-coloring problem reduce to the maximum isotropic space problem and the isotropic c-decomposition problem, respectively. Next, we show that several topics and results about independent sets and vertex colorings have natural correspondences for isotropic spaces and decompositions. These include algorithmic problems, such as the maximum independent set problem for bipartite graphs, and exact exponential-time algorithms for the chromatic number, as well as mathematical questions, such as the number of maximal independent sets, and the relation between the maximum degree and the chromatic number. These connections lead to new interactions between graph theory and algebra. Some results have concrete applications to group theory and manifold theory, and we initiate a variant of these structures in the context of quantum information theory. Finally, we propose several open questions for further exploration.

Dedicated to the memory of Ker-I Ko

2012 ACM Subject Classification Mathematics of computing → Graph coloring; Computing methodologies → Algebraic algorithms; Computing methodologies → Linear algebra algorithms

Keywords and phrases independent set, vertex coloring, graphs, matrix spaces, isotropic subspace

11th Innovations in Theoretical Computer Science Conference (ITCS 2020).
Editor: Thomas Vidick; Article No. 8; pp. 8:1–8:48

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.8

Funding *Shiteng Chen*: S. C. was partly supported by the National Key R&D Program of China (Grant No: 2018YFA0306701), and the National Natural Science Foundation of China (Grant No:61702489).
Ji Guan: J. G. was partly supported by the National Key R&D Program of China (Grant No: 2018YFA0306701), the National Natural Science Foundation of China (Grant No: 61832015), and the Key Research Program of Frontier Sciences, CAS, Grant No. QYZDJ-SSW-SYS003.
Youming Qiao: Y. Q. was partly supported by the Australian Research Council DECRA DE150100720.
Xiaoming Sun: X. S. was supported in part by the National Natural Science Foundation of China Grants No. 61832003, 61761136014, the Strategic Priority Research Program of Chinese Academy of Sciences Grant No. XDB28000000, and K. C. Wong Education Foundation.

Acknowledgements The authors would like to thank Nengkun Yu, James B. Wilson, and Gábor Ivanyos for discussions related to this paper. They would also like to thank George Glauberman and László Pyber for answering their questions on groups, including pointing out the reference [89] and clarifying the field characteristic issue in [22].

1 Introduction

1.1 Between graphs and matrix spaces: some known bridges

The bridge between perfect matchings and full-rank matrices

It is well-known that some graph-theoretic problems reduce to certain problems about linear spaces of matrices. A classical example, tracing back to Tutte [105], and then more systematically examined by Lovász [79, 81], concerns perfect matchings.

Let $\mathbb{F}$ be a field, and $[n] := \{1, \ldots, n\}$. Let $G = ([n], E)$ be a simple and undirected graph, so E can be viewed as a subset of $\{\{i, j\} : i, j \in [n], i \neq j\}$. For $n \in \mathbb{N}$ and $\{i, j\}$ where $i, j \in [n]$, $i < j$, the elementary alternating[1] matrix $A_{i,j}$ of size $n \times n$ is the matrix with the (i, j)th entry being 1, the (j, i)th entry being -1, and the rest entries being 0. Let $\mathcal{A}_G$ be the linear space of alternating matrices spanned by $A_{i,j}$, $\{i, j\} \in E$. Then when the field is large enough, G has a perfect matching if and only if $\mathcal{A}_G$ contains a full-rank matrix.

A similar construction for bipartite graphs is also classical. Let $G = (L \cup R, E)$ be a bipartite graph where $L = R = [n]$, so E can be viewed as a subset of $[n] \times [n]$. For $n \in \mathbb{N}$ and $i, j \in [n]$, the elementary matrix $E_{i,j}$ of size $n \times n$ is the matrix with the (i, j)th entry being 1, and the rest entries being 0. Let $\mathcal{B}_G$ be the linear space of matrices spanned by $E_{i,j}$, $(i, j) \in E$. Then when the field is large enough, G has a perfect matching if and only if $\mathcal{B}_G$ contains a full-rank matrix.

As noted by Lovász [79], these observations give efficient randomized algorithms for deciding the existence of perfect matchings on bipartite graphs and graphs over a large enough $\mathbb{F}$ via the celebrated Schwartz-Zippel lemma [97, 117]. Furthermore, because the determinant polynomial can be evaluated efficiently in parallel [14, 31], these are actually randomized NC algorithms.

This work of Lovász has inspired several prominent results, including randomized NC algorithms for constructing perfect matchings [66, 87], and the recent breakthrough of quasi-NC algorithms for perfect matchings on bipartite graphs [46] and on general graphs [100].

[1] An $n \times n$ matrix A over $\mathbb{F}$ is *alternating*, if for any $v \in \mathbb{F}^n$, $v^t A v = 0$. An alternating matrix is always skew-symmetric (i.e. $A^t = -A$), and a skew-symmetric matrix is also alternating over fields of characteristic not 2.

Furthermore, derandomizing the corresponding algorithm for *general* linear spaces of matrices – not necessarily those of the form $\mathcal{B}_G$ or $\mathcal{A}_G$ – is now known as the symbolic determinant identity testing problem, and turns out to be of fundamental significance in complexity theory, as that would imply strong circuit lower bounds which are considered to be beyond current techniques [28, 65].

In the following, we shall call linear spaces of (alternating) matrices as (alternating) matrix spaces. For a field $\mathbb{F}$, we use $\mathrm{M}(s \times t, \mathbb{F})$ to denote the linear space of all $s \times t$ matrices over $\mathbb{F}$, and write $\mathcal{B} \leq \mathrm{M}(s \times t, \mathbb{F})$ to denote that $\mathcal{B}$ is a matrix space in $\mathrm{M}(s \times t, \mathbb{F})$. Let $\mathrm{M}(n, \mathbb{F}) := \mathrm{M}(n \times n, \mathbb{F})$, and $\Lambda(n, \mathbb{F})$ be the linear space of all $n \times n$ alternating matrices over $\mathbb{F}$.

The bridge between shrunk subsets and shrunk subspaces

For bipartite graphs, a structure closely related to perfect matchings is the following. Given a bipartite graph $G = (L \cup R, E)$ where $L = R = [n]$, we say that a subset $S \subseteq L$ is a shrunk[2] subset of G, if $|S| > |N(S)|$ where $N(S)$ is the set of neighbours of S in R. The celebrated Hall's marriage theorem [55] says that G has a perfect matching if and only if it does not have a shrunk subset.

On the matrix space side, it is then natural to define the so-called shrunk subspaces. Specifically, given a matrix space $\mathcal{B} \leq \mathrm{M}(n, \mathbb{F})$, a subspace $U \leq \mathbb{F}^n$ is a shrunk subspace of $\mathcal{B}$, if $\dim(U) > \dim(\mathcal{B}(U))$ where $\mathcal{B}(U) := \langle \cup B(U) : B \in \mathcal{B} \rangle$, and $B(U)$ denotes the image of U under B.

As in the perfect matching case, a bipartite graph G has a shrunk subset if and only if $\mathcal{B}_G$ has a shrunk subspace [81]. However, for general matrix spaces, the natural analogue of Hall's theorem, namely a matrix space contains full-rank matrices if and only if it has no shrunk subspaces, does not hold, as evidenced by the space of all 3×3 alternating matrices. (The only if direction trivially holds, though.) Therefore, the bridge between shrunk subsets and shrunk subspaces is different from the one between perfect matchings and full-rank matrices.

The problem of testing whether a matrix space has a shrunk subspace arises naturally from several mathematical and computational displines, including algebraic complexity, non-commutative algebra, invariant theory, and analysis [48, 49, 61]. Not surprisingly then, this problem has had several names. We adopt the *non-commutative rank problem* which seems widely used now, and refer an interested reader to [48, 61] for the origin of this name.

With all these motivations, the non-commutative rank problem recently received considerable attention, and substantial progress has been made. First raised by Cohn [34] four decades ago in the study of free fields, it was more recently reached at by Mulmuley in the context of derandomizing the Noether's Normalization Lemma [85, 86], and also by Hrubeš and Wigderson in the context of non-commutative arithmetic circuits with divisions [56]. Only known to be in PSPACE before 2015 [35], this problem was shown to be in P over the rational number field [48] and over any field [61].

The techniques supporting the solutions to the non-commutative rank problem are reminiscent of the corresponding techniques for the perfect matching problem on bipartite graphs. In [48], it is the scaling algorithm [78, 99], generalized to the quantum operator setting [54]. In [61], it is the classical augmenting path algorithm, generalized to the matrix

[2] We call S to be "shrunk" instead of "shrinking", as we think of the bipartite graph G as shrinking the set S.

space setting [58, 60]. Ingredients from invariant theory are also crucial. For [48], Garg et al. needed the exponential upper bound on generating the ring of matrix semi-invariants [38]. For [61], Ivanyos et al. need the polynomial upper bound [39], which in turn relies crucially on the regularity lemma developed in [60].

1.2 Between graphs and matrix spaces: a new bridge

In this paper we lay the foundation for yet another bridge between graphs and matrix spaces. We focus on undirected simple graphs, hence it is natural, as Tutte and Lovász did with perfect matchings, to work with alternating matrix spaces. We start from independent sets and vertex colorings, two central structures in graph theory with numerous results from various motivations [40, 63]. By identifying analogues of them in the alternating matrix space setting, we arrive at isotropic spaces and isotropic decompositions, which we define now.

▶ **Definition 1.** *Let $\mathcal{A} \leq \Lambda(n, \mathbb{F})$ be an alternating matrix space. A subspace $U \leq \mathbb{F}^n$ is an* isotropic space *of $\mathcal{A}$, if for any $u, u' \in U$, and any $A \in \mathcal{A}$, we have $u^t A u' = 0$. For $c \in \mathbb{N}$, an* isotropic c-decomposition *of $\mathcal{A}$ is a direct sum decomposition of $\mathbb{F}^n$ into c nonzero subspaces $U_1 \oplus \cdots \oplus U_c$, where every U_i is an isotropic space.*

Recall that for a graph $G = ([n], E)$, an independent set of G is a subset $S \subseteq [n]$ such that for any $i, j \in S$, there is no edge from E connecting these two vertices. A vertex c-coloring of G is a partition of the vertex set into c independent sets. Therefore, the definitions of isotropic spaces and isotropic decompositions do mimic those of independent sets and vertex colorings. It is then natural to introduce the following definitions and the corresponding algorithmic problems.

▶ **Definition 2.** *Let $\mathcal{A} \leq \Lambda(n, \mathbb{F})$. The* isotropic number *of $\mathcal{A}$, denoted as $\alpha(\mathcal{A})$, is the maximum $d \in \mathbb{N}$ such that $\mathcal{A}$ has an isotropic space of dimension d. The* isotropic decomposition number*, denoted as $\chi(\mathcal{A})$, is the minimum $c \in \mathbb{N}$ such that $\mathcal{A}$ admits an isotropic c-decomposition.*

Given $d \in \mathbb{N}$ and a linear basis of $\mathcal{A} \leq \Lambda(n, \mathbb{F})$, the maximum isotropic space problem *asks to decide whether $\alpha(\mathcal{A}) \geq d$. Given $c \in \mathbb{N}$ and a linear basis of $\mathcal{A} \leq \Lambda(n, \mathbb{F})$, the* isotropic c-decomposition problem *asks to decide whether $\chi(\mathcal{A}) \leq c$.*

Note that $\alpha(\cdot)$ and $\chi(\cdot)$ are used to denote the independent number and the chromatic number of graphs [40], and these choices are deliberate. Also note that for any $\mathcal{A} \leq \Lambda(n, \mathbb{F})$, we have $\alpha(\mathcal{A}) \geq 1$, and $\chi(\mathcal{A}) \leq n$. Indeed, due to the alternating condition, any $\mathcal{A} \leq \Lambda(n, \mathbb{F})$ enjoys the property that any 1-dimensional subspace of $\mathbb{F}^n$ is an isotropic space of $\mathcal{A}$. It follows that any direct sum decomposition of $\mathbb{F}^n$ into n dimension-1 subspaces is an isotropic n-decomposition of $\mathcal{A}$. This property corresponds nicely to that for any undirected simple graph, every single vertex is an independent set. On the other hand, symmetric matrix spaces do not satisfy this property in general. Therefore, this small but pleasant coincidence suggests that working with alternating matrix spaces is a natural choice in this setting.

Our first result follows what Lovász did with perfect matchings, and provides a first indication on the new connection. Recall that given a graph $G = ([n], E)$, we can associate an alternating matrix spaces $\mathcal{A}_G \leq \Lambda(n, \mathbb{F})$, spanned by those elementary alternating matrices $A_{i,j}$ with $\{i, j\} \in E$.

▶ **Theorem 3.** *Let G and $\mathcal{A}_G$ be as above. Then we have*

1. *G has a size-s independent set if and only if $\mathcal{A}_G$ has a dimension-s isotropic space. In particular, $\alpha(G) = \alpha(\mathcal{A}_G)$.*
2. *G has a vertex c-coloring if and only if $\mathcal{A}_G$ has an isotropic c-decomposition. In particular, $\chi(G) = \chi(\mathcal{A}_G)$.*

The proof is in Section 4. Theorem 3 demonstrates that the maximum isotropic space problem and the isotropic decomposition problem are genuine generalizations[3] of the maximum independent set problem and the vertex c-coloring problem, respectively. It also implies the following.

▶ **Corollary 4.** *The maximum isotropic space problem and the isotropic* 3*-decomposition problem for alternating matrix spaces are NP-hard.*

Emboldened by Theorem 3, we propose to view isotropic spaces and decompositions as linear algebraic analogues of independent sets and vertex colorings, and study these two structures from the perspectives of graph theory and algorithms. This leads to natural and interesting mathematical and algorithmic problems, whose solutions bring together strategies, techniques, and results from several areas, including graph theory, algorithm design, computer algebra, and algebraic complexity. We regard these results as laying the foundation of a new bridge between graphs and alternating matrix spaces.

While our investigation started with an analogy, isotropic spaces and decompositions are actually classical notions, with natural interpretations in group theory and manifold theory. Therefore, some of our results have concrete applications to these areas. We also initiate a variant of this theory in quantum information, and find an interesting information theoretic interpretation of isotropic spaces in quantum error correction. These demonstrate the usefulness of this new bridge.

Before we go on to detailed descriptions of our results, we set up some notation. We use $\mathbb{F}_q$, $\mathbb{Q}$, $\mathbb{R}$, and $\mathbb{C}$ to denote the finite field with q elements, the rational number field, the real number field, and the complex number field, respectively. Elements in $\mathbb{F}^n$ are *column* vectors. In algorithms, subspaces of $\mathbb{F}^n$ and $\Lambda(n, \mathbb{F})$ are represented by linear bases. We may write $\Lambda(n, \mathbb{F}_q)$ as $\Lambda(n, q)$ for convenience. For more details on the computation model, see Section 2.1.

We also recall some well-known graph-theoretic and/or algorithmic results [40], which will be useful in seeing the analogues.

1. Whether a graph is bipartite can be tested in deterministic polynomial time.
2. On bipartite graphs, the maximum independent set problem is in P.
3. Any n-vertex and m-edge graph has an independent set of size $\geq \frac{n^2}{2m+n}$ [104][4].
4. The number of maximal independent sets on an n-vertex graph is $\leq 3^{\frac{n}{3}}$ [84].
5. The chromatic number of an n-vertex graph can be computed in time $(1+3^{\frac{1}{3}})^n \cdot \mathrm{poly}(n)$ [72][5].

All the above results will be found to have natural correspondences in the alternating matrix space setting. The reader may find some fun in trying to formulate the correspondences by him/herself.

1.3 Linear algebraic analogues of bipartite testing and maximum independent set on bipartite graphs

After Corollary 4, the isotropic 2-decomposition problem is of particular interest, as the vertex 2-coloring problem just asks whether a graph is bipartite, which can be tested efficiently by breadth-first search. A moment's thought sugggests that a breadth-first search type idea seems not applicable to the isotropic 2-decomposition problem (see also Appendix A).

[3] Note that for the maximum isotropic space problem and the isotropic decomposition problem, we consider all alternating matrix spaces, not necessarily of the form $\mathcal{A}_G$ coming from a graph G.

[4] This follows from Turán's celebrated result in extremal graph theory, which is usually stated for cliques, and implies this by simply taking the complement graph.

[5] This classical result of Lawler was from the 1970's, and the current status of the art is $2^n \cdot \mathrm{poly}(n)$ [17].

Fortunately, it turns out that this problem has been studied in computer algebra over finite fields by Brooksbank, Maglione, and Wilson in [20]. Their strategy can be readily applied to $\mathbb{R}$ and $\mathbb{C}$, using some ingredients from [59].

▶ **Theorem 5** ([20, Theorem 3.6], [59])**.** *The isotropic 2-decomposition problem can be solved in randomized polynomial time over $\mathbb{F}_q$ with q odd, and in deterministic polynomial time over $\mathbb{R}$ and $\mathbb{C}$. Furthermore, over $\mathbb{F}_q$ with q odd, the algorithm also outputs the linear bases of the two subspaces in an isotropic 2-decomposition.*

While a proof for $\mathbb{F}_q$ was already sketched in [20], we still give an exposition of this proof in Section 5. Besides indicating how to handle $\mathbb{R}$ and $\mathbb{C}$, we wish to give some reader a flavor of how the so-called $*$-algebra technique, pioneered by Wilson [111, 112] in computer algebra, is applied to this setting. This technique was recently shown to be useful in polynomial identity testing and multivariate cryptography [59].

Theorem 5 and its proof reveal that the isotropic spaces and the isotropic decompositions do have connections with, and implications to, other disciplines, just like the case of non-commutative ranks. Furthermore, a quantum variant of the theory can be developed, and the corresponding isotropic 2-decomposition problem can be solved efficiently using quantum information theoretic techniques (see Section 12). As mentioned above, the techniques used to solve the non-commutative rank problem also have their roots in algebra [61] and quantum information [48]. Perhaps it is not so coincidental that techniques from these areas are useful again.

In fact, the non-commutative rank problem arises naturally in our context. Since a bipartite graph is just a graph admitting a vertex 2-coloring, it is natural to make the following definition.

▶ **Definition 6.** *An alternating matrix space is* bipartite*, if it admits an isotropic 2-decomposition.*

A well-known fact in graph theory is that, on bipartite graphs, the maximum independent set problem can be solved in deterministic polynomial time, through a reduction to the minimum vertex cover problem. The latter problem is equivalent to the maximum matching problem via Kőnig's theorem. It is then interesting to examine whether bipartite alternating matrix spaces admit an efficient algorithm for the maximum isotropic space problem. It turns out that the isotropic number of a bipartite alternating matrix space is closely related to the non-commutative rank of some matrix space, as we shall see now.

We have mentioned the decision version of the non-commutative rank problem in Section 1.1. We now define the non-commutative rank in a slightly more general setting. Given $\mathcal{B} \leq \mathrm{M}(s \times t, \mathbb{F})$, its non-commutative rank is $\mathrm{ncrk}(\mathcal{B}) := s + t - \max\{\dim(U) + \dim(V) : \forall u \in U, v \in V, B \in \mathcal{B}, u^t B v = 0\}$ [47]. Note that the recent works [48, 61] used a slightly different formulation in the setting $s = t$.

Given a bipartite $\mathcal{A} \leq \Lambda(n, \mathbb{F})$, up to isometry (i.e. the action of $T \in \mathrm{GL}(n, \mathbb{F})$ sending $\mathcal{A}$ to $T^t \mathcal{A} T := \{T^t A T : A \in \mathcal{A}\}$), every $A \in \mathcal{A}$ is of the form $\begin{bmatrix} \mathbf{0} & B \\ -B^t & \mathbf{0} \end{bmatrix}$, where B is of size $s \times t$ (see Section 3). Let $\mathcal{B} \leq \mathrm{M}(s \times t, \mathbb{F})$ be the space of such B arising from some $A \in \mathcal{A}$. Then we have:

▶ **Theorem 7.** *Let $\mathcal{A} \leq \Lambda(n, \mathbb{F})$ and $\mathcal{B} \leq \mathrm{M}(s \times t, \mathbb{F})$ be from above. Then $\alpha(\mathcal{A}) = n - \mathrm{ncrk}(\mathcal{B})$.*

Thanks to the solution of the non-commutative rank problem over any field [61], and Theorem 5 in the case of $\mathbb{F}_q$ with odd q, we have

▶ **Corollary 8.** *The maximum isotropic space problem for bipartite alternating matrix spaces of size $n \times n$ over $\mathbb{F}_q$, q odd, can be solved in randomized* $\mathrm{poly}(n, \log q)$ *time.*

The proofs of Theorem 7 and Corollary 8 are in Section 6. In some sense, the non-commutative rank may be considered as corresponding to the minimum vertex cover size in the bipartite alternating matrix space setting. However, unlike in the graph case, where the relation between independent sets and vertex covers is so straightforward, the proof of Theorem 7 requires some twists, because of the "flexibility" of vectors and matrices.

Having seen the implication of the non-commutative rank problem to our setting, let us examine the following mathematical problem that arises naturally in our context, whose solution turns out to come from algebraic geometry. Again, let us trace back to the graph setting, and consider $\alpha(n, m) := \min\{\alpha(G) : G \text{ a graph with } n \text{ vertices and } m \text{ edges}\}$, where $\alpha(G)$ is the independence number. A celebrated result of Turán [104] in extremal graph theory implies that

$$\alpha(n, m) \leq \left\lceil \frac{n^2}{2m + n} \right\rceil. \tag{1}$$

Turning to the alternating matrix space setting, it is natural to define $\alpha(\mathbb{F}, n, m) := \min\{\alpha(\mathcal{A}) : \mathcal{A} \leq \Lambda(n, \mathbb{F}), \dim(\mathcal{A}) = m\}$. This quantity has been studied by Buhler, Gupta, and Harris [22] in relation to abelian subgroups of p-groups [4, 23]. The main result of [22], proved using algebraic geometric techniques, is as follows: for any $m > 1$, we have

$$\alpha(\mathbb{F}, n, m) \leq \left\lfloor \frac{m + 2n}{m + 2} \right\rfloor, \tag{2}$$

where the equality is attainable over algebraically closed fields[6]. This inequality was also obtained earlier by Ol'shanskii [89]. Comparing Equations 1 and 2, we see that $\alpha(n, m)$ and $\alpha(\mathbb{F}, n, m)$ behave quite differently. For example, by Equation 1, every graph with n vertices and $2n$ edges has an independent set of size at least $n/5$. On the other hand, by Equation 2, there exists a dimension-$2n$ alternating matrix space in $\Lambda(n, \mathbb{F})$ with no isotropic space of dimension ≥ 2.

Motivated by the discussion in the last paragraph, we study the algorithmic problem of deciding whether there exists an isotropic space of dimension ≥ 2 for $\mathcal{A} \leq \Lambda(n, \mathbb{F})$. This is equivalent to ask whether $\mathcal{A}$ has an isotropic $(n - 1)$-decomposition. Note that the corresponding problem on graphs is trivial, as a graph has an independent set of size ≥ 2 if and only if it is not the complete graph. It turns out that over $\mathbb{Q}$, this problem is substantially more difficult.

▶ **Theorem 9.** *Over $\mathbb{Q}$, assuming the generalized Riemann hypothesis, there is a randomized polynomial-time reduction from deciding quadratic residuosity modulo squarefree composite numbers to the problem of deciding whether an alternating matrix space has an isotropic space of dimension ≥ 2.*

The proof is in Section 7. It relies crucially on Rónyai's fundamental work on computing algebra structures [92]. One ingredient here is the introduction of the existential singularity problem for matrix spaces, which turns out to have rich connections to several mathematical disciplines (see Problem 25 and Section 7.1).

[6] While in [22] the main result was stated for fields of characteristic $\neq 2$, the proof, at least the inequality, works for any characteristic.

1.4 Linear algebraic analogues of bounding the number of maximal independent sets and exact exponential algorithms for chromatic numbers

An independent set on a graph is maximal if it is not properly contained in some other independent set. The study of maximal independent sets is a classical demonstration of how graph theory and algorithm study are intertwined.

In the 1960's, Erdős and Moser raised the question of bounding the number of maximal independent sets on a graph. It was subsequently solved by Moon and Moser [84], and alternative proofs have been found [108, 115]. They show that the number of maximal independent set of an n-vertex graph is upper bounded by $3^{n/3}$, and this bound is tight. (Some refinement is required when n is not a multiple of 3.) Since the 1970's, the problem of outputting all maximal independent sets or maximal cliques received considerable attention [7, 64, 73, 103]. One application was provided by Lawler [72], who showed that the Moon-Moser bound together with dynamic programming give an algorithm for computing the chromatic number of an n-vertex graph in time $(1 + \sqrt[3]{3})^n \cdot \mathrm{poly}(n)$. This algorithm was the starting point of exact exponential-time algorithms for chromatic numbers. Subsequent improvements [15, 25, 44] lead to the breakthrough by Björklund, Husfeldt, and Koivisto, who presented an algorithm in time $2^n \cdot \mathrm{poly}(n)$ [17].

Getting back to alternating matrix spaces, the natural correspondence would be maximal isotropic spaces. Formally, for an alternating matrix space $\mathcal{A} \leq \Lambda(n, \mathbb{F})$, an isotropic space is *maximal*, if there is no isotropic space properly containing it. We then ask analogous questions over finite fields, namely upper bounding the number of maximal isotropic spaces of $\mathcal{A} \leq \Lambda(n, \mathbb{F}_q)$, and exact exponential-time algorithms for computing the isotropic decomposition number $\chi(\mathcal{A})$. Interestingly, on one hand, these problems demonstrate behaviours different from the combinatorial counterpart. On the other hand, strategies and techniques from graph theory and algorithm design do carry over, in a non-trivial way, to these problems. Again, such phenomena have been witnessed in the non-commutative rank problem, and it is interesting to see these happening in this context. Furthermore, our result on the number of maximal isotropic spaces has a direct application to group theory, as we will see in Section 1.6.

We now describe our results in more details. To start with, we note that, as in the graph setting, an easy greedy algorithm outputs one maximal isotropic space (see Proposition 18). We then consider the number of maximal isotropic spaces for alternating matrix spaces in $\Lambda(n, q)$, analogously as done by Moon and Moser for graphs [84]. A trivial upper bound is the number of all subspaces of $\mathbb{F}_q^n$. This number, $q^{\frac{1}{4}n^2+\Theta(n)}$, is well-known and classical (see Fact 29). Any alternating matrix space spanned by a single full-rank alternating matrix has $q^{\frac{1}{8}n^2+\Theta(n)}$ maximal isotropic spaces, providing a lower bound. This is also classical but perhaps not that well-known (see Proposition 30). We show a non-trivial upper bound as follows.

▶ **Theorem 10.** *The number of maximal isotropic spaces of any $\mathcal{A} \leq \Lambda(n, q)$ is upper bounded by $q^{\frac{1}{6}n^2+O(n)}$.*

The proof is in Section 8. We adapt the proof strategy of the upper bound on maximal independent sets by Wood [115]. This requires analogues of certain graph-theoretic concepts such as degrees and neighbours in the alternating matrix space setting, which have been developed in [90]. It works up to some point, but after that, we have to resort to certain linear algebraic techniques. We leave closing the gap between $q^{\frac{1}{8}n^2+\Omega(n)}$ and $q^{\frac{1}{6}n^2+O(n)}$ an interesting open problem.

The proof of Theorem 10 is constructive (see Corollary 35), so we can enumerate all maximal isotropic spaces in time $q^{\frac{1}{6}n^2+O(n)}$. We then consider the problem of computing the isotropic decomposition number for $\mathcal{A} \leq \Lambda(n,q)$. A naive brute-force algorithm, namely enumerating all direct sum decompositions of $\mathbb{F}_q^n$, runs in time $q^{n^2+O(n)}$. Inspired by Lawler's strategy in [72], we combine our Corollary 35 with a dynamic programming idea to obtain the following.

▶ **Theorem 11.** *The isotropic decomposition number of $\mathcal{A} \leq \Lambda(n,q)$ can be computed in time $q^{\frac{5}{12}n^2+O(n)}$.*

The proof is in Section 9. An open question is whether the strategy in [17] for chromatic numbers can be adapted to obtain an algorithm for isotropic decomposition numbers in time $q^{\frac{1}{4}n^2+O(n)}$. This is because the number of subspaces of $\mathbb{F}_q^n$ is $q^{\frac{1}{4}n^2+\Theta(n)}$, while the algorithm in [17] runs in time $2^n \cdot \mathrm{poly}(n)$ where 2^n is the number of subsets of $[n]$.

1.5 Complexity-theoretic upper bound over $\mathbb{C}$, and the dependence of the independence number on the maximum degree

We first consider complexity-theoretic upper bounds for the maximum isotropic space problem and the isotropic 3-decomposition problem. Clearly, these problems are in NP over finite fields. Over $\mathbb{C}$, we have the following, by resorting to a celebrated result of Koiran on the Hilbert Nullstellensatz problem [69]. The proof is in Section 10.

▶ **Proposition 12.** *Let $\mathcal{A} \leq \Lambda(n,\mathbb{C})$ be given by a linear basis consisting of integral matrices. The maximum isotropic space problem and the isotropic 3-decomposition problem are in PSPACE unconditionally, and in PH assuming the generalized Riemann hypothesis.*

Our next result has two diverse motivations.

The first motivation is from linear algebra. Given a single alternating matrix A, its canonical form suggests that $\langle A \rangle$ admits an isotropic 2-decomposition (see Section 2). Given a pair of alternating matrices A_1 and A_2, it is also known that $\langle A_1, A_2 \rangle$ admits an isotropic 2-decomposition [51, 52, 96] (see also [20, Lemma 3.7]). A natural question is what happens for alternating matrix spaces of dimension 3, or in general, any constant c.

The second motivation is from graph theory. Given a graph $G = ([n], E)$, let $\Delta(G)$ be the maximum degree over vertices of G. It is well-known that a simple greedy algorithm yields that $\chi(G) \leq \Delta(G) + 1$ [40, pp. 122]. For $\mathcal{A} \leq \Lambda(n,\mathbb{F})$, the *degree* of $v \in \mathbb{F}^n$ in $\mathcal{A}$ can be defined as $\deg_{\mathcal{A}}(v) := \dim(\langle Av : A \in \mathcal{A} \rangle)$ [90]. As mentioned in Section 1.4, this notion was already useful in the proof of Theorem 10. Let $\Delta(\mathcal{A}) = \max\{\deg_{\mathcal{A}}(v) : v \in \mathbb{F}^n\}$. It is then natural to ask the relation between $\chi(\mathcal{A})$ and $\Delta(\mathcal{A})$ in analogy to the graph setting. This question is closely related to the one in the last paragraph, since $\deg_{\mathcal{A}}(v) \leq \dim(\mathcal{A})$ for any $v \in \mathbb{F}^n$, so $\Delta(\mathcal{A}) \leq \dim(\mathcal{A})$.

We now present the following result, also deduced from a greedy algorithm.

▶ **Proposition 13.** *Let $\mathcal{A} \leq \Lambda(n,\mathbb{F})$. Then $\chi(\mathcal{A}) \leq O(\Delta(\mathcal{A}) \cdot \log n)$. Furthermore, an isotropic C-decomposition with $C = O(\Delta(\mathcal{A}) \cdot \log n)$ can be found in polynomial time.*

The proof is in Section 10. Note that this implies that when $\dim(\mathcal{A})$ is a constant, then $\chi(\mathcal{A}) \leq O(\log n)$. We leave it an open problem for further improvement of the bound in Proposition 13.

1.6 Applications of our results

After studying problems on alternating matrix spaces mostly by way of analogy with graphs, it is natural to ask whether some results have concrete applications. The answer is quite affirmative.

In this subsection, we first provide two applications to finite groups, one being computational, and the other being enumerative. As will be explained below, these applications are based on a family of finite groups, for which testing isomorphism has long been known to be difficult, and is becoming more urgent in light of Babai's recent breakthrough on graph isomorphism [9].

We then describe a variant of our theory in the context of quantum information theory, and present an information theoretic interpretation of isotropic spaces in the context of quantum error correction. In Section 3.1, we also present the connections to manifold theory, and mention a potential application. All these suggest that our results could be of interest to group theorists, quantum information theorists, and geometers, in particular to those who work on the computational aspects of these disciplines.

Of course, these applications and connections are not surprising to readers in these fields, because alternating bilinear maps, and therefore alternating matrix tuples and spaces[7], naturally arise in group theory via the commutator bracket, and in manifold geometry via the cup product in cohomology. Therefore, certain isotropic spaces and decompositions have natural group-theoretic or geometric interpretations (see Section 3.1).

On the other hand, these applications may look exotic to some other readers, as they will be stated purely in group theoretic or quantum information theoretic terms. This is natural and expected, after a bridge is built. Indeed, the present bridge enables us to transfer problems, techniques, and results in graph theory and algorithm study, to other mathematical and computational disciplines which otherwise seem barely related to graph theory.

We now describe the first application to finite groups, more specifically, to computing with matrix groups over finite fields. Matrix groups over finite fields given by generators form an important model of computing with finite groups. In theoretical computer science, the study of this model led to the inventions of the black-box group model by Babai and Szemerédi [11], and the Arthur-Merlin class by Babai [8]. Though some algorithms with worst-case analyses can be found in [13], even the very basic membership testing problem was only recently known to be solvable in randomized polynomial time under a number-theoretic oracle [10].

Overall, our knowledge about this model is rather limited, and many questions await investigations. One interesting problem is to compute an abelian subgroup of the largest size. Large abelian subgroups, besides motivations from computational group theory [95], are useful in controlling the character degrees of the group, which in turn are useful in the group-theoretic approach for fast matrix multiplication [33]. As a consequence of Corollary 4, we have the following result, whose proof is in Section 11.

▶ **Theorem 14.** *Let p be an odd prime. Given a matrix group G over $\mathbb{F}_p$, and $s \in \mathbb{N}$, deciding whether G has an abelian subgroup of order $\geq s$ is NP-hard.*

The proof of Theorem 14 relies on the connection between alternating matrix spaces over $\mathbb{F}_p$, and p-groups of class 2 and exponent p for odd p, via Baer's correspondence [12] (see Section 3.1).

[7] For the relations among alternating matrix tuples and spaces, and alternating bilinear maps, see Sec. 2 and B.

It has long been known that p-groups of class 2 and exponent p form a bottleneck for testing isomorphism of finite groups. To solve the group isomorphism problem in time polynomial in the group order is a long-standing open problem [83]. This problem is becoming more prominent in light of Babai's breakthrough on graph isomorphism [9], as Babai indicated the group isomorphism problem as a key bottleneck to put graph isomorphism in P [9, arXiv version, Section 13.2]. Some interesting progress on testing isomorphism of such p-groups was recently made by utilizing the connection to alternating matrix spaces [76].

Let us turn to the second application to finite groups. The question of bounding the number of maximal abelian subgroups has been considered for various group families [6, 42, 109, 116], but to the best of our knowledge, there had been no results on this question for p-groups of class 2 and exponent p. Let P be such a group, so that the center $Z(P) \cong \mathbb{Z}_p^m$ and the central quotient $P/Z(P) \cong \mathbb{Z}_p^n$. The number of maximal abelian subgroups is upper bounded trivially by $p^{\frac{1}{4}n^2+O(n)}$, the number of subgroups of $\mathbb{Z}_p^n$. Our Theorem 10 then provides the following improvement, whose proof is in Section 11.

▶ **Theorem 15.** *Let P be as above. Then the number of maximal abelian subgroups of P is upper bounded by $p^{\frac{1}{6}n^2+O(n)}$.*

Recall that the proof of Theorem 10 starts by following the strategy of Wood's proof [115] of bounding the number of maximal independent sets on a graph. We view this as an interesting and somewhat unexpected example of transferring techniques from graph theory to group theory.

We also initiate a quantum variant of the theory in Section 12. There, the objects are a special type of quantum channels, and isotropic spaces and isotropic decompositions are defined on the Kraus operators of such channels. Furthermore, we require the isotropic decompositions to be orthogonal. One can then transform classical connected graphs into such channels, and prove an analogue of Theorem 3. More surprisingly, we also obtain an efficient isotropic 2-decomposition algorithm, as an analogue of Theorem 5, by resorting to the recent development on the periodicity of quantum Markov chains [53].

We then present an information theoretic interpretation for isotropic spaces in the context of quantum error correction. Briefly speaking, from the viewpoint of certain natural generalizations of quantum gate fidelities [88], isotropic spaces can be viewed as the opposite structure of noiseless subspaces (Proposition 47), which have been studied intensively in quantum error correction [68, 77]. Indeed, noiseless subspaces are shelters for the information residing in them under quantum noise, while the information in an isotropic space would be completely destroyed by quantum noise.

Let us conclude this subsection with a remark on these applications. After building a bridge, we expect it to serve as a two-way street between the two sides. However, in reality there is usually more traffic in one direction than the other. For example, the traffic between perfect matchings and full-rank matrices mostly goes from the algebra side to the combinatorial side, e.g., the randomized NC algorithm for perfect matchings [79]. The traffic between shrunk subsets and shrunk subspaces mostly goes in the other direction, e.g., linear algebraic analogues of augmenting paths [58] and scaling [54]. So far, our applications in this work mostly go in the direction from combinatorics to algebra, following the pattern of the shrunk subset vs. shrunk subspace case. It will be very interesting to explore implications in the other direction in the future.

1.7 Outlook

Summary of our contributions

The concepts of isotropic spaces and isotropic decompositions for alternating bilinear maps are classical, with natural interpretations in group theory and manifold theory. Our key new insight is that they can be viewed and studied as linear algebraic analogues of independent sets and vertex colorings. This insight leads us to study algorithmic and mathematical problems about isotropic spaces and isotropic decompositions, by drawing inspirations from results and techniques from graph theory and algorithm study. The techniques used to address the problems range from combinatorics, to algebra, and to quantum information.

We believe that this investigation is fruitful, for the following reasons.

1. First, it discloses new algorithmic and mathematical questions. For example, in Section 1.4 we proposed and studied upper bounding the number of maximal isotropic spaces, and exact exponential-time algorithms for isotropic decomposition numbers over finite fields.
2. Second, the results obtained have concrete applications to other mathematical and computational disciplines. For example, in Section 1.6, we described the applications of our results to finite groups and quantum information.
3. Third, it sheds new lights on known results from different research directions. For example, in Section 1.3 we compared Turán's extremal graph result with Buhler, Gupta, and Harris' algebraic geometric result.

This investigation then lays the foundation for yet another bridge between graphs and alternating matrix spaces, adding to the classical ones established by Tutte and Lovász.

Open ends

Several interesting open problems have been mentioned before, and here we give a summary and propose some new ones.

1. By Theorem 10, the number of maximal isotropic spaces of $\mathcal{A} \leq \Lambda(n, q)$ is upper bounded by $f(n, q) = q^{\frac{1}{6}n^2+O(n)}$. There exists an alternating matrix space with $g(n, q) = q^{\frac{1}{8}n^2+\Omega(n)}$ many maximal isotropic spaces (see Section 1.4). Either improve the current upper bound $f(n, q)$, or construct an alternating matrix space with more than $g(n, q)$ maximal isotropic spaces. Note that resolving this problem would lead to a sharp bound on the number of maximal abelian subgroups of p-groups of class 2 and exponent p.
2. Improve the exact exponential-time algorithm for computing the isotropic decomposition number for $\mathcal{A} \leq \Lambda(n, q)$ in Theorem 11. An interesting question is whether the strategy in [17] can be adapted here. The results in [16] should be useful in this context.
3. Despite Theorem 9, the complexities of deciding whether an alternating matrix space has an isotropic space of dimension ≥ 2 are not clear over various fields. Even over $\mathbb{Q}$, our proof for Theorem 9 relies on a special case of the underlying existential singularity problem for matrix spaces (see Section 7.1), so it is left open even for the general case of that problem over $\mathbb{Q}$.
4. Investigate the behaviours of the isotropic and isotropic decomposition numbers in the linear algebraic Erdős-Rényi model [18, 76].
5. Improve the dependence of the isotropic decomposition number on the maximum degree, or the dimension of the alternating matrix space (see Proposition 13). Note that this problem has motivations from classical geometry (see the discussions before Proposition 13).

The structure of the paper

We present certain preliminaries in Section 2. Then in Section 3, we collect some basic facts and properties about isotropic spaces and decompositions, including their meanings in group theory and manifold geometry in Section 3.1. We then prove Theorem 3 in Section 4, which is the basis connecting the graph-theoretic structures and those structures on alternating matrix spaces. We then prove all the main results mentioned above in the following sections. (We have mentioned the corresponding section numbers when describing those results.) An appendix then follows, containing some background material and discussions on certain conceptual questions.

2 Preliminaries

Notation

For $n \in \mathbb{N}$, $[n] := \{1, \ldots, n\}$. We use $\uplus$ for disjoint union of sets. The base of the logarithm is 2 unless otherwise stated.

Let $\mathbb{F}$ be a field. We use $\mathbb{F}^n$ to denote the vector space of *column* vectors of length n over $\mathbb{F}$. The standard basis of $\mathbb{F}^n$ consists of vectors $e_1, \ldots, e_n$, where e_i is the vector with the ith entry being 1, and other entries being 0. The linear span of several vectors or matrices is denoted by $\langle \cdot \rangle$.

For $n, d \in \mathbb{N}$, let $\mathrm{M}(n \times d, \mathbb{F})$ be the linear space of $n \times d$ matrices over $\mathbb{F}$, and $\mathrm{GL}(n \times d, \mathbb{F})$ the set of $n \times d$ matrices over $\mathbb{F}$ of rank $\min(n, d)$. We also let $\mathrm{M}(n, \mathbb{F}) := \mathrm{M}(n \times n, \mathbb{F})$, and $\mathrm{GL}(n, \mathbb{F}) := \mathrm{GL}(n \times n, \mathbb{F})$. Dimension-$d$ subspaces of $\mathbb{F}^n$ will be understood as represented by elements from $\mathrm{GL}(n \times d, \mathbb{F})$. Given $A \in \mathrm{M}(n \times d, \mathbb{F})$, the transpose of A is denoted by $A^t \in \mathrm{M}(d \times n, \mathbb{F})$. For convenience, we sometimes write a vector v in $\mathbb{F}^n$ as $v = (v_1, \ldots, v_n)^t$. Let $\mathbb{K}/\mathbb{F}$ be a quadratic extension, and let $\overline{\alpha}$ denote the image of $\alpha \in \mathbb{K}$ under the quadratic involution. For a matrix $A \in \mathrm{M}(n \times d, \mathbb{K})$, $A^\dagger$ denotes the conjugate transpose of A.

Depending on the context, $\mathbf{0}$ may denote either the zero space, a zero vector, or a zero matrix. The identity matrix in $\mathrm{M}(n, \mathbb{F})$ is denoted by I_n; we may drop the subscript n when it is understood from the context. Given a matrix $A \in \mathrm{M}(n, \mathbb{F})$, its kernel and image are denoted by $\ker(A)$ and $\mathrm{im}(A)$, respectively. For $U \leq \mathbb{F}^n$, the image of U under A is denoted by $A(U)$.

Linear algebra

Given $U \leq \mathbb{F}^n$, a complementary subspace, or just a complement, is some $V \leq \mathbb{F}^n$ such that $V \cap U = \mathbf{0}$, and $\langle U \cup V \rangle = \mathbb{F}^n$. Note that complement subspaces of U are not unique. Indeed, the number of complements of a dimension-d subspace $U \leq \mathbb{F}_q^n$ is $q^{d(n-d)}$. The space orthogonal to U is $\{v \in \mathbb{F}^n : \forall u \in U, v^t u = 0\}$. (Over $\mathbb{C}$, the conjugate transpose is used.) Note that the space orthogonal to U is not necessarily a complement to U.

On matrix spaces

Given a matrix space $\mathcal{A} \leq \mathrm{M}(s \times t, \mathbb{F})$, the image of $U \leq \mathbb{F}^t$ under $\mathcal{A}$ is $\mathcal{A}(U) := \langle \cup_{A \in \mathcal{A}} A(U) \rangle$. The dimension of $\mathcal{A}$ is denoted by $\dim(\mathcal{A})$. The (maximum) rank of $\mathcal{A}$ is $\mathrm{rk}(\mathcal{A}) := \max\{\mathrm{rk}(A) : A \in \mathcal{A}\}$. Let $\mathcal{B} \leq \mathrm{M}(s \times t, \mathbb{F})$ be another matrix space. We say that $\mathcal{A}$ and $\mathcal{B}$ are *equivalent*, if there exist $C \in \mathrm{GL}(s, \mathbb{F})$ and $D \in \mathrm{GL}(t, \mathbb{F})$, such that $\mathcal{A} = C\mathcal{B}D := \{CBD : B \in \mathcal{B}\}$. When working with $\mathcal{A}$, an equivalence transformation is meant to left multiply $\mathcal{A}$ with some $C \in \mathrm{GL}(s, \mathbb{F})$ and right multiply it with some $D \in \mathrm{GL}(s, \mathbb{F})$.

On alternating matrices

Let $A, B \in \Lambda(n, \mathbb{F})$. We say that A and B are *isometric*, if there exists $T \in \mathrm{GL}(n, \mathbb{F})$, such that $A = T^t B T$. Given a dimension-d $U \leq \mathbb{F}^n$ represented by $T \in \mathrm{GL}(n \times d, \mathbb{F})$, the *restriction* of A to U by T, denoted as $A|_{U,T}$, is $T^t A T \in \Lambda(d, \mathbb{F})$. The radical of A is the subspace $\{u \in \mathbb{F}^n : \forall v \in \mathbb{F}^n, v^t A u = 0\}$, which is just $\ker(A)$. The rank of A is always even. If $\mathrm{rk}(A) = 2r$, then A is isometric to $\begin{bmatrix} \mathbf{0} & I_r & \mathbf{0} \\ -I_r & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}$ (see e.g. [71, Chap. XV, Sec. 8]). We say that $A \in \Lambda(n, \mathbb{F})$ is *non-degenerate*, if A is full-rank (so n is even).

On alternating matrix spaces

Let $\mathcal{A}, \mathcal{B} \leq \Lambda(n, \mathbb{F})$. We say that $\mathcal{A}$ and $\mathcal{B}$ are *isometric*, if there exists $T \in \mathrm{GL}(n, \mathbb{F})$, such that $\mathcal{A} = T^t \mathcal{B} T := \{T^t B T : B \in \mathcal{B}\}$. Given a dimension-$d$ $U \leq \mathbb{F}^n$ represented by $T \in \mathrm{GL}(n \times d, \mathbb{F})$, the *restriction* of $\mathcal{A}$ on U via T is $\mathcal{A}|_{U,T} := \{T^t A T : A \in \mathcal{A}\} \leq \Lambda(d, \mathbb{F})$. When it does not cause confusion, we may not write T explicitly, and just say the restriction of $\mathcal{A}$ to U, denoted by $\mathcal{A}|_U$. This corresponds to the concept of induced subgraphs in graph theory. Indeed, we see that U is an isotropic space if and only if $\mathcal{A}|_U$ is the zero (alternating matrix) space. Given $v \in \mathbb{F}^n$, the *radical of v in $\mathcal{A}$*, denoted as $\mathrm{rad}_{\mathcal{A}}(v)$, is $\{u \in \mathbb{F}^n : \forall A \in \mathcal{A}, u^t A v = 0\}$ which is a subspace of $\mathbb{F}^n$. Elements in $\mathrm{rad}_{\mathcal{A}}(v)$ correspond to non-neighbours in graph theory. The *codegree* of v in $\mathcal{A}$, denoted as $\mathrm{codeg}_{\mathcal{A}}(v)$, is $\dim(\mathrm{rad}_{\mathcal{A}}(v))$. Note that $\mathrm{codeg}_{\mathcal{A}}(v) \geq 1$ for nonzero v, as $v \in \mathrm{rad}_{\mathcal{A}}(v)$. The *degree* of v in $\mathcal{A}$ is $\deg_{\mathcal{A}}(v) := n - \mathrm{codeg}_{\mathcal{A}}(v)$. More generally, for $U \leq \mathbb{F}^n$, $\mathrm{rad}_{\mathcal{A}}(U) = \{v \in \mathbb{F}^n : \forall u \in U, \forall A \in \mathcal{A}, u^t A v = 0\}$. When $\mathcal{A}$ is clear from the context, we may drop the subscript $\mathcal{A}$ in $\mathrm{rad}_{\mathcal{A}}(v)$, $\mathrm{codeg}_{\mathcal{A}}(v)$, $\deg_{\mathcal{A}}(v)$, etc.. It is easy to see that for any $v \in \mathrm{rad}(U)$, we have $U \leq \mathrm{rad}(v)$, or in other words, $U \leq \mathrm{rad}(\mathrm{rad}(U))$.

A vector $v \in \mathbb{F}^n$ is called *isolated* in $\mathcal{A}$, if for any $A \in \mathcal{A}$, $Av = \mathbf{0}$, which is equivalent to say that $\deg(v) = 0$. This corresponds to the concept of isolated vertices in graph theory. The *radical* of $\mathcal{A}$, $\mathrm{rad}(\mathcal{A})$, is the subspace of $\mathbb{F}^n$ consisting of all isolated vectors. We say that $\mathcal{A}$ is *non-degenerate*, if $\mathrm{rad}(\mathcal{A}) = \mathbf{0}$, and *degenerate* otherwise. If $\mathcal{A} \leq \Lambda(n, \mathbb{F})$ is degenerate with $\dim(\mathrm{rad}(\mathcal{A})) = d > 0$, then $\mathcal{A}$ is isometric to $\mathcal{A}'$ where each $A \in \mathcal{A}'$ is of the form $\begin{bmatrix} A' & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}$, where $A' \in \Lambda(n - d, \mathbb{F})$.

Sets, tuples, and spaces

Let $\mathcal{A} \leq \Lambda(n, \mathbb{F})$ be given by a linear basis $A_1, \ldots, A_m \in \Lambda(n, \mathbb{F})$. We can collect them as a set $\mathsf{A} = \{A_1, \ldots, A_m\} \subseteq \Lambda(n, \mathbb{F})$. Sometimes it is also useful to impose an order on them, and form a tuple $\mathbf{A} = (A_1, \ldots, A_m) \in \Lambda(n, \mathbb{F})^m$. We shall use calligraphic fonts for spaces, bold fonts for tuples, and sans serif fonts for sets.

Suppose $\mathcal{A} \leq \Lambda(n, \mathbb{F})$ is given by a linear basis $A_1, \ldots, A_m \in \Lambda(n, \mathbb{F})$. Then it is clear that, given $U \leq \mathbb{F}^n$, for any $u, u' \in U$ and any $A \in \mathcal{A}$, $u^t A u' = 0$, if and only if for any $u, u' \in U$ and any $i \in [m]$, $u^t A_i u' = 0$. We therefore can define isotropic spaces, and isotropic decompositions, for sets or tuples of alternating matrices. In particular, since alternating matrix tuples represent alternating bilinear maps naturally (see Appendix B), this observation suggests that isotropic spaces and decompositions for alternating matrix spaces and for alternating bilinear maps are basically the same object. Furthermore, many, though not all, concepts introduced in Section 2 about alternating matrix spaces can be translated natually to alternating bilinear maps, including degrees, degeneracy, radicals, etc..

(Indeed, some notions there are actually borrowed from alternating bilinear maps.) On the other hand, note that the maximum rank in $\mathcal{A}$ is more natually associated with the space perspective. More discussions on the relation between these two notions are in Appendix B.

Therefore, in the context of isotropic spaces and decompositions, the choices between alternating matrix spaces and tuples usually do not matter much. The tuple perspective is more natural from the algorithm perspective, because the input of an alternating matrix space to an algorithm is usually an ordered basis. The space perspective is more natural for forming the analogy with graphs, and more naturally allows for some more notions including the maximum rank, which is important in e.g. the proof of Theorem 9. The set perspective will be used in the quantum variant of the theory in Section 12. Therefore, it is best to keep all three perspectives in mind, and see how they fit into our problems.

2.1 Computational models

We will work with two computational models, depending on the problems. The first model may be called the exact model; see e.g. [80]. This is the model to work with, if field extensions are unavoidable. In this model, input matrices or vectors are over a field $\mathbb{E}$ where $\mathbb{E}$ is a finite field extension over its prime field $\mathbb{F}$, so $\mathbb{F}$ is either a field of prime order or $\mathbb{Q}$. Suppose $\dim_{\mathbb{F}}(\mathbb{E}) = d$. Then $\mathbb{E}$ is an extension of $\mathbb{F}$ by a single generating element α. We represent α by the minimal polynomial of α over $\mathbb{F}$, and an isolating interval for α in the case of $\mathbb{R}$, or an isolating rectangle for α in the case of $\mathbb{C}$. Note that from this representation, one can approximate the numerical value of α arbitrarily closely. When we say that we work over $\mathbb{R}$ or $\mathbb{C}$, the input is given as over some number field $\mathbb{E}$ in $\mathbb{R}$ or $\mathbb{C}$. The algorithm is allowed to work with extension fields of $\mathbb{E}$ in $\mathbb{R}$ or $\mathbb{C}$, as long as the extension degrees are polynomially bounded.

The second model may be called the arithmetic model. In this model, only basic field operations are performed, and the issue of working with different field extensions does not arise. Still, over number fields we will be concerned with the bit complexities, though it is possible that we may be able to only bound the number of arithmetic steps, but not the bit complexities.

We shall mostly work with $\mathbb{F}_q$, $\mathbb{Q}$, $\mathbb{R}$, and $\mathbb{C}$ in this article, though some results extend to number fields naturally. Sometimes, we make further restrictions like requiring q to be odd, or the input to be integral.

3 Basic facts and properties

In this section we collect some basic results about isotropic spaces and isotropic decompositions.

On the definitions

The following is a somewhat more intuitive definition of these two notions.

Let $\mathcal{A} \leq \Lambda(n, \mathbb{F})$, and let $U \leq \mathbb{F}^n$ be an isotropic space. Suppose $d = \dim(U)$. Then form a change of basis matrix $T \in \mathrm{GL}(n, \mathbb{F})$, such that its first d columns form a basis of U, and the rest columns together with the first d ones span $\mathbb{F}^n$. Then for any $A \in \mathcal{A}$, we see that $T^t AT$ is of the form $\begin{bmatrix} \mathbf{0} & B \\ -B^t & C \end{bmatrix}$ where $\mathbf{0}$ is of size $d \times d$. It is not hard to see that $\mathcal{A}$ has a dimension-d isotropic space if and only if there exists such a change of basis matrix T.

Similarly, $\mathcal{A}$ has an isotropic c-decomposition, if and only if there exist $T \in \mathrm{GL}(n, \mathbb{F})$, and $d_1, \ldots, d_c \in \mathbb{Z}^+$ with $\sum_i d_i = n$, such that for any $A \in \mathcal{A}$, $T^t A T = \begin{bmatrix} \mathbf{0} & A_{1,2} & \ldots & A_{1,c} \\ -A_{1,2}^t & \mathbf{0} & \ldots & A_{2,c} \\ \vdots & \vdots & \ddots & \vdots \\ -A_{1,c}^t & -A_{2,c}^t & \ldots & \mathbf{0} \end{bmatrix}$, where the ith $\mathbf{0}$ on the diagonal is of size d_i.

Computing the radical

For many problems about isotropic spaces, given a degenerate $\mathcal{A} \leq \Lambda(n, \mathbb{F})$, usually it is possible to reduce to the non-degenerate case. This is facilitated by the fact that the radical of $\mathcal{A}$ is easy to compute.

▶ **Observation 16.** *Suppose $\mathcal{A} = \langle A_1, \ldots, A_m \rangle \leq \Lambda(n, \mathbb{F})$ is given by a linear basis. There is a polynomial-time algorithm that computes a linear basis of* $\mathrm{rad}(\mathcal{A})$.

Proof. Observe that $\mathrm{rad}(\mathcal{A}) = \cap_{i=1}^n \ker(A_i)$. Computing $\ker(A_i)$ and the intersection of $\ker(A_i)$'s are standard linear algebraic tasks that can be performed as stated. ◀

Isotropic spaces and radicals of subspaces

Let $\mathcal{A} \leq \Lambda(n, \mathbb{F})$. Recall that for $U \leq \mathbb{F}^n$, we defined $\mathrm{rad}_{\mathcal{A}}(U) = \{v \in \mathbb{F}^n : \forall u \in U, v^t A u = 0\}$. The following observation is immediate.

▶ **Observation 17.** *Let $U \leq \mathbb{F}^n$ and $\mathcal{A} \leq \Lambda(n, \mathbb{F})$. Then we have the following.*

1. *U is an isotropic space of $\mathcal{A}$ if and only if $U \subseteq \mathrm{rad}(U)$.*
2. *U is a maximal isotropic space of $\mathcal{A}$ if and only if $U = \mathrm{rad}(U)$.*

An easy application of Observation 17 gives a greedy algorithm for computing one maximal isotropic space.

▶ **Proposition 18.** *Given a matrix space $\mathcal{A} \leq \Lambda(n, \mathbb{F})$, a maximal isotropic space can be computed in polynomially many arithmetic steps.*

Proof. We first present the algorithm. Recall that e_i is the ith standard basis vector of $\mathbb{F}^n$.

1. Let $U = \langle e_1 \rangle$.
2. While $U \subsetneq \mathrm{rad}(U)$:
   a. Take any $u \in \mathrm{rad}(U) \setminus U$.
   b. $U \leftarrow \langle U, u \rangle$.
3. Output U.

To see the correctness, note that in Step (2.a), by the choice of u, we have $U \subseteq \mathrm{rad}(\langle U, u \rangle)$ and $u \in \mathrm{rad}(\langle U, u \rangle)$, so $\langle U, u \rangle$ is an isotropic space by Observation 17 (1). In Step (3), U satisfies $U = \mathrm{rad}(U)$, so U is maximal by Observation 17 (2).

To see the running time, note that the while loop will be executed by at most n times, since $\dim(U)$ increases by 1 in each execution. Each step involves basic linear algebraic computations which require only polynomially many arithmetic operations. ◀

Over $\mathbb{R}$ or $\mathbb{C}$, the bit sizes in the above algorithm may blow up, at least with a straightforward implementation, due to the iterative computations of the radicals.

On field extensions

Let $\mathbb{K}$ be an extension field of $\mathbb{F}$. Given $\mathcal{A} \leq \Lambda(n, \mathbb{F})$, we let $\mathcal{A}_{\mathbb{K}} \leq \Lambda(n, \mathbb{K})$ be the alternating matrix spaces when we allow linear combinations over $\mathbb{K}$, or in other words, $\mathcal{A}_{\mathbb{K}} = \mathcal{A} \otimes_{\mathbb{F}} \mathbb{K}$. We may write $\mathcal{A}$ as $\mathcal{A}_{\mathbb{F}}$ for further distinction. For $\mathcal{A}_{\mathbb{K}}$ we allow for isotropic spaces to come from $\mathbb{K}^n$. Since isotropic spaces and c-decompositions of $\mathcal{A}_{\mathbb{F}}$ naturally give isotropic spaces and c-decompositions of $\mathcal{A}_{\mathbb{K}}$, we have the following (see also [50, Lemma 6]).

▶ **Proposition 19.** *Let $\mathcal{A}_{\mathbb{F}} \leq \Lambda(n, \mathbb{F})$ and $\mathcal{A}_{\mathbb{K}} \leq \Lambda(n, \mathbb{K})$ be as above. Then $\alpha(\mathcal{A}_{\mathbb{F}}) \leq \alpha(\mathcal{A}_{\mathbb{K}})$, and $\chi(\mathcal{A}_{\mathbb{F}}) \geq \chi(\mathcal{A}_{\mathbb{K}})$.*

In Proposition 19, the inqualities for $\alpha(\cdot)$ and $\chi(\cdot)$ could be strict, as shown by Buhler, Gupta, and Harris [22, pp. 277]. Recall that in Section 1.3 we defined $\alpha(\mathbb{F}, n, m) = \{\alpha(\mathcal{A}) : \mathcal{A} \leq \Lambda(n, \mathbb{F}), \dim(\mathcal{A}) = m\}$, and Buhler et al. showed that $\alpha(\mathbb{F}, n, m) \leq \lceil \frac{m+2n}{m+2} \rceil$, when $m > 1$ and the characteristic of $\mathbb{F}$ is not 2. Furthermore the equality can be attained for algebraically closed fields. Buhler et al. then demonstrated examples over $\mathbb{F}_q$ and $\mathbb{Q}$, for which the inequality is strict. Consider the example over $\mathbb{Q}$, which is some $\mathcal{A} \leq \Lambda(n, \mathbb{Q})$ of dimension n. They show that $\alpha(\mathcal{A}_{\mathbb{Q}}) = 1$ and $\alpha(\mathcal{A}_{\mathbb{C}}) = 2$, which is equivalent to that $\chi(\mathcal{A}_{\mathbb{Q}}) = n$ and $\chi(\mathcal{A}_{\mathbb{C}}) \leq n - 1$. This gives the desired separations. Some interesting discussions on $\mathbb{R}$ vs $\mathbb{C}$ can be found in [22, Sec. 3].

3.1 Isotropic spaces and decompositions in group theory and manifold theory

In this subsection, we explain the origins of isotropic spaces and decompositions for alternating matrix spaces in group theory and manifold theory. These are classical, so the purpose here is to provide references for interested readers who have not met with these before.

Group theory

Let p be a prime > 2. We consider p-groups of class 2 and exponent p, that is, a group G of prime power order, with the commutator subgroup $[G, G]$ contained in the centre $Z(G)$, and every group element $g \in G$ satisfying $g^p = 1$.

Let P be such a group of order p^ℓ. We have that the commutator subgroup $[P, P] \cong \mathbb{Z}_p^m$, and the commutator quotient $P/[P, P] \cong \mathbb{Z}_p^n$. The commutator bracket then gives an alternating bilinear map $\phi : P/[P, P] \times P/[P, P] \to [P, P]$, or, after fixing bases of $[P, P]$ and $P/[P, P]$, an alternating bilinear map $\phi : \mathbb{Z}_p^n \times \mathbb{Z}_p^n \to \mathbb{Z}_p^m$. On the other hand, given an alternating bilinear map $\phi : \mathbb{F}_p^n \times \mathbb{F}_p^n \to \mathbb{F}_p^m$, one can construct a p-group of class 2 and exponent p, P_ϕ, called the *Baer group* corresponding to ϕ, as follows. The group elements are $(v, u) \in \mathbb{F}_p^n \times \mathbb{F}_p^m$, and the group multiplication $\circ$ is by $(v_1, u_1) \circ (v_2, u_2) = (v_1 + v_2, u_1 + u_2 + \frac{1}{2} \cdot \phi(v_1, v_2))$. This sets up a two-way correspondence between p-groups of class 2 and exponent p and alternating biliear maps.

Having set up the connection, let us see the group-theoretic interpretations of isotropic spaces and decompositions. The following is classical: an isotropic space of ϕ corresponds to a normal abelian subgroup of P containing the commutator subgroup (see e.g. [4]). More recently, Lewis and Wilson proposed the concept of a hyperbolic pair of P [75], which just consists of two normal abelian subgroups of P which together generate P, and whose intersection equals $[P, P]$. A natural generalization is then the following. A hyperbolic c-system of P consists of c normal abelian subgroups $A_1, A_2, \ldots, A_c$, such that P is generated by A_i, and for any $i, j \in [c]$, $i \neq j$, $A_i \cap A_j = [P, P]$. A hyperbolic c-system then naturally corresponds to an isotropic c-decomposition of ϕ.

Manifold theory

We then turn to the manifold theory side. We shall just walk through some examples of the connection, and point the interested reader to the survey [41] for more detailed information.

Let M be a compact Kähler manifold. Let $H^i(M;\mathbb{C})$ be the ith cohomology group of M with coefficients in $\mathbb{C}$. The cup product $\smile: H^1(M;\mathbb{C}) \times H^1(M;\mathbb{C}) \to H^2(M;\mathbb{C})$ is a skew-symmetric bilinear map. Then as an application of results by Castelnuovo and de Franchis, Catanese [29] showed that there exists a constant holomorphic map $f : M \to C$, where C is a curve of genus $g \geq 2$, if and only if, $\smile$ has a dimension-g maximal isotropic space. Catanese went on to generalize this connection much further in [29].

Let M be a smooth closed orientable n-dimensional manifold. Let $H^i(M;\mathbb{Q})$ be the ith cohomology group of M with coefficients in $\mathbb{Q}$. Gelbukh showed that an isotropic space of $\smile$ corresponds to a geometric structure on M [50, Lemma 10, Definition 11, Theorem 13], called an isotropic system, which consists of smooth closed orientable connected codimension-one submanifolds that are homologically non-intersecting, homologically independent, and intersecting transversely. The isotropic index defined in [50] for an alternating bilinear map is just the isotropic number introduced in Definition 1. In particular, our Theorem 3 shows that computing this isotropic index is NP-hard. The relations of isotropic indices with other basic notions in manifold cohomology, including the first Betti number and the co-rank of the fundamental group, are also studied there.

4 Proof of Theorem 3

Recall that we have a graph $G = ([n], E)$, and an alternating matrix space $\mathcal{A}_G$ which is spanned by those elementary alternating matrices $A_{i,j}$ where $\{i, j\} \in E$.

(1) Isotropic spaces and independent sets

We need to show that $G = ([n], E)$ has a size-s independent set if and only if $\mathcal{A}_G$ has a dimension-s isotropic space.

For the only if direction, let $T = \{i_1, \ldots, i_s\}$ be a size-s independent set of G. Let U be the subspace of $\mathbb{F}^n$ spanned by $e_{i_1}, \ldots, e_{i_s}$; recall that e_i denotes the ith standard basis vector of $\mathbb{F}^n$. It is easy to verify that U is an isotropic space of $\mathcal{A}$ of dimension s.

For the if direction, let $U = \langle u_1, \ldots, u_s \rangle$ be a dimension-s isotropic space of $\mathcal{A}_G$. We form an $n \times s$ matrix U such that the ith column of U is u_i, that is, $U = [u_1, \ldots, u_s] \in \mathrm{GL}(n \times s, \mathbb{F})$. Suppose $U^t = [w_1, \ldots, w_n] \in \mathrm{GL}(s \times n, \mathbb{F})$, $w_i \in \mathbb{F}^s$. Since U is of rank s, there exist integers $i_1, \ldots, i_s$, $1 \leq i_1 < \cdots < i_s \leq n$, such that $w_{i_1}, \ldots, w_{i_s}$ are linearly independent. We now claim that $\{i_1, \ldots, i_s\}$ forms an independent set of the original graph $G = ([n], E)$. If not, suppose $\{i_j, i_{j'}\}$, $1 \leq j < j' \leq s$, is in E. As U is an isotropic space of $\mathcal{A}_G$, we have that for any $\{k, \ell\} \in E$, $U^t A_{k,\ell} U = \mathbf{0}$. As $A_{k,\ell} = e_k e_\ell^t - e_\ell e_k^t$, we have

$$U^t A_{k,\ell} U = U^t (e_k e_\ell^t - e_\ell e_k^t) U = w_k w_\ell^t - w_\ell w_k^t = 0,$$

which implies that w_k and w_ℓ are linearly dependent. It follows that w_{i_j} and $w_{i_{j'}}$ are linearly dependent. We then arrive at a contradiction.

(2) Isotropic decompositions and vertex colorings

We need to show that $G = ([n], E)$ has a vertex c-coloring if and only if $\mathcal{A}_G$ has an isotropic c-decomposition.

For the only if direction, assume G has a vertex c-coloring, and let $[n] = T_1 \uplus T_2 \uplus \cdots \uplus T_c$ be a partition of $[n]$ into disjoint union of independent sets. Suppose $|T_j| = t_j$, and $T_j = \{i_{j,1}, \ldots, i_{j,t_j}\} \subseteq [n]$. Let $U_j = \langle e_{i_{j,1}}, \ldots, e_{i_{j,t_j}} \rangle \leq \mathbb{F}^n$, so $\mathbb{F}^n = U_1 \oplus U_2 \oplus \cdots \oplus U_c$. By (1), every U_j is an isotropic space. This gives an isotropic c-decomposition of $\mathcal{A}_G$.

For the if direction, let $\mathbb{F}^n = U_1 \oplus U_2 \oplus \cdots \oplus U_c$ be an isotropic c-decomposition. Let $d_i = \dim(U_i)$, and $b_i = \sum_{j=1}^{i} d_j$, for $i \in [c]$. Set $b_0 = 0$. Let $P = [p_1, \ldots, p_n]$ be an $n \times n$ invertible matrix, where $p_i \in \mathbb{F}^n$, such that $p_{b_{i-1}+1}, \ldots, p_{b_i}$ form a basis of U_i. By abuse of notation, we also let $U_i = [p_{b_{i-1}+1}, \ldots, p_{b_i}] \in \mathrm{GL}(n \times d_i, \mathbb{F})$, so $P = [U_1, \ldots, U_c]$. Let $W_i = U_i^t = [w_{i,1}, \ldots, w_{i,n}] \in \mathrm{GL}(d_i \times n, \mathbb{F})$. Since U_i is an isotropic space, by (1), we know that for any $\{k, \ell\} \in E$, $w_{i,k}$ and $w_{i,\ell}$ are linearly dependent.

We then use the following simple linear algebraic result, which is a consequence of the Laplacian expansion. For $A, B \subseteq [n]$ of the same size, we let $C|_{A,B}$ to denote the submatrix of C with row indices from A and column indices from B.

▶ **Lemma 20.** *Let $P = [U_1, \ldots, U_c] \in \mathrm{GL}(n, \mathbb{F})$, $U_i \in \mathrm{GL}(n \times d_i, \mathbb{F})$, and suppose $\det(P) \neq 0$. Then there exists a partition of $[n] = T_1 \uplus T_2 \uplus \cdots \uplus T_c$, where $|T_i| = d_i$, such that $\forall i \in [c]$, $\mathrm{rk}(U_i|_{T_i,[d_i]}) = d_i$.*

We claim that the partition of $[n] = T_1 \uplus T_2 \uplus \cdots \uplus T_c$ from Lemma 20 gives a vertex c-coloring of G. To see this, observe that, the condition $\mathrm{rk}(U_i|_{T_i,[d_i]}) = d_i$ is equivalent to that the vectors $w_{i,j}$, $j \in T_i$, are linearly independent. This implies that G cannot have edges of the form $\{k, \ell\}$ where $k, \ell \in T_i$, as otherwise $w_{i,k}$ and $w_{i,\ell}$ would be linearly dependent. Hence T_i is an independent set for any $i \in [c]$. This completes the proof of the second part of Theorem 3.

5 An exposition of the proof of Theorem 5

As mentioned in Section 1.3, we give an exposition of the proof of Theorem 5 for $\mathbb{F}_q$ in [20], using some ingredients from [59] to handle $\mathbb{R}$ and $\mathbb{C}$. The main purpose is to give the reader a flavor of how the so-called $*$-algebra technique is applied in this context. We could not give all the details here, as that would be too long and unnecessary; the interested reader may wish to go to [21, 59, 111], which contain detailed proofs for using $*$-algebras to solve several closely related problems.

Recall that we are given $\mathcal{A} = \langle A_1, \ldots, A_m \rangle \leq \Lambda(n, \mathbb{F})$, and our goal is to find a non-trivial direct sum decomposition $\mathbb{F}^n = U_1 \oplus U_2$, such that $\mathcal{A}|_{U_i} = \mathbf{0}$ for $i = 1, 2$. We first reduce to the non-degenerate setting as follows. Suppose $\mathcal{A}$ is degenerate, that is, there exists $T \leq \mathbb{F}^n$ of dimension n' such that $\mathcal{A}' = \mathcal{A}|_T$ is non-degenerate. Then it is not hard to verify that $\mathcal{A}$ admits an isotropic 2-decomposition if and only if $\mathcal{A}'$ admits an isotropic 2-decomposition.

In the following we assume that $\mathcal{A}$ is non-degenerate. Let $\mathbf{A} = (A_1, \ldots, A_m) \in \Lambda(n, \mathbb{F})^m$. The adjoint algebra of $\mathbf{A}$ is defined as $\mathrm{Adj}(\mathbf{A}) := \{D \in \mathrm{M}(n, \mathbb{F}) : \exists B \in \mathrm{M}(n, \mathbb{F}), \forall i \in [m], B^t A_i = A_i D\}$. Since $\mathbf{A}$ is non-degenerate, if such B exists, then it is unique. Then a natural involution (an anti-automorphism of order at most 2) on $\mathrm{Adj}(\mathbf{A})$ is to send $D \in \mathrm{Adj}(\mathbf{A})$ to this unique B satisfying $B^t A_i = A_i D$ for any $i \in [m]$, also denoted as D^*. The adjoint algbera is the key device for the algorithm.

We now translate the isotropic 2-decomposition problem for $\mathcal{A}$, and therefore $\mathbf{A}$, to a problem about $\mathrm{Adj}(\mathbf{A})$. Any direct sum decomposition $\mathbb{F}^n = U_1 \oplus U_2$ can be encoded as a projection matrix $P \in \mathrm{M}(n, \mathbb{F})$, that is, $P^2 = P$, $\mathrm{im}(P) = U_1$, $\ker(P) = U_2$. The key observation in [20] is that, P corresponds to an isotropic 2-decomposition if and only if $P \in \mathrm{Adj}(\mathbf{A})$ and $P^* = I - P$. This means that we need to search for an idempotent P in $\mathrm{Adj}(\mathbf{A})$ satisfying $P^* = I - P$. Following [20], we call such an idempotent a hyperbolic idempotent.

To do that, we utilize the $*$-algebra structure of $\mathrm{Adj}(\mathbf{A})$. For this, we recall in a nutshell the structure of $*$-algebras. Let $\mathfrak{A}$ be a $*$-algebra. The Jacobson radical of $\mathfrak{A}$, denoted by $\mathrm{rad}(\mathfrak{A})$, is the largest nilpotent ideal of $\mathfrak{A}$, and it is invariant under $*$. The factor algebra $\mathfrak{A}/\mathrm{rad}(\mathfrak{A})$ is semi-simple, namely it is a direct sum of simple algebras. Let $\mathfrak{A}/\mathrm{rad}(\mathfrak{A}) \cong S_1 \oplus \cdots \oplus S_k$, where each S_i is simple. The $*$ either switches between S_i and S_j, $i \neq j$, or preserves S_i. Both cases are referred to as $*$-simple. The former case is called the exchange type. In the latter case, S_i is a simple algebra with an involution. Over any field, Wedderburn's theory gives a characterization of such simple algebras (see e.g. [5, Chap. 5]). Based on this, involutions on simple algebras are also classified [3, Chap. X.4], and explicit lists for $\mathbb{F}_q$, $\mathbb{R}$, and $\mathbb{C}$ can be found in [59].

Given this structure, the idea is to reduce the search for a hyperbolic idempotent from general $*$-algebras to simple $*$-algebras. Clearly, if $\mathfrak{A}$ contains a hyperbolic idempotent, then $\mathfrak{A}/\mathrm{rad}(\mathfrak{A})$, and each $*$-simple summand of $\mathfrak{A}/\mathrm{rad}(\mathfrak{A})$, all contain hyperbolic idempotents. On the other hand, suppose each $*$-simple summand of $\mathfrak{A}/\mathrm{rad}(\mathfrak{A})$ contains a hyperbolic idempotent. Then the sum of these idempotents is a hyperbolic idempotent for $\mathfrak{A}/\mathrm{rad}(\mathfrak{A})$. From here, to obtain a hyperbolic idempotent for $\mathfrak{A}$, we can use the classical idempotent lifting technique (see e.g. [112, Lemma 5.10]).

Therefore, it remains to handle the $*$-simple case. Let $\mathbb{K}$ denote some appropriate division algebra containing $\mathbb{F}$. The reader may as well think of $\mathbb{K}$ as an extension field, as for $\mathbb{F}_q$, $\mathbb{R}$, and $\mathbb{C}$, the only "non-field" case is the quaternion algebra over $\mathbb{R}$. The exchange type is easy to handle: it is $*$-isomorphic to $\mathrm{M}(\ell, \mathbb{K}) \oplus \mathrm{M}(\ell, \mathbb{K})^{op}$ with $(A, B)^* = (B, A)$. So one hyperbolic idempotent can be $(I, \mathbf{0})$. The simple case is more interesting. It is isomorphic to $\mathrm{M}(\ell, \mathbb{K})$ with the involution defined by some non-degenerate classical form F; this includes alternating, symmetric, and Hermitian ones.[8] Then for $A \in \mathrm{M}(\ell, \mathbb{K})$, $A^* = F^{-1}A^\dagger F$, where $\dagger$ denotes either transpose (for alternating and symmetric) or conjugate transpose (for Hermitian). The problem is then to find a hyperbolic idempotent, or equivalently, an isotropic 2-decomposition, for this form F. But now this is a single form, so one can bring it to say a canonical form, and examine case by case.

For example, over $\mathbb{C}$, there are two types, symmetric and alternating. (The Hermitian type does not appear because $*$ is required to preserve $\mathbb{C}$.) A non-degenerate alternating form can always be transformed to $\begin{bmatrix} \mathbf{0} & I \\ -I & \mathbf{0} \end{bmatrix}$, so isotropic 2-decomposable. A non-degenerate symmetric form can be transformed to the identity matrix I. When I is of odd size ℓ, then it does not admit an isotropic 2-decomposition. Because if so, then I is isometric to $J = \begin{bmatrix} \mathbf{0} & A \\ A^t & \mathbf{0} \end{bmatrix}$, where A is of size $i \times (\ell - i)$. But then $\mathrm{rk}(J)$ is either $2i$ or $2(\ell - i)$, an even number, so J is degenerate, a contradiction. When I is of even size ℓ, then it has an isotropic 2-decomposition. This is because we have $\begin{bmatrix} 1 & i \\ i & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & i \\ i & 1 \end{bmatrix} = \begin{bmatrix} 0 & 2i \\ 2i & 0 \end{bmatrix}$. We can then use this to bring I to $\begin{bmatrix} \mathbf{0} & 2iI' \\ 2iI' & \mathbf{0} \end{bmatrix}$, where I' is the $\ell/2 \times \ell/2$ identity matrix. Similar reasonings can be carried over $\mathbb{R}$ and $\mathbb{F}_q$.

To make the above procedure constructive, we need to compute the algebra structure efficiently. This can be done, over $\mathbb{F}_q$ with randomness [93] and over $\mathbb{C}$ deterministically [43,94]. We also need to compute the $*$-algebra structure, e.g. the forms associated with a simple $*$-algebra, by [21, 59, 111]. Finally, we need to compute the canonical forms of various forms by [107, 113].

[8] While in principle this is correct, depending on the field, some type may not exist. For details see [59].

For the algorithm analysis, we work in the exact model, as going to extension fields is already unavoidable in computing the algebra structure. Over $\mathbb{F}_q$, one can always recover, from the solutions in the simple cases, an explicit hyperbolic projection matrix P over the original field, and then we can obtain the bases of the two subspaces in an isotropic 2-decomposition by computing the image and kernel of P. Over $\mathbb{R}$ and $\mathbb{C}$, one can represent this projection matrix as a product of matrices over different extension fields [59, Sec. 3.5].

This concludes an exposition of the algorithm.

6 Proof of Theorem 7

Let us first recall the alternative definition of non-commutative rank for a slightly more general situation. Given $\mathcal{B} \leq \mathrm{M}(s \times t, \mathbb{F})$, an isotropic pair is a pair of vector spaces (U, V), $U \leq \mathbb{F}^s$, $V \leq \mathbb{F}^t$, such that for any $u \in U, v \in V$, and any $B \in \mathcal{B}$, we have $u^t B v = 0$. The non-commutative rank of $\mathcal{B}$ is then defined as $\mathrm{ncrk}(\mathcal{B}) := (s+t) - \max\{c+e : c = \dim(U), e = \dim(V), (U, V) \text{ is an isotropic pair of } \mathcal{B}\}$. Note that the recent works [48, 61] mostly deal with the setting that $s = t$. Suppose $\mathrm{ncrk}(\mathcal{B}) = r$. By equivalence transformations, we can assume that every $\mathcal{B}$ is of the form $\begin{bmatrix} B_1 & B_2 \\ \mathbf{0} & B_3 \end{bmatrix}$, where B_2 is of size $a \times b$ such that $a + b = r$.

We review the setting for Theorem 7. Let $\mathcal{A} \leq \Lambda(n, \mathbb{F})$ be a bipartite alternating matrix space. By isometric transformations, we can assume that every $A \in \mathcal{A}$ is of the form $\begin{bmatrix} \mathbf{0} & B \\ -B^t & \mathbf{0} \end{bmatrix}$ where $B \in \mathrm{M}(s \times t, \mathbb{F})$, $s + t = n$. All such B form a matrix space $\mathcal{B} \leq \mathrm{M}(s \times t, \mathbb{F})$. We call such $\mathcal{B}$ a matrix space induced from the bipartite structure of $\mathcal{A}$.

Before proving Theorem 7, let us examine some examples of isotropic spaces of $\mathcal{A}$.

1. First note that $\alpha(\mathcal{A}) \geq \max\{s, t\}$.
2. Second, suppose $\mathrm{ncrk}(\mathcal{B}) = r$, so there exists $P \in \mathrm{GL}(s, \mathbb{F})$ and $Q \in \mathrm{GL}(t, \mathbb{F})$, such that every matrix in $P\mathcal{B}Q$ is of the form $\begin{bmatrix} B_1 & B_2 \\ \mathbf{0} & B_3 \end{bmatrix}$, where B_2 is of size $a \times b$ such that $a + b = r$. Let $R = \begin{bmatrix} P^t & \mathbf{0} \\ \mathbf{0} & Q \end{bmatrix}$. Then we have
$$\begin{aligned} R^t A R &= \begin{bmatrix} P & \mathbf{0} \\ \mathbf{0} & Q^t \end{bmatrix} \begin{bmatrix} \mathbf{0} & B \\ -B^t & \mathbf{0} \end{bmatrix} \begin{bmatrix} P^t & \mathbf{0} \\ \mathbf{0} & Q \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{0} & PBQ \\ -(PBQ)^t & \mathbf{0} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{0} & \mathbf{0} & B_1 & B_2 \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & B_3 \\ -B_1^t & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ -B_2^t & -B_3^t & \mathbf{0} & \mathbf{0} \end{bmatrix}, \end{aligned}$$
from which we get an isotropic space, consisting of the zero blocks in the middle part, of dimension $n - r$. (Note that the zero matrix at the $(2, 3)$ position is of size $(s - a) \times (t - b)$, and the isotropic space corresponding to the zero blocks in the middle part is of size $(s - a) + (t - b) = (s + t) - (a + b) = n - r$.)
3. The third example we now describe represents a difference from the graph-theoretic setting. Suppose $s = t$, and every $B \in \mathrm{M}(s, \mathbb{F})$ is symmetric. So $n = 2s$. Then let $U = \{u \in \mathbb{F}^n = \mathbb{F}^{2s} : u = (u_1, \ldots, u_s, u_1, \ldots, u_s)^t \in \mathbb{F}^n\} \leq \mathbb{F}^n$. That is, U consists of those vectors whose ith component equals the $(i + s)$th component, for $i \in [s]$, and

$\dim(U) = s$. We claim that U is an isotropic space of $\mathcal{A}$. To see this, for a given $u \in U$, we can write it as $\begin{bmatrix} v \\ v \end{bmatrix}$ where $v \in \mathbb{F}^s$. So for $A \in \mathcal{A}$ such that $A = \begin{bmatrix} \mathbf{0} & B \\ -B^t & \mathbf{0} \end{bmatrix}$, we have $\begin{bmatrix} v^t & v^t \end{bmatrix} \begin{bmatrix} \mathbf{0} & B \\ -B^t & \mathbf{0} \end{bmatrix} \begin{bmatrix} v \\ v \end{bmatrix} = -v^t B^t v + v^t B v = -v^t B v + v^t B v = 0$.

The proof of Theorem 7 basically suggests that isotropic spaces of the third type are not going to matter for the comparison with $\alpha(\mathcal{A})$. We now go into the proof.

Proof of Theorem 7. Let $r = \mathrm{ncrk}(\mathcal{B})$ and $d = \alpha(\mathcal{A})$.

We first show that $\alpha(\mathcal{A}) \geq n - \mathrm{ncrk}(\mathcal{B})$. Note that $r \leq \min\{s,t\}$, because we have the trivial isotropic pairs $(\mathbb{F}^s, \mathbf{0})$ and $(\mathbf{0}, \mathbb{F}^t)$. If $r = \min\{s,t\}$, note that $n - \min\{s,t\} = \max\{n-s, n-t\} = \max\{s,t\}$, and we do have isotropic spaces of dimensions s and t, respectively, by (1) from above. If $r < \min\{s,t\}$, then by (2) from above, there is an isotropic space of dimensions $n - r$. This shows that $\alpha(\mathcal{A}) \geq n - \mathrm{ncrk}(\mathcal{B})$.

We then show that $\alpha(\mathcal{A}) \leq n - \mathrm{ncrk}(\mathcal{B})$, or equivalently, $\mathrm{ncrk}(\mathcal{B}) \leq n - \alpha(\mathcal{A})$. Again, if $\alpha(\mathcal{A}) = \max\{s,t\}$, then $\mathrm{ncrk}(\mathcal{B}) \leq n - \max\{s,t\} = \min\{s,t\}$, which trivially holds. So we assume that $\alpha(\mathcal{A}) > \max\{s,t\}$. Let U be an isotropic space of dimension $d = \alpha(\mathcal{A})$, and take an $n \times d$ matrix whose columns form a basis of U, which, by abuse of notation, is also denoted by U. Let $U = \begin{bmatrix} v'_1 & v'_2 & \dots & v'_d \\ w'_1 & w'_2 & \dots & w'_d \end{bmatrix}$, where $v'_i \in \mathbb{F}^s$, and $w'_i \in \mathbb{F}^t$. Let $V' = \begin{bmatrix} v'_1 & v'_2 & \dots & v'_d \end{bmatrix}$, and $W' = \begin{bmatrix} w'_1 & w'_2 & \dots & w'_d \end{bmatrix}$. By doing a linear combination over the columns, we can assume that U is of the form $\begin{bmatrix} V & \mathbf{0} \\ W'' & W \end{bmatrix}$ where $V \in \mathrm{M}(s \times c, \mathbb{F})$, $W'' \in \mathrm{M}(t \times c, \mathbb{F})$, and $W \in \mathrm{M}(t \times e, \mathbb{F})$, such that $c + e = d$, $\mathrm{rk}(V) = c$, and $\mathrm{rk}(W) = e$. By abuse of notation, let V be the subspace of $\mathbb{F}^s$ spanned by columns in V, and W the subspace of $\mathbb{F}^t$ spanned by columns in W.

▷ Claim 21. Let V and W be as above. Then (V, W) form an isotropic pair for $\mathcal{B}$.

Proof. From above we have that $U = \begin{bmatrix} V & \mathbf{0} \\ W'' & W \end{bmatrix}$, and suppose $U = \begin{bmatrix} v_1 & v_2 & \dots & v_d \\ w_1 & w_2 & \dots & w_d \end{bmatrix}$. This means that $v_i = \mathbf{0}$ for $c < i \leq d$. Since U is an isotropic space of $\mathcal{A}$, we have, for any $i, j \in [d]$, and $A = \begin{bmatrix} \mathbf{0} & B \\ -B^t & \mathbf{0} \end{bmatrix} \in \mathcal{A}$, $\begin{bmatrix} v_i^t & w_i^t \end{bmatrix} \begin{bmatrix} \mathbf{0} & B \\ -B^t & \mathbf{0} \end{bmatrix} \begin{bmatrix} v_j \\ w_j \end{bmatrix} = -w_i^t B^t v_j + v_i^t B w_j = -v_j^t B w_i + v_i^t B w_j = 0$, so $v_i^t B w_j = v_j^t B w_i$. In particular, for any column vector $v \in V$ and any column vector $w \in W$, we have $v^t B w = \mathbf{0}^t B w'' = 0$ for some column vector $w'' \in W''$. This concludes the proof. ◁

We then get that $\mathrm{ncrk}(\mathcal{B}) \leq (s+t) - (c+e) = n - d = n - \alpha(\mathcal{A})$, and the proof is concluded. ◀

We now set out to prove Corollary 8. For this we need one more ingredient. In the literature [48, 61], the computation of the non-commutative ranks only deals with the case when $s = t$. To use that for general $\mathrm{M}(s \times t, \mathbb{F})$ we need a little twist.

▶ **Proposition 22.** *Over any field $\mathbb{F}$, computing the non-commutative rank of $\mathcal{B} \leq \mathrm{M}(s \times t, \mathbb{F})$ can be done in deterministic polynomial time.*

Proof. If $s = t$, this follows from [61]. Without loss of generality, let us assume then $s < t$. We shall construct some $\mathcal{C} \leq \mathrm{M}(t, \mathbb{F})$ as follows. First, for any $B \in \mathcal{B}$, let $B' = \begin{bmatrix} \mathbf{0} \\ B \end{bmatrix}$, where $\mathbf{0}$ is of size $(t-s) \times t$, so $B' \in \mathrm{M}(t, \mathbb{F})$. Second, recall that $E_{i,j}$ is the elementary matrix with the (i,j)th entry being 1, and the rest entries being 0. Then $\mathcal{C}$ is the matrix space spanned by all B' and $E_{i,j}$ with $1 \leq i \leq t - s$ and $1 \leq j \leq t$.

We claim that $\mathrm{ncrk}(\mathcal{B})+(t-s) = \mathrm{ncrk}(\mathcal{C})$. To see that $\mathrm{ncrk}(\mathcal{B})+(t-s) \geq \mathrm{ncrk}(\mathcal{C})$, let (U, V) be an isotropic pair of $\mathcal{B}$, where $U \leq \mathbb{F}^s$, $V \leq \mathbb{F}^t$, such that $\mathrm{ncrk}(\mathcal{B}) = s+t-\dim(U)-\dim(V)$. Let $U' \leq \mathbb{F}^t$ be the image of U under the embedding $\mathbb{F}^s$ to $\mathbb{F}^t$ by sending e_i to e_{i+t-s}. Clearly, (U', V) is an isotropic pair for $\mathcal{C}$, so $\mathrm{ncrk}(\mathcal{C}) \leq 2t - \dim(U') - \dim(V) = (t-s) + s + t - \dim(U) - \dim(V) = (t-s) + \mathrm{ncrk}(\mathcal{B})$.

To show that $\mathrm{ncrk}(\mathcal{B}) + (t-s) \leq \mathrm{ncrk}(\mathcal{C})$, let (U, V) be an isotropic pair of $\mathcal{C}$. If $\mathrm{ncrk}(\mathcal{C}) = t$, then the equality is trivial. So in the following we assume $\mathrm{ncrk}(\mathcal{C}) < t$. We claim that if $V \neq \mathbf{0}$, then U is a subspace of $\langle e_{t-s+1}, \ldots, e_t \rangle$. Suppose not, then U contains a vector $u = [u_1, \ldots, u_t]^t$ with some $u_i \neq 0$ for $1 \leq i \leq t-s$. Because $E_{i,j}$ is present in $\mathcal{C}$ for $1 \leq j \leq t$, for $v \in \mathbb{F}^n$ to satisfy that $u^t E_{i,j} v = 0$ for any $1 \leq j \leq t$, v has to be $\mathbf{0}$. This implies that V has to be $\mathbf{0}$. Therefore all isotropic pairs of $\mathcal{C}$, except a trivial one $(\mathbb{F}^t, \mathbf{0})$, are also isotropic pairs of $\mathcal{B}$. Therefore, if $\mathrm{ncrk}(\mathcal{C}) < t$, and (U, V) is an isotropic pair for $\mathrm{ncrk}(\mathcal{C})$ such that $\mathrm{ncrk}(\mathcal{C}) = 2t - \dim(U) - \dim(V) < t$, we have $V \neq \mathbf{0}$, so $U \leq \langle e_{t-s+1}, \ldots, e_t \rangle$. Let U' be the image of U under the projection from $\mathbb{F}^t$ to $\mathbb{F}^s$ by sending $(v_1, \ldots, v_t)^t$ to $(v_{t-s+1}, \ldots, v_t)^t$. We see then $\dim(U') = \dim(U)$, and (U', V) is an isotropic space for $\mathcal{B}$. From this we can conclude the proof. ◀

We are now ready to prove Corollary 8.

Proof of Corollary 8. Given $\mathcal{A} \leq \Lambda(n, \mathbb{F}_q)$, q odd, first put it into the explicit bipartite form using Theorem 5, which also produces the bases of the two subspaces in an isotropic 2-decomposition. Then for a matrix space $\mathcal{B} \leq \mathrm{M}(s \times t, \mathbb{F})$ induced from the bipartite structure, compute its non-commutative rank $r = \mathrm{ncrk}(\mathcal{B})$ using Proposition 22. The isotropic number of $\mathcal{A}$ is then $n - r$, by Theorem 7. ◀

▶ Remark 23. To obtain analogues of Corollary 8 over $\mathbb{R}$ and $\mathbb{C}$, the bottleneck is that over $\mathbb{R}$ and $\mathbb{C}$, Theorem 5 only outputs the projection matrix as the product of a sequence of matrices over different extension fields. This prevents us from working with the bases of the subspaces in an isotropic 2-decomposition directly. Of course, if we are content with approximating those algebraic numbers up to certain precision, we can use the representation of the projection as a product of matrices over different extension fields to get such, and then work with that.

7 Proof of Theorem 9

Let us first work with general $\mathbb{F}$, and then restrict to our target field $\mathbb{Q}$ at some point. Recall that the problem is to decide whether an alternating matrix space $\mathcal{A} \leq \Lambda(n, \mathbb{F})$ has an isotropic space of dimension 2. We first make the following easy observation.

▶ **Observation 24.** *Let $\mathcal{A} = \langle A_1, \ldots, A_m \rangle \leq \Lambda(n, \mathbb{F})$. Then $\mathcal{A}$ has an isotropic space of dimension 2, if and only if there exist linearly independent $v, w \in \mathbb{F}^n$ such that for any $A \in \mathcal{A}$, $v^t A w = 0$, which is further equivalent to that for any $i \in [m]$, $v^t A_i w = 0$.*

We now need to prove some auxiliary results. Let $\mathcal{B} = \langle B_1, \ldots, B_m \rangle \leq \mathrm{M}(n, \mathbb{F})$, and let $\mathbf{B} = (B_1, \ldots, B_m) \in \mathrm{M}(n, \mathbb{F})^m$. Here is a natural problem about matrix spaces.

▶ **Problem 25.** The existential singularity problem for matrix spaces, or the linear $\exists$-singularity problem, asks the following: given $\mathcal{B} \in \mathrm{M}(n, \mathbb{F})$, decide whether there exists a singular (e.g. non-full-rank) *non-zero* matrix in $\mathcal{B}$.

The linear ∃-singularity problem turns out to be quite interesting. We discuss this problem in detail in Section 7.1. For the sake of proving Theorem 9, we need the following result, whose proof can be found there.

▶ **Lemma 26.** *Over $\mathbb{Q}$, assuming the generalized Riemann hypothesis, there is a randomized polynomial-time reduction from deciding quadratic residuosity modulo squarefree composite numbers to the linear ∃-singularity problem.*

We then show that the linear ∃-singularity problem reduces to deciding whether an alternating matrix space has an isotropic space of dimension 2. This reduction works over any field.

For this purpose, it will be convenient to define an intermediate problem, which may be viewed as just a reformulation of Problem 25.

Recall that $\mathcal{B} = \langle B_1, \ldots, B_m \rangle \leq \mathrm{M}(n, \mathbb{F})$, and $\mathbf{B} = (B_1, \ldots, B_m) \in \mathrm{M}(n, \mathbb{F})^m$. Think of $\mathbf{B}$ as a 3-tensor $T_{\mathbf{B}}$ of size $n \times n \times m$, such that $T_{\mathbf{B}}(i, j, k) = B_k(i, j)$. That is, B_k's are the slices according to the third index (lateral slices). We will also be interested in the matrices obtained according to the first index (horizontal slices) and the second index (vertical slices). Specifically, define $\mathbf{B}^v$ be the n-tuple of $n \times m$ matrices that are vertical slices of $T_{\mathbf{B}}$. That is, $\mathbf{B}' = (B'_1, \ldots, B'_n) \in \mathrm{M}(n \times m, \mathbb{F})^n$, where $B'_j = [B_1 e_j, \ldots, B_m e_j]$, or in other words, $B'_j(i, k) = T_{\mathbf{B}}(i, j, k) = B_k(i, j)$. Similarly we can define the matrix tuple consisting of the horizontal slices of $T_{\mathbf{B}}$.

We now consider the matrix space $\mathbf{B}' \in \mathrm{M}(n \times m, \mathbb{F})^n$. For $v = (v_1, \ldots, v_m)^t \in \mathbb{F}^m$, its right degree in $\mathbf{B}'$ is defined to be the rank of $[B'_1 v, \ldots, B'_m v] = v_1 B_1 + \cdots + v_m B_m$. Therefore, every non-zero $v \in \mathbb{F}^n$ has right degree n in $\mathbf{B}'$, if and only if every matrix in $\mathcal{B}$ is of rank n. Lemma 26 then immediately implies the following.

▶ **Corollary 27.** *Over $\mathbb{Q}$, assuming the generalized Riemann hypothesis, there is a randomized polynomial-time reduction from deciding quadratic residuosity modulo squarefree composite numbers to deciding whether there exists a non-zero $v \in \mathbb{F}^m$ of right degree $< n$ w.r.t. a matrix tuple $\mathbf{B}' \in \mathrm{M}(n \times m, \mathbb{F})$.*

Given $\mathbf{B}' = (B'_1, \ldots, B'_n) \in \mathrm{M}(n \times m, \mathbb{F})$, we construct a tuple of alternating matrices of size $(n+m) \times (n+m)$, as follows. For $i \in [n]$, let $A_i = \begin{bmatrix} \mathbf{0} & B'_i \\ -B'^t_i & \mathbf{0} \end{bmatrix}$. For $1 \leq i < j \leq n$, let $C_{i,j} = e_i e_j^t - e_j e_i^t$. For $1 \leq k < \ell \leq m$, let $D_{k,\ell} = e_{n+k} e_{n+\ell}^t - e_{n+\ell} e_{n+k}^t$. Note that $C_{i,j}$ and $D_{k,\ell}$ are elementary alternating matrices. Let $\mathbf{A} = (A_1, \ldots, A_n, C_{1,2}, \ldots, C_{n-1,n}, D_{1,2}, \ldots, D_{m-1,m})$. We now claim the following.

▷ Claim 28. Let $\mathbf{B}' \in \mathrm{M}(n \times m, \mathbb{F})^n$ and $\mathbf{A} \in \Lambda(n+m, \mathbb{F})^{n+\binom{n}{2}+\binom{m}{2}}$ be as above. Let $\mathcal{A} = \langle \mathbf{A} \rangle \leq \Lambda(n+m, \mathbb{F})$. Then there exists a non-zero $v' \in \mathbb{F}^m$ of right degree $< n$ in $\mathbf{B}'$ if and only if $\mathcal{A}$ has an isotropic space of dimension 2.

Proof. By Observation 24, to decide whether $\mathcal{A}$ has an isotropic space of dimension ≥ 2, we only need to test whether there exist linearly independent $u, v \in \mathbb{F}^{n+m}$, such that for any $E = A_i$, or $C_{i,j}$, or $D_{k,\ell}$, $u^t E v = 0$.

The only if direction is easy to verify. Suppose $v' \in \mathbb{F}^m$ is of right degree $n-1$ w.r.t. $\mathbf{B}'$, namely $[B'_1 v', \ldots, B'_m v']$ is of rank $< n$. Then take any non-zero $u' \in \mathbb{F}^n$ in the left kernel of $[B'_1 v', \ldots, B'_m v']$, and we have that for $i \in [m]$, $u'^t B'_i v' = 0$. Now construct $u = \begin{bmatrix} u' \\ \mathbf{0} \end{bmatrix} \in \mathbb{F}^{n+m}$, and $v = \begin{bmatrix} \mathbf{0} \\ v' \end{bmatrix} \in \mathbb{F}^{n+m}$. For $i \in [n+m]$, let the ith component of u (resp. v) be u_i (resp.

v_i). Then for $i \in \{n+1, \ldots, n+m\}$, $u_i = 0$. For $i \in [n]$, $v_i = 0$. Clearly, u and v are linearly independent. Furthermore, it is easy to verify that (1) $u^t A_i v = u'^t B'_i v' = 0$, (2) $u^t C_{i,j} v = u_i v_j - u_j v_i = u_i \cdot 0 - u_j \cdot 0 = 0$, as $i, j \in [n]$, and similarly (3) $u^t D_{k,\ell} v = 0$. Then u and v spans a dimension-2 isotropic space of $\mathcal{A}$.

For the if direction, suppose u and v are linearly independent vectors in $\mathbb{F}^{n+m}$, and satisfy that for any $E = A_i$, or $C_{i,j}$, or $D_{k,\ell}$, $u^t E v = 0$. Write $u = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$, where $u_1 \in \mathbb{F}^n$ and $u_2 \in \mathbb{F}^m$. Similarly write $v = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$, where $v_1 \in \mathbb{F}^n$ and $v_2 \in \mathbb{F}^m$. By $u^t C_{i,j} v = 0$, we have that u_1 and v_1 are linearly dependent. By $u^t D_{k,\ell} v = 0$, we have that u_2 and v_2 are linearly dependent. We first observe that it cannot be the case that $u_1 = v_1 = \mathbf{0}$, nor $u_2 = v_2 = \mathbf{0}$. As otherwise, say if $u_1 = v_1 = \mathbf{0}$, then because u_2 and v_2 are linearly dependent, we have u and v are linearly dependent, a contraction. Therefore, without loss of generality, we assume that $u_1 \neq \mathbf{0}$, so $v_1 = \alpha_1 u_1$ for some $\alpha_1 \in \mathbb{F}$. We then have two cases. In the first case, if $u_2 = \mathbf{0}$, then $v_2 \neq \mathbf{0}$, and $v' = v - \alpha_1 u$ and u are linearly independent. In the second case, if $u_2 \neq \mathbf{0}$, then $v_2 \neq \alpha_1 u_2$, as otherwise u and v would be linearly dependent. Then again, letting $v' = v - \alpha_1 u$, we have u and v' are linearly independent. Clearly, u and v' still satisfy that for any $E = A_i$, or $C_{i,j}$, or $D_{k,\ell}$, $u^t E v' = 0$. Write v' as $\begin{bmatrix} v'_1 \\ v'_2 \end{bmatrix}$ where $v'_1 \in \mathbb{F}^n$, and $v'_2 \in \mathbb{F}^m$, so $v'_1 = \mathbf{0}$, and $v'_2 \neq \mathbf{0}$. We then have $u_2 = \alpha_2 v'_2$. Letting $u' = u - \alpha_2 v'$, we have u' and v' are linearly independent, and for any $E = A_i$, or $C_{i,j}$, or $D_{k,\ell}$, $u'^t E v' = 0$. Write u' as $\begin{bmatrix} u'_1 \\ u'_2 \end{bmatrix}$ where $u'_1 \in \mathbb{F}^n$, and $u'_2 \in \mathbb{F}^m$, so $u'_1 \neq \mathbf{0}$, and $u'_2 = \mathbf{0}$. It is then straightforward to verify that the condition $u'^t A_i v' = 0$ is equivalent to $u'^t_1 B'_i v'_2 \neq 0$. Recall that neither u'_1 nor v'_2 is the zero vector; this just translates to say that v'_2 is of right degree $< n$ w.r.t. $\mathbf{B}'$. ◁

Theorem 9 follows by combining Claim 28 and Corollary 27.

7.1 The existential singularity problem for matrix spaces

In this subsection, we discuss on Problem 25, which we believe is a very interesting problem in its own right. We therefore examine this problem over various fields, and prove Lemma 26 over $\mathbb{Q}$.

The affine version of Problem 25 has been studied in [24]. More specifically, the $\exists$-singularity problem for affine matrix spaces asks to decide whether an affine matrix space contains a non-full-rank matrix (not necessarily nonzero). In [24], this problem was called the singularity problem. This may cause some confusion, because in [48, 61] the singularity problem for matrix spaces is to decide whether all matrices in a matrix space are singular. For clarification, we then call the problem in [24] the affine $\exists$-singularity problem, and Problem 25 the linear $\exists$-singularity problem.

In [24], it was shown that the affine $\exists$-singularity problem is NP-hard over $\mathbb{F}_q$, $\mathbb{Q}$, or $\mathbb{R}$. We first note that the linear $\exists$-singularity problem reduce to that for affine matrix spaces, but the inverse direction is not clear.[9] Furthermore, the proof strategy of [24] cannot be adapted directly to tackle Problem 25, because of the introduction of field constants in the reduction. In particular, the use of field constants in the transformation from algebraic branching programs to symbolic determinants by Valiant [106] seems particularly crucial. Indeed, as we will see below, the $\exists$-singular problems for matrix spaces and for affine matrix spaces demonstrate quite different behaviors.

[9] To reduce the matrix space case to the affine case is easy: if $\mathcal{B} = \langle B_1, \ldots, B_m \rangle$, then form m affine spaces $B_i + \langle B_1, \ldots, B_{i-1}, B_{i+1}, \ldots, B_m \rangle$.

Matrix spaces in which every nonzero matrix is of full-rank has been studied in mathematics for a long time. More broadly, if $\mathcal{B} \leq \mathrm{M}(n, \mathbb{F})$ satisfies that every nonzero matrix in $\mathcal{B}$ is of a fixed rank r, we say that $\mathcal{B}$ satisfies the fixed rank r condition. Such matrix spaces are of interests in algebraic geometry (mostly when over algebraically closed fields), differential topology (mostly when over $\mathbb{R}$), number theory (mostly when over $\mathbb{Q}$), and algebra (mostly when over finite fields). It turns out that several results from these different branches of mathematics will be useful for our algorithmic purposes as well.

To start with, the following quantity has been studied extensively in the literature. Let $\rho(n, r, \mathbb{F})$ be the maximum dimension over those $\mathcal{B} \leq \mathrm{M}(n, \mathbb{F})$ satisfying the fixed rank r condition. Also let $\tau(n, r, \mathbb{F})$ be the maximum dimension over those affine matrix spaces $\mathcal{C} \subseteq \mathrm{M}(n, \mathbb{F})$ satisfying the fixed rank r condition. Also let $\rho(n, \mathbb{F}) := \rho(n, n, \mathbb{F})$, and $\tau(n, \mathbb{F}) := \tau(n, n, \mathbb{F})$. As pointed out in [37], $\rho(n, \mathbb{F}) \leq n$, and $\tau(n, \mathbb{F}) = \binom{n}{2}$. Two remarks are due here. First, $\rho(n, \mathbb{F})$ can be much smaller than n for certain fields; see below. Second, the $\binom{n}{2}$ bound for $\tau(n, \mathbb{F})$ can be easily achieved at $I_n + \mathcal{U}$ where $\mathcal{U}$ is the linear space of strictly upper triangular matrices. This distinction already suggests that the difference between the linear and the affine cases can be significant.

In this following, we discuss on $\mathbb{C}$, $\mathbb{R}$, and $\mathbb{Q}$, comparing the linear and affine settings, and presenting some algorithms for the linear case, including a proof of Lemma 26. We refer the interested reader to [98] for the finite field case.

Over $\mathbb{C}$

The affine $\exists$-singularity problem over $\mathbb{C}$ is only known to be in RP [24].

We then turn to the linear $\exists$-singularity problem over $\mathbb{C}$. Sylvester showed that $\rho(n, \mathbb{C}) \leq 1$ [101], and Westwick generalized that to $\rho(n, r, \mathbb{C}) \leq 2n - 2r + 1$ [110]. Some subsequent developments include [19, 57].

Sylvester's result immediately translates to a deterministic efficient algorithm for the linear $\exists$-singularity problem over $\mathbb{C}$: if the input matrix space $\mathcal{B} \leq \mathrm{M}(n, \mathbb{C})$ is of dimension ≥ 2, then return "exists." Otherwise, $\mathcal{B} = \langle B \rangle$, and return "exists" if and only if B is of full-rank.

Over $\mathbb{R}$

The affine $\exists$-singularity problem over $\mathbb{R}$ is NP-hard [24].

We then turn to the linear $\exists$-singularity problem over $\mathbb{R}$. Based on the Radon-Hurwitz construction and Adams' vector field theorem [1], $\rho(n, \mathbb{R})$ is equal to the so-called Hurwitz-Radon function (see [2]). For $n \in \mathbb{N}$, write n in the form of $2^{4a+b} \cdot (2c+1)$ where $b \in \{0, 1, 2, 3\}$, and the Hurwitz-Radon function is $HR(n) = 8a + 2^b$. The significance of this result for our algorithmic purpose is that $HR(n) \leq 2(\log n + 4)$. Some subsequent developments include [30, 70, 82].

Therefore, the linear $\exists$-singularity problem over $\mathbb{R}$ admits the following quasipolynomial-time algorithm. If the input matrix space $\mathcal{B} \leq \mathrm{M}(n, \mathbb{C})$ is of dimension $\geq HR(n)$, then return "exists." Otherwise, $\mathcal{B} = \langle B_1, \ldots, B_m \rangle$ where $m \leq HR(n) \leq 2(\log n + 4)$. We form m affine spaces, $\mathcal{C}_i := B_i + \langle B_1, \ldots, B_{i-1}, B_{i+1}, \ldots, B_m \rangle$, for every $i \in [m]$. The question then becomes whether any of the $\mathcal{C}_i$'s contains a singular matrix. This can be done by computing $f_i := \det(B_1 x_1 + \cdots + B_{i-1} x_{i-1} + B_i + B_{i+1} x_{i+1} + \cdots + B_m x_m)$ explicitly. Since the polynomial f_i, involving $O(\log n)$ variables, is of degree n, f_i has $n^{O(\log n)}$ monomials,

and we can fully write out f_i in time polynomial in $n^{O(\log n)}$.[10] After that, we can use the existential theory of reals [27, 91] to determine whether f_i has a non-trivial zero in time $n^{O(\log n)}$. Return "exists" if and only if one of these f_i's is solvable. This concludes the proof.

Over $\mathbb{Q}$

The affine $\exists$-singularity problem over $\mathbb{Q}$ is NP-hard [24].

We then turn to the linear $\exists$-singularity problem over $\mathbb{Q}$. To start with, observe that $\rho(n, \mathbb{Q}) \geq n$. This is because we can take a degree-n extension field $\mathbb{K}$ over $\mathbb{Q}$, and use the regular representation of $\mathbb{K}$. We now prove Lemma 26, which suggests that the linear $\exists$-singularity problem is not so easy either.

Proof of Lemma 26. We consider a special case of Problem 25 as follows. Assume $\mathcal{B} \leq \mathrm{M}(n, \mathbb{Q})$ is closed under matrix multiplication, so $\mathcal{B}$ forms an algebra over $\mathbb{Q}$. In this setting, Problem 25 just asks whether $\mathcal{B}$ is not a division algebra. We can even specialize further by considering $\mathcal{B}$ being a central simple algebra over $\mathbb{Q}$.

In [92], Rónyai considered the problem of testing whether a central simple algebra over $\mathbb{Q}$ of dimension 4 is isomorphic to $\mathrm{M}(2, \mathbb{Q})$. He showed that assuming the generalized Riemann hypothesis, there is a randomized efficient reduction from deciding quadratic residuosity modulo squarefree composite numbers to this problem. In [92], the algebras are represented by structural constants, but these can be turned into matrix representations in $\mathrm{M}(4, \mathbb{Q})$ (see e.g. [62]). It follows that there is an analogous reduction for matrix algebras.

We can then conclude the proof, because such an algebra is either isomorphic to $\mathrm{M}(2, \mathbb{Q})$ (in which there exists a nonzero singular matrix) or a division algebra (in which every nonzero matrix is full-rank). ◀

8 Proof of Theorem 10

8.1 Some basic statistics

All results in this subsection are either classical or straightforward. We collect them here, and provide proofs, partly for completeness, and partly because we will use some of the arguments here in the following.

We first recall the following bound on the number of subspaces of $\mathbb{F}_q^n$.

▶ **Fact 29.**

1. *For $d \leq \mathbb{N}$, $0 \leq d \leq n$, the number of dimension-d subspaces of $\mathbb{F}_q^n$ is equal to the Gaussian binomial coefficient*

$$\binom{n}{d}_q := \frac{(q^n - 1) \cdot (q^n - q) \cdot \ldots \cdot (q^n - q^{d-1})}{(q^d - 1) \cdot (q^d - q) \cdot \ldots \cdot (q^d - q^{d-1})}.$$

2. *The Gaussian binomial coefficient satisfies:*

$$q^{(n-d)d} \leq \binom{n}{d}_q \leq q^{(n-d)d+d}.$$

3. *The number of subspaces of $\mathbb{F}_q^n$ is $q^{\frac{1}{4}n^2+\Theta(n)}$.*

[10] There are several ways of doing this. One approach is to transform the determinant expression into an arithmetic circuit, and then compute along this circuit to get the final polynomial.

Proof. (1) is well-known. For (2), it is enough to verify that for any prime power q, and $n, d, k \in \mathbb{N}$, $n \geq d > k$, we have

$$q^{n-d} \leq \frac{q^n - q^k}{q^d - q^k} \leq q^{n-d+1}.$$

For (3), it is well-known that $\binom{n}{d}_q$ achieves maximal over d at $d = \lfloor n/2 \rfloor$. So we have

$$q^{\frac{1}{4}n^2 - \frac{1}{4}} \leq \binom{n}{\lfloor n/2 \rfloor}_q \leq \sum_{d=0}^{n} \binom{n}{d}_q \leq (n+1) \cdot \binom{n}{\lfloor n/2 \rfloor}_q \leq q^{\frac{1}{4}n^2 + \lfloor n/2 \rfloor + \log(n+1)},$$

from which the result follows. ◀

Analogously, we consider the number of isotropic spaces of a non-degenerate alternating form $A \in \Lambda(n, q)$.

▶ **Proposition 30.** *Let $A \in \Lambda(n, q)$, n even, be a non-degenerate alternating matrix. Then we have the following.*

1. *For $d \in \mathbb{N}$, $0 \leq d \leq n/2$, the number of dimension-d isotropic spaces of A is*

$$I(A, d) := \frac{(q^n - 1) \cdot (q^{n-1} - q) \cdot \ldots \cdot (q^{n-(d-1)} - q^{d-1})}{(q^d - 1) \cdot (q^d - q) \cdot \ldots \cdot (q^d - q^{d-1})}.$$

   For $d \in \mathbb{N}$, $d > n/2$, there are no dimension-d isotropic spaces.
2. *For $d \in \mathbb{N}$, $0 \leq d \leq n/2$, $I(A, d)$ is bounded as follows:*

$$q^{nd - \frac{3}{2}d^2 + \frac{1}{2}d} \leq I(A, d) \leq q^{nd - \frac{3}{2}d^2 + \frac{3}{2}d}.$$

3. *The number of isotropic spaces of A is $q^{\frac{1}{6}n^2 + \Theta(n)}$.*
4. *The number of maximal isotropic spaces of A is $q^{\frac{1}{8}n^2 + \Theta(n)}$.*

Proof. For (1), suppose we have chosen $u_1, \ldots, u_i$ such that $\langle u_1, \ldots, u_i \rangle$ is an isotropic space. We then need to select the next eligible u_{i+1}, such that $\langle u_1, \ldots, u_{i+1} \rangle$ forms an isotropic space. Since u_{i+1} needs to satisfy $u_{i+1}^t A u_j = 0$ for $1 \leq j \leq i$, and A is non-degenerate, u_{i+1} should be from a dimension-$(n-i)$ subspace, namely the subspace orthogonal to Au_j, $1 \leq j \leq i$. Furthermore, u_{i+1} is not in $\langle u_1, \ldots, u_i \rangle$. So there are $q^{n-i} - q^i$ choices of u_{i+1} in the ith step. This explains the numerator. The denominator is of such form, because for each isotropic space there are these many ordered bases.

For (2), it follows from the same argument as the proof for Fact 29 (2).

For (3), note that $nd - \frac{3}{2}d^2$ achieves its maximum at $d = \frac{1}{3}n$.

For (4), note that maximal isotropic spaces are of dimension $n/2$. This is because for any isotropic space U of dimension $d < n/2$, we can choose an eligible u_{d+1} as in the proof for (1), such that $\langle U, u_{d+1} \rangle$ is also an isotropic space. ◀

▶ **Proposition 31.** *Let $A \in \Lambda(n, q)$, n even, be a non-degenerate alternating matrix. Then all isotropic spaces of A can be enumerated in time $q^{\frac{1}{6}n^2 + O(n)}$.*

Proof. We enumerate isotropic spaces according to dimensions in an increasing order. Each subspace of $\mathbb{F}_q^n$ is represented by an ordered basis. We will maintain a list L of all isotropic subspaces, and for each isotropic space U of dimension d, maintain a list of isotropic spaces of dimension $d+1$ that contain U, denoted as $L(U)$. Note that for a fixed U, there are at most q^{n-d} such spaces. In other words, we will record the lattice of isotropic spaces.

Suppose we have enumerated all isotropic spaces of dimensions $\leq d$. To enumerate isotropic spaces of dimension $d+1$, we maintain a list of such spaces. Then for each isotropic space U of dimension d, and for each $u \in \mathrm{rad}(U) \setminus U$, we form $U' = \langle u, U \rangle$, and test whether U' is in $L(U)$. If not, then we add U' to L. We also add it to $L(U)$, and for every dimension d-subspace $\tilde{U}$ of U', add U' to $L(\tilde{U})$. Otherwise we move on.

Clearly, in the above procedure, each isotropic space will added, and only added to L once. This procedure runs in time $N \cdot q^{O(n)}$, where N denotes the number of isotropic spaces of A. We can then conclude by resorting to Proposition 30 (3). ◀

When working with maximal isotropic spaces, it is enough to restrict our attention to just non-degenerate matrix spaces.

▶ **Observation 32.** *For $\mathcal{A} \leq \Lambda(n, \mathbb{F})$, any maximal isotropic space of $\mathcal{A}$ contains* $\mathrm{rad}(\mathcal{A})$.

8.2 A non-trivial upper bound on the number of maximal isotropic spaces

For $\mathcal{A} \leq \Lambda(n, q)$, let $\mathrm{MI}(\mathcal{A})$ be the set of maximal isotropic spaces of $\mathcal{A}$, and $\mathrm{NMI}(\mathcal{A})$ be the number of maximal isotropic spaces of $\mathcal{A}$, e.g. the size of $\mathrm{MI}(\mathcal{A})$. Let $\mathrm{MaxNMI}(n, q)$ be the maximum of $\mathrm{NMI}(\mathcal{A})$ over all $\mathcal{A} \leq \Lambda(n, q)$. By Fact 29 (3) and Proposition 30 (4),

$$q^{\frac{1}{4}n^2+O(n)} \geq \mathrm{MaxNMI}(n, q) \geq q^{\frac{1}{8}n^2+\Omega(n)}.$$

Theorem 10, slightly reformulated

Let $\mathrm{MaxNMI}(n, q)$ be as above. Then $\mathrm{MaxNMI}(n, q) \leq q^{\frac{1}{6}n^2+C\cdot n}$ for some large enough absolute constant C.

Let us illustrate the proof strategy for Theorem 10, before we enter the details.

The starting point of our proof is the alternative proof bounding the number of maximal independent sets by Wood [115].

The core of Wood's argument is the following. Let $G = (V, E)$ be a graph on n vertices. Recall that we want to prove that the number of maximal independent sets in G is no more than $g(n) = 3^{\frac{n}{3}}$. We shall do an induction on n. Let $v \in V$ be a vertex of minimal degree d. Let $N(v)$ be the set of neighbours of v together with v (e.g. the closed neighbourhood of v). Then any maximal independent set I contains some $w \in N(v)$, as otherwise $I \cup \{v\}$ would be a larger independent set. If I contains $w \in N(v)$, then $I \setminus \{w\}$ would be a maximal independent set of $G|_{V \setminus N(w)}$, the induced subgraph of G on $V \setminus N(w)$. Since $G|_{V \setminus N(w)}$ is of size $\leq n - d - 1$, we then have

$$g(n) \leq (d+1) \cdot g(n-d-1). \tag{3}$$

From this relation and the induction hypothesis, the result follows in a rather straightforward fashion.

In the following, we will develop a linear algebraic analogue of Equation 3. However, just applying this does not suffice, when there are many vectors of degree 1.

We remedy this by showing that in this setting, the maximum rank is large, which allows us to use an argument similar to one in Proposition 30. More specifically, recall that in Proposition 30 (3), we showed that the number of isotropic spaces of a non-degenerate alternating matrix is bounded from above by $q^{\frac{1}{6}n^2+O(n)}$. Note that any maximal isotropic space of $\mathcal{A}$ is an isotropic space of any $A \in \mathcal{A}$. So if $\mathcal{A}$ contains a non-degenerate A, we

can immediately obtain Theorem 10 in this case. However, there are non-degenerate matrix spaces that do not contain non-degenerate alternating matrices. For example, the following is an alternating matrix space of maximum rank 2, written in a parametrized form:

$$A = \begin{bmatrix} 0 & x_1 & \dots & x_n \\ -x_1 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ -x_n & 0 & \dots & 0 \end{bmatrix}.$$

For our purpose, we will need to bound the number of isotropic spaces for matrix spaces of rank $> \frac{2}{3}n$. So the following lemma is required; its proof is postponed to Section 8.2.1.

▶ **Lemma 33.** *Let $A \in \Lambda(n, q)$ be of rank $> \frac{2}{3}n$. Then the number of isotropic spaces of A is bounded from above by $q^{\frac{1}{6}n^2+Dn}$ for some large enough absolute constant D.*

We are now ready to prove Theorem 10.

Proof of Theorem 10. Let $\mathcal{A} \leq \Lambda(n, q)$. By Observation 32, for our purpose, we can assume that $\mathcal{A}$ is non-degenerate. Let $g_q(n) = q^{\frac{1}{6}n^2+Cn}$. We prove by an induction on n. Assume $\mathrm{MaxNMI}(\ell, q) \leq g_q(\ell)$ holds for any $\ell < n$. Our goal is to show that $\mathrm{NMI}(\mathcal{A}) \leq g_q(n)$.

Let $d = \min\{\deg_{\mathcal{A}}(v) : v \in \mathbb{F}_q^n, v \neq \mathbf{0}\}$. As $\mathcal{A}$ is non-degenerate, $d \geq 1$. Take any $v \in \mathbb{F}_q^n$ of degree d, and let $c = n - d$ be the codegree of v. Let $N(v) := (\mathbb{F}_q^n \setminus \mathrm{rad}_{\mathcal{A}}(v)) \cup \{v\} = \{u \in \mathbb{F}_q^n : \exists A \in \mathcal{A}, u^t Av \neq 0\} \cup \{v\}$. We call $N(v)$ the closed neighbourhood of v. Note that $|N(v)| = q^n - q^c + 1$.

Let $U \leq \mathbb{F}_q^n$ be a maximal isotropic space of $\mathcal{A}$. Clearly, $U \cap N(v) \neq \emptyset$. As otherwise, we have $U \subseteq \mathrm{rad}(v)$ and $v \notin U$. This is equivalent to that $v \in \mathrm{rad}(U)$ and $v \notin U$. It follows that $U \subsetneq \mathrm{rad}(U)$, so by Observation 17, U is not maximal, a contradiction.

Therefore there exists some $w \in N(v) \cap U$. It follows that $U \subseteq \mathrm{rad}(w)$. Since U is maximal isotropic in $\mathcal{A}$, U is also a maximal isotropic space of $\mathcal{A}|_{\mathrm{rad}(w)}$. As $\deg(w) \geq \deg(v) = d$, $\dim(\mathrm{rad}(w)) \leq c$. Furthermore, note w is an isolated vector in $\mathcal{A}|_{\mathrm{rad}(w)}$. We then have

$$\begin{aligned} \mathrm{NMI}(\mathcal{A}) &\leq \sum_{w \in N(v)} \mathrm{NMI}(\mathcal{A}|_{\mathrm{rad}(w)}) && (4) \\ &\leq (q^n - q^c + 1) \cdot g_q(c-1), && (5) \end{aligned}$$

where the second inequality is due to the induction hypothesis. Note that on the right hand side, we have $g_q(c-1)$ instead of $g_q(c)$, because w is an isolated vector in $\mathcal{A}|_{\mathrm{rad}(w)}$, and Observation 32. The reader may want to compare this with Equation 3.

Now suppose $d \geq 2$, that is, $c \leq n - 2$. We then have

$$\begin{aligned} \mathrm{NMI}(\mathcal{A}) &\leq q^n \cdot g_q(n-3) \\ &\leq q^n \cdot q^{\frac{1}{6}(n-3)^2+C(n-3)} \\ &= q^{\frac{1}{6}n^2+Cn+(\frac{3}{2}-3C)} \\ &\leq q^{\frac{1}{6}n^2+Cn}. \end{aligned}$$

Note that the second inequality is by the induction hypothesis, and the last inequality holds as long as $C \geq 1$.

Now suppose $d = 1$. In this case, Equation 5 is not enough for our purpose. We then need the following refinement. Partition $N(v)$ as $N_1(v) \cup N_{\geq 2}(v)$, where $N_1(v) = \{w \in \mathbb{F}_q^n : w \in N(v), \deg(w) = 1\}$, and $N_{\geq 2}(v) = N(v) \setminus N_1(v)$. A refinement of Equation 4 gives that

$$\mathrm{NMI}(\mathcal{A}) \leq |N_1(v)| \cdot g_q(n-2) + |N_{\geq 2}(v)| \cdot g_q(n-3). \quad (6)$$

If $|N_1(v)| \leq q^{\frac{2}{3}n}$, then we have

$$\begin{aligned}\mathrm{NMI}(\mathcal{A}) &\leq q^{\frac{2}{3}n} \cdot g_q(n-2) + q^n \cdot g_q(n-3)\\ &\leq q^{\frac{2}{3}n} \cdot q^{\frac{1}{6}(n-2)^2+C(n-2)} + q^n \cdot q^{\frac{1}{6}(n-3)^2+C(n-3)}\\ &\leq q^{\frac{1}{6}n^2+Cn+(\frac{2}{3}-2C)} + q^{\frac{1}{6}n^2+Cn+(\frac{3}{2}-3C)}\\ &\leq q^{\frac{1}{6}n^2+Cn-1} + q^{\frac{1}{6}n^2+Cn-1}\\ &\leq q^{\frac{1}{6}n^2+Cn}.\end{aligned}$$

Note that the second inequality is by the induction hypothesis, the second to the last inequality holds as long as $C \geq 1$.

If $|N_1(v)| > q^{\frac{2}{3}n}$, then we first prove the following.

▷ Claim 34. We have $\mathrm{rk}(\mathcal{A}) > \frac{2}{3}n$.

Proof for Claim 34. Suppose $\dim(\mathcal{A}) = m$. Let $s = \lfloor \frac{2}{3}n \rfloor + 1$, the smallest integer larger than $\frac{2}{3}n$. We will show that there exists a linear basis of some $\tilde{\mathcal{A}}$ that is isometric to $\mathcal{A}$, $\tilde{A}_1, \ldots, \tilde{A}_m \in \Lambda(n, q)$, such that

$$\tilde{A}_1 = \begin{bmatrix} B_1 & -C_1^t \\ C_1 & D_1 \end{bmatrix}, \tilde{A}_2 = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & D_2 \end{bmatrix}, \ldots, \tilde{A}_m = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & D_m \end{bmatrix}, \tag{7}$$

where $B_1 \in \Lambda(s, q)$, $C_1 \in M((n-s) \times s, q)$, and $D_i \in \Lambda(n-s, q)$. From this linear basis, it is clear that $\begin{bmatrix} B_1 \\ C_1 \end{bmatrix}$ is of rank $s > \frac{2}{3}n$, as otherwise $\tilde{\mathcal{A}}$ would be degenerate. It would follow then that $\mathrm{rk}(\mathcal{A}) = \mathrm{rk}(\tilde{\mathcal{A}}) > \frac{2}{3}n$.

We first start with an arbitrary linear basis of $\mathcal{A}$, say $A_1, \ldots, A_m \in \Lambda(n, q)$. Recall that v is of degree 1, and $|N_1(v)| > q^{\frac{2}{3}n}$. For later convenience, rename v as u_1. Then there exist $u_2, u_3, \ldots, u_s \in \mathbb{F}_q^n$, such that for $i \geq 2$, $u_i \in N_1(v)$, and $u_1, \ldots, u_s$ are linearly independent. As otherwise, suppose the maximum number of linearly independent u_i's from $N_1(v)$ we can find is $t < s$. Then since $N_1(v) > q^{\frac{2}{3}n} \geq q^t = |\langle u_1, \ldots, u_t \rangle|$, we can find $u_{t+1} \in N_1(v) \setminus \langle u_1, \ldots, u_t \rangle$, a contradiction.

We then can arrange a change of basis matrix T whose first s columns are $u_1, \ldots, u_s$. Apply this change of basis matrix T (by $T^t \cdot T$) to $A_1, \ldots, A_m$ to get a linear basis $\bar{A}_1, \ldots, \bar{A}_m$ for $\bar{\mathcal{A}} = T^t \mathcal{A} T$. Recall that e_i denotes the ith standard basis vector. Since u_i's are of degree 1, for any $i \in [s]$, we have $\bar{\mathcal{A}}(e_i)$ is of dimension 1. For $2 \leq i \leq s$, since $u_i \in N(v)$, we have

$$e_i^t(\bar{A}_1, \ldots, \bar{A}_m)e_1 \neq (0, \ldots, 0). \tag{8}$$

Without loss of generality, assume $\bar{A}_1 e_1 \neq \mathbf{0}$. As $\bar{\mathcal{A}}(e_1)$ is of dimension 1, we have for any $2 \leq j \leq m$, $\bar{A}_j e_1 = \lambda_j \bar{A}_1 e_1$ for some $\lambda_j \in \mathbb{F}_q$. We claim that for any $2 \leq i \leq s$, the ith entry of $\bar{A}_1 e_1$, $\bar{A}_1 e_1(i) \neq 0$. If not, then for any $2 \leq j \leq m$, $\bar{A}_j e_1(i) = \lambda_j \bar{A}_1 e_1(i) = 0$. This is equivalent to say that $e_i^t(\bar{A}_1, \ldots, \bar{A}_m)e_1 = 0$, contradicting Equation 8.

As $\bar{A}_i$'s are alternating matrices, we have for any i, j, k, $(\bar{A}_i e_j)(k) = -(\bar{A}_i e_k)(j)$. It follows that for $2 \leq i \leq s$, $\bar{A}_1 e_i(1) = -\bar{A}_1 e_1(i) \neq 0$, and for $2 \leq j \leq m$, $\bar{A}_j e_i(1) = -\bar{A}_j e_1(i) = -\lambda_j \bar{A}_1 e_1(i) = \lambda_j \bar{A}_1 e_i(1)$. Since $\bar{\mathcal{A}}(e_i)$ is of dimension 1 for $2 \leq i \leq s$, we infer that for $2 \leq j \leq m$ and $2 \leq i \leq s$, $\bar{A}_j e_i = \lambda_j \bar{A}_1 e_i$. We then let $\tilde{A}_1 = \bar{A}_1$, and for $2 \leq j \leq m$, $\tilde{A}_j = \bar{A}_j - \lambda_j \bar{A}_1$. Clearly, $\tilde{A}_1, \ldots, \tilde{A}_m$ still form a basis of $\tilde{\mathcal{A}}$, and they are of the form in Equation 7. The claim then follows. ◁

Combining Claim 34 and Lemma 33, the proof is concluded. ◀

8.2.1 Proof of Lemma 33

Let c be the corank of A. We then have $c < \frac{1}{3}n$.

Let $(u_1, \ldots, u_d)$ be an ordered basis of an isotropic space U of A of dimension d. For $i \in [d]$, let $U_i = \langle u_1, \ldots, u_i \rangle$, and let $a_i = \dim(A(U_i))$. We also let $U_0 = \mathbf{0}$, and $a_0 = 0$. Note that $U = U_d$, and we also let $a = a_d$. We call such an isotropic space of (d, a) type. Note that $\dim(\langle U_i, \mathrm{rad}(A) \rangle) = \dim(U_i) + \dim(\mathrm{rad}(A)) - \dim(U_i \cap \mathrm{rad}(A)) = \dim(A(U_i)) + \dim(\mathrm{rad}(A)) = a_i + c$.

After fixing $u_1, \ldots, u_i$, a valid u_{i+1} can come from two sources.

1. If $u_{i+1} \notin \langle U_i, \mathrm{rad}(A) \rangle$, then since u_{i+1} needs to satisfy $u_{i+1}^t A u_j = 0$ for $j = 1, \ldots, i$, the number of choices of u_{i+1} is upper bounded by $q^{n-a_i} - q^i$.
2. If $u_{i+1} \in \langle U_i, \mathrm{rad}(A) \rangle$, then the number of choices of u_{i+1} is upper bounded by $q^{c+a_i} - q^i$.

So the following indices are important: for $i \in [a]$, let b_i be the smallest $j \in [d]$ such that $a_j = \dim(A(U_j)) = i$. We then have $0 < b_1 < b_2 < \cdots < b_a \leq d$. We also let $b_0 = 0$ and $b_{a+1} = d$. We call such an ordered basis of $(b_1, \ldots, b_a)$ type of an isotropic space of (d, a) type.

The number of possible types of an isotropic space is trivially upper bounded by n^2, and the number of possible types of ordered bases of isotropic spaces of type (d, a) is upper bounded by $\binom{d}{a} \leq 2^d \leq 2^n$. So by a multiplicative factor of $n^2 \cdot 2^n$, we can restrict to consider ordered basis $(u_1, \ldots, u_d)$ of a fixed type $\mathbf{b} = (b_1, \ldots, b_a)$. By the discussion above, if $j = b_i$, then the number of choices for u_j is upper bounded by $t_{d,a,\mathbf{b}}(j) := q^{n-(i-1)} - q^{j-1}$. If $b_i < j < b_{i+1}$, the number of choices for u_j is upper bounded by $t_{d,a,\mathbf{b}}(j) := q^{c+i} - q^{j-1}$. Recall that $(q^n - q^i)/(q^d - q^i) \leq q^{n-d+1}$, for any q and $i < d \leq n$. If $j = b_i$, we have

$$t_{d,a,\mathbf{b}}(j)/(q^d - q^{j-1}) \leq q \cdot q^{n-(i-1)-d}. \tag{9}$$

If $b_i < j < b_{i+1}$, we have

$$t_{d,a,\mathbf{b}}(j)/(q^d - q^{j-1}) \leq q \cdot q^{(c+i)-d} \leq q \cdot q^{(c+a)-d} \tag{10}$$

Each dimension-d subspace of $\mathbb{F}_q^n$ has $(q^d - 1)(q^d - q) \ldots (q^d - q^{d-1})$ ordered bases, and each ordered basis of an isotropic space of type (d, a) is of a particular type. The number of dimension-d isotropic spaces of type (d, a) can be upper bounded by

$$\begin{aligned}
& \sum_{\text{type } \mathbf{b}=(b_1,\ldots,b_a)} \frac{t_{d,a,\mathbf{b}}(1) \cdot \ldots \cdot t_{d,a,\mathbf{b}}(d)}{(q^d - 1) \cdot \ldots \cdot (q^d - q^{d-1})} \\
= & \sum_{\text{type } \mathbf{b}=(b_1,\ldots,b_a)} \frac{t_{d,a,\mathbf{b}}(1)}{q^d - 1} \cdot \ldots \cdot \frac{t_{d,a,\mathbf{b}}(j)}{q^d - q^{j-1}} \cdot \ldots \cdot \frac{t_{d,a,\mathbf{b}}(d)}{q^d - q^{d-1}} \\
\leq & \sum_{\text{type } \mathbf{b}=(b_1,\ldots,b_a)} q^d \cdot q^{na - \sum_{i=1}^{a}(i-1) - da} \cdot q^{(c+a)(d-a) - d(d-a)} \\
\leq & \; 2^n \cdot q^{na - a^2/2 + (c+a)(d-a) - d^2 + d + a/2}.
\end{aligned}$$

Let us explain the first inequality. The q^d term is because of the q terms on the right hand sides of Equations 9 and 10. The $q^{na - \sum_{i=1}^{a}(i-1) - da}$ is by collecting those terms from Equation 9, and the $q^{(c+a)(d-a) - d(d-a)}$ term is by collecting those terms from Equation 10.

It is then clear that we need to bound $f(n, d, a) = na - a^2/2 + (c + a)(d - a) - d^2$ for $1 \leq a \leq d \leq n$. After some arrangement, we have

$$f(n, d, a) = -\frac{3}{2}(a - \frac{1}{3}(n + d - c))^2 + \frac{1}{6}(n + d - c)^2 - d^2 + dc.$$

We then distinguish between two cases.

1. Case (i): when $\frac{1}{3}(n+d-c) \le d$ holds, namely $d \ge (n-c)/2$. Only in this case, a can be set to $\frac{1}{3}(n+d-c)$, and the maximum can be set to $g(n,d) := \frac{1}{6}(n+d-c)^2 - d^2 + dc$. After some arrangement, we have

$$g(n,d) = -\frac{5}{6}(d - \frac{1}{5}(n+2c))^2 + \frac{1}{30}(n+2c)^2 + \frac{1}{6}(n-c)^2.$$

Since $c < n/3$, we have $(n-c)/2 > (n+2c)/5$. Recall that $d \ge (n-c)/2$. So $g(n,d)$ achieves maximal at $d = (n-c)/2$. Plugging this in, the maximal value is

$$h(n) := g(n,(n-c)/2) = -\frac{3}{8}(c - \frac{1}{3}n)^2 + \frac{1}{6}n^2 < \frac{1}{6}n^2.$$

2. Case (ii): when $\frac{1}{3}(n+d-c) > d$ holds, namely $d < (n-c)/2$. In this case, $f(n,d,a)$ achieves the maximal value at $a = d$, and

$$f(n,d,d) = -\frac{3}{2}(d - \frac{1}{3}n)^2 + \frac{1}{6}n^2 \le \frac{1}{6}n^2,$$

where the inequality becomes an equality at $d = n/3$.

Since in both cases, the maximal value is no more than $\frac{1}{6}n^2$, we can then conclude the proof.

8.3 Turning Theorem 10 into an algorithm

The proof of Theorem 10 can be turned into an algorithm for enumerating all maximal isotropic spaces in time $q^{\frac{1}{6}n^2+O(n)}$. We briefly indicate some algorithmic issues for doing this. Firstly, note that in time $q^{O(n)}$, one can compute $\deg_{\mathcal{A}}(v)$ for all $v \in \mathbb{F}_q^n$. Secondly, the Equation 4 naturally suggests a recursive algorithm structure. In the cases when $d \ge 2$, or $d = 1$ and $|N_1(v)| \le q^{\frac{2}{3}n}$, this recursive structure readily gives the desired algorithm. If $d = 1$ and $|N_1(v)| > q^{\frac{2}{3}n}$, we need to make the proofs of Claim 34 and Lemma 33 constructive. Then for each isotropic space of some $A \in \Lambda(n,\mathbb{F})$, A of rank $> \frac{2}{3}n$, test whether it is maximal using Observation 17.

For Claim 34, note that the selection of u_i's from $N_1(v)$ can be done easily in a greedy way. Other steps are readily constructive. For Lemma 33, we use the same procedure as in Proposition 31, whose running time is bounded in time $q^{\frac{1}{6}n^2+O(n)}$ by Lemma 33.

We then have the following.

▶ **Corollary 35.** *Given $\mathcal{A} \le \Lambda(n,q)$, all maximal isotropic spaces can be enumerated in time $q^{\frac{1}{6}n^2+O(n)}$.*

9 Proof of Theorem 11

Review of Lawler's algorithm

We first review Lawler's dynamic programming idea for computing the chromatic number [72], and then adapt that idea to our problem.

Given a graph $G = (V,E)$, Lawler's algorithm for computing $\chi(G)$ goes as follows. The idea is to build a table storing $\chi(H)$ for every induced subgraph H of G. Note that this table is of size 2^n. To fill in this table, the starting point is the empty graph with chromatic number 0. Suppose we have computed the chromatic numbers of those induced subgraphs of size $< \ell$. Let $H = (U,F)$ be an induced subgraph of size ℓ. Then the chromatic number of H can be computed by the following formula:

$$\chi(H) = 1 + \min_{I \subseteq U}\{\chi(H[U \setminus I])\},$$

where I goes over all maximal independent sets of H, and $H[U \setminus I]$ is the induced subgraph of H restricting to vertex set $U \setminus I$. Since there are at most $3^{\ell/3}$ maximal independent sets of H and they can be enumerated in time $O(3^{\ell/3} \cdot n)$, the exponential part of the time complexity of this algorithm is $\sum_{\ell=0}^{n} \binom{n}{\ell} \cdot 3^{\ell/3} = (1 + \sqrt[3]{3})^n$.

Directly applying Lawler's idea to isotropic numbers

The above idea can be adapted to compute $\chi(\mathcal{A})$ for $\mathcal{A} \leq \Lambda(n, q)$ as follows. To start with, recall that in the above algorithm we used the following simple fact: if a graph G admits a vertex c-coloring, then there is a vertex c-coloring in which one part is a maximal independent set. We leave the reader to check that the analogue of this fact in the alternating matrix space setting also holds.

Given $\mathcal{A} \leq \Lambda(n, q)$, we also store a table storing $\chi(\mathcal{B})$ for every induced alternating matrix spaces $\mathcal{B}$ of $\mathcal{A}$. Note that this table is of size $q^{\frac{1}{4}n^2+O(n)}$. To fill in this table, the starting point is the zero space with isotropic decomposition number 0. Suppose we have computed the isotropic decomposition numbers of those induced alternating matrix spaces of dimension $< \ell$. Let $\mathcal{B} \leq \Lambda(\ell, q)$ be an induced alternating matrix space corresponding to $U \leq \mathbb{F}_q^n$ of dimension ℓ. Then the isotropic decomposition number of $\mathcal{B}$ can be computed by the following formula

$$\chi(\mathcal{B}) = 1 + \min_{V \leq U, W \leq U} \{\chi(\mathcal{B}|_W)\},$$

where V goes over all maximal isotropic spaces of $\mathcal{B}$, and W goes over all complement subspaces of V in U. Note that here we also need to enumerate all complements of V, while in the graph setting, the complement set is unique. Recall that by Theorem 10, there are at most $q^{\frac{1}{6}\ell^2+O(\ell)}$ maximal isotropic spaces of $\mathcal{B}$, and they can be enumerated in time $q^{\frac{1}{6}\ell^2+O(\ell)}$. This gives a bound on the number of V. We bound the number of W using the trivial $q^{\frac{1}{4}\ell^2+O(\ell)}$ bound. Note that we will need to test whether W is a complement of V, which can be done easily. Another more efficient approach would be to enumerate all complements of U in time $q^{d(n-d)} \cdot \mathrm{poly}(n, \log q)$ (see [76, Proposition 17 in the arXiv version]).

So to fill in those entries corresponding to alternating matrix spaces induced by ℓ-dimensional subspaces, the time complexity is bounded by

$$\begin{aligned}
& \binom{n}{\ell}_q \cdot q^{\frac{1}{6}\ell^2+O(\ell)} \cdot q^{\frac{1}{4}\ell^2+O(\ell)} \\
\leq\ & q^{\ell(n-\ell)+\ell} \cdot q^{\frac{1}{6}\ell^2+O(\ell)} \cdot q^{\frac{1}{4}\ell^2+O(\ell)} \\
=\ & q^{\ell n - \frac{7}{12}\ell^2+O(\ell)} \\
=\ & q^{-\frac{7}{12}(\ell-\frac{6}{7}n)^2+\frac{3}{7}n^2+O(\ell)} \\
\leq\ & q^{\frac{3}{7}n^2+O(n)}.
\end{aligned}$$

Summing over $\ell \in \{0, 1, \ldots, n\}$, we see that the total time complexity is also bounded by $q^{\frac{3}{7}n^2+O(n)}$.

A new dynamic programming scheme

In the above, we see that directly following Lawler's dynamic programming scheme does lead to an improved algorithm for computing the isotropic decomposition number. However, a key difference with the classical setting, namely the magnitude of complement subspaces, impacts the analysis. In the following, we shall use another dynamic programming scheme, still combined with the $q^{\frac{1}{6}n^2+O(n)}$ upper bound on the number of maximal isotropic spaces, to achieve the $q^{\frac{5}{12}n^2+O(n)}$ running time as promised in Theorem 11.

To do that, we first make a simple observation.

▶ **Observation 36.** *Let $\mathcal{A} \leq \Lambda(n, \mathbb{F})$. Then $\chi(\mathcal{A}) \leq k$, if and only if, there exist k maximal isotropic spaces $U_1, \ldots, U_k$, such that $\mathbb{F}^n = \langle \cup_{i\in[k]} U_i \rangle$.*

Proof. For the only if direction, recall that every isotropic space is contained in a maximal one. For the if direction, note that from $U_1, \ldots, U_k$, we can construct $U'_1, \ldots, U'_k$, such that $U'_i \leq U_i$, and $U'_1, \ldots, U'_k$ form a direct sum decomposition of $\mathbb{F}^n$. This shows that $\chi(\mathcal{A}) \leq k$. ◀

The key to our algorithm is the following function. For $k \in [n]$ and $W \leq \mathbb{F}_q^n$, let $f(k, U)$ be the boolean function such that $f(k, W) = 1$ if and only if $W = \langle \cup_{i\in[k]} U_i : U_i \text{ maximal isotropic} \rangle$. For example, $f(1, W) = 1$ if and only if W is a maximal isotropic space.

The following is then a dynamic programming scheme computing $f(k, W)$ for every $k \in [n]$ and $W \leq \mathbb{F}_q^n$. Let $\mathcal{A} \leq \Lambda(n, q)$ be an alternating matrix space. We assume that $\chi(\mathcal{A}) > 1$, as $\chi(\mathcal{A}) = 1$ if and only if $\mathcal{A}$ is the zero space.

1. Use Corollary 35 to compute the set of maximal isotropic spaces of $\mathcal{A}$, and let MI be this set.
2. Build a table f, indexed by (k, W) where $k \in [n]$ and $W \leq \mathbb{F}_q^n$, and initiate $f(k, W) = 0$ for every k and W.
3. For every $W \leq \mathbb{F}_q^n$, do:
   a. If W is maximal isotropic, then $f(1, W) = 1$.
4. For $k = 2, \ldots, n$, do:
   a. For every $W \leq \mathbb{F}_q^n$ and every $T \in MI$, do:
      i. If $f(k-1, W) = 1$, then let $U = \langle W \cup T \rangle$, and set $f(k, U) = 1$.
      ii. If $U = \mathbb{F}_q^n$, then return "$\chi(\mathcal{A}) = k$."

To prove the correctness of the algorithm, we first note that by induction, the algorithm correctly computes $f(k, W)$ for every k and W. Then suppose the algorithm returns with reporting that $\chi(\mathcal{A}) = k$. Note that in this case, it does find k maximal isotropic subspaces covering the whole space $\mathbb{F}_q^n$. So by Observation 36, $\chi(\mathcal{A}) \leq k$. So we are left to show that $\chi(\mathcal{A}) \geq k$. By way of contradiction, suppose $\chi(\mathcal{A}) = k' < k$, so by Observation 36, there exist maximal isotropic subspaces $U_1, \ldots, U_{k'}$ that cover $\mathbb{F}_q^n$. Let $W = \langle U_2 \cup \cdots \cup U_{k'} \rangle$. Then W is a proper subspace of $\mathbb{F}_q^n$, as otherwise by Observation 36 $\chi(\mathcal{A}) \leq k' - 1 < k'$, contradicting that $\chi(\mathcal{A}) = k'$. But this means that $f(k' - 1, W) = 1$, so in Step (4.a), when enumerating W and $T = U_1$, the algorithm would go through steps (4.a.i) and (4.a.ii), and outputs that $\chi(\mathcal{A}) = k'$. This gives us the desired contradiction.

To estimate the running time of the algorithm, note that Step (1) costs $q^{\frac{1}{6}n^2+O(n)}$ by Corollary 35. All subspaces can be enumerated in time $q^{\frac{1}{4}n^2+O(n)}$ by the same technique as in the proof of Proposition 31. The total running time is then dominated by the loop in steps (4) and (4.a), which is $n \cdot q^{\frac{1}{6}n^2+O(n)} \cdot q^{\frac{1}{4}n^2+O(n)} = q^{\frac{5}{12}n^2+O(n)}$. This concludes the proof of Theorem 11.

10 Proofs for Propositions 12 and 13

Proof of Proposition 12. When $\mathbb{F} = \mathbb{C}$ and the input instance is over $\mathbb{Z}$, we shall formulate the maximum isotropic space problem as a problem about the solvability of a system of integeral polynomial equations over $\mathbb{C}$. The result would follow then by using Koiran's result that the Hilbert Nullstellensatz is in PH, assuming the generalized Riemann hypothesis [69]. We first cite Koiran's result as follows, following the formulation of [86, Theorem 2.10].

▶ **Theorem 37** ([69]). *The problem Hilbert's Nullstellensatz of deciding whether a given system of multivariate integral polynomials, specified as arithmetic circuits, has a solution over* $\mathbb{C}$ *is in PSPACE unconditionally, and in* $RP^{NP} \subseteq \Pi_2$ *assuming the generalized Riemann hypothesis.*

Therefore, to put the maximum isotropic space problem over $\mathbb{C}$ in PSPACE unconditionally, and in PH assuming the generalized Riemann hypothesis, for instances given by integral alternating matrices, we only need to formulate this problem as deciding the solvability of a system of integral polynomials represented by arithmetic circuits. This can be done as follows. Suppose we are given $\mathcal{A} = \langle A_1, \ldots, A_m \rangle \leq \Lambda(n, \mathbb{C})$ where A_i's are integral matrices, and we want to know whether there exists an isotropic space of dimension d for $\mathcal{A}$. Then $\mathcal{A}$ has an isotropic space of dimension d if and only if there exists an invertible matrix T such that for any $i \in [m]$, the left-upper $d \times d$ submatrix of $T^t A_i T$ consists of all zero entries. We then set up an $n \times n$ variable matrix $X = (x_{i,j})_{i,j \in [n]}$, and a variable y. For every $i \in [m]$, set the entries of the left-upper $d \times d$ submatrix of $X^t A_i X$ to be zero. This gives md^2 integral quadratic polynomials in $x_{i,j}$'s. To enforce that the valid solutions are from invertible matrices, we set up the equation $\det(X) \cdot y = 1$, which is also an integral polynomial. Note that the polynomial $\det(X)$ can be expressed as a small arithmetic circuit. It is straightforward to verify that these $(md^2 + 1)$ equations in $x_{i,j}$ and y have a non-trivial solution if and only if $\mathcal{A}$ has a dimension-d isotropic space.

For isotropic 3-decomposition problem, the idea is basically the same. The only small complication is that we need to specify the dimensions of the three isotropic spaces in a 3-isotropic decomposition. But the number of possibilities is at most n^3, which we can enumerate. After fixing some (d_1, d_2, d_3), where $d_i \in \mathbb{Z}^+$, $n \geq d_1 \geq d_2 \geq d_3 \geq 1$, and $d_1 + d_2 + d_3 = n$, we can construct a system of integral polynomial equations to express the condition that there exists a 3-isotropic decomposition with these dimensions, just as in the case of the maximum isotropic space problem. This concludes the proof. ◀

Proof of Proposition 13. Recall that we have $\mathcal{A} \leq \Lambda(n, \mathbb{F})$, and our goal is to prove $\chi(\mathcal{A}) \leq O(\Delta(\mathcal{A}) \cdot \log n)$. Here, $\Delta(\mathcal{A}) := \max\{\deg_{\mathcal{A}}(v) : v \in \mathbb{F}^n\}$, and $\deg_{\mathcal{A}}(v) := \dim(\langle Av : A \in \mathcal{A} \rangle)$. We will also use a greedy algorithm to construct an isotropic C-decomposition with $C \leq O(\Delta(\mathcal{A}) \cdot \log n)$.

For $v \in \mathbb{F}^n$, recall that $\mathrm{rad}_{\mathcal{A}}(v) = \{u \in \mathbb{F}^n : u^t A v = 0\}$. Consider the following algorithm.

1. Set $k = 0$, and $U = \mathbf{0}$;
2. While $\dim(U) < n$, do:
   a. Set $k = k + 1$;
   b. Let W be any complementary subspace of U;
   c. Let $S = \emptyset$;
   d. While $\dim(W) > |S|$, do:
      i. Take any $w \in W \setminus \langle S \rangle$; // $\langle \emptyset \rangle := \mathbf{0}$
      ii. $S = S \cup w$;
      iii. $W = W \cap \mathrm{rad}_{\mathcal{A}}(w)$;
   e. Let $U_k = \langle S \rangle$;
   f. $U = \langle U \cup U_k \rangle$;
3. Return $U_1 \oplus U_2 \oplus \cdots \oplus U_k$.

We first argue that $U_1 \oplus U_2 \oplus \cdots \oplus U_k$ is an isotropic k-decomposition of $\mathcal{A}$. To see this, we note that because of Step (2.d.iii), the condition $W \subseteq \mathrm{rad}(\langle S \rangle)$ holds in the loop of Step (d), so $\langle S \rangle$ maintains as an isotropic space by Observation 17.

We then show that $k = O(\Delta(\mathcal{A}) \cdot \log n)$ when the algorithm terminates. For this, let $d_i = \dim(U_i)$, and $D_i = d_1 + \cdots + d_i$. Observe that $\dim(\mathrm{rad}_{\mathcal{A}}(w)) \geq n - \Delta(\mathcal{A})$. It follows that $\dim(W \cap \mathrm{rad}_{\mathcal{A}}(w)) = \dim(W) + \dim(\mathrm{rad}_{\mathcal{A}}(w)) - \dim(\langle W \cup \mathrm{rad}_{\mathcal{A}}(w)\rangle) \geq \dim(W) + (n - \Delta(\mathcal{A})) - \dim(\langle W \cup \mathrm{rad}_{\mathcal{A}}(w)\rangle) \geq \dim(W) - \Delta(\mathcal{A})$. Therefore, in the computing procedure of U_i, we have $d_i = \dim(U_i) \geq \frac{n - D_{i-1}}{\Delta(\mathcal{A})}$. This implies that $n - D_i = n - D_{i-1} - d_i \leq (n - D_{i-1})(1 - 1/\Delta(\mathcal{A}))$. Therefore, adding a new U_i to the direct sum decomposition decreases the value of $n - D_i$ by a factor of at least $1 - 1/\Delta(\mathcal{A})$. Therefore the algorithm terminates in at most $\log_{1-1/\Delta(\mathcal{A})} \frac{1}{n} = O(\Delta(\mathcal{A}) \cdot \log n)$ steps. ◀

11 Proofs of theorems 14 and 15

In this section we prove theorems 14 and 15. While the proofs are straightforward for experts, we include details for completeness. We shall refer to some facts in Section 3.1 from time to time.

Proof of Theorem 14. Recall that the goal is to show that deciding whether a matrix group contains an abelian subgroup of order $\geq s$ is NP-hard for some $s \in \mathbb{N}$. We shall reduce the maximum isotropic space problem, which is NP-hard by Corollary 4, to this problem.

For this we shall need the following classical construction[11]. Let p be an odd prime, and let $\mathcal{A} \leq \Lambda(n, p)$ be given by an ordered linear basis $\mathbf{A} = (A_1, \ldots, A_m)$. Recall that e_i denotes the ith standard basis vector. From $\mathbf{A}$, for $i \in [n]$, construct $B_i = [A_1 e_i, \ldots, A_m e_i] \in \mathrm{M}(n \times m, p)$. That is, the jth column of B_i is the ith column of A_j. Then for $i \in [n]$, construct

$$\tilde{B}_i = \begin{bmatrix} 1 & e_i^t & 0 \\ 0 & I_n & B_i \\ 0 & 0 & I_m \end{bmatrix} \in \mathrm{GL}(1 + n + m, p),$$

and for $j \in [m]$, construct

$$\tilde{C}_j = \begin{bmatrix} 1 & 0 & e_j^t \\ 0 & I_n & 0 \\ 0 & 0 & I_m \end{bmatrix} \in \mathrm{GL}(1 + n + m, p).$$

Let $G_{\mathbf{A}}$ be the matrix group generated by $\tilde{B}_i$ and $\tilde{C}_j$. Then it can be verified easily that, $G_{\mathbf{A}}$ is isomorphic to the Baer group (see Section 3.1) corresponding to the alternating bilinear map defined by $\mathbf{A}$ (see Appendix B). In particular, $[G, G] \cong \mathbb{Z}_p^m$, and $G/[G, G] \cong \mathbb{Z}_p^n$. By the correspondence between isotropic spaces of $\mathcal{A}$ and abelian normal subgroups of $G_{\mathbf{A}}$ containing the commutator subgroup (see Section 3.1), deciding whether $\mathcal{A}$ has an isotropic space of dimension $\geq d$ is equivalent to deciding whether $G_{\mathbf{A}}$ has an abelian subgroup of order $\geq s = p^{m+d}$. This completes the reduction. ◀

Proof of Theorem 15. Let P be a p-group of class 2 and exponent p, and let $\phi : P/[P, P] \times P/[P, P] \to [P, P]$ be the commutator map. The proof of Theorem 15 basically follows from the correspondence between abelian subgroup containing $[P, P]$ and isotropic spaces of ϕ as described in Section 3.1. The only small caveat here is that we need a bound on the dimension of $P/Z(P)$ instead of the dimension of $P/[P, P]$. To overcome this, we first observe that a maximal abelian subgroup of P necessarily contains the center $Z(P)$, which in turn contains $[P, P]$ by the class-2 condition. Then we only need to note that $Z(P)/[P, P]$ corresponds to the radical of ϕ, and recall that the number of maximal isotropic spaces only depends on the non-degenerate part of ϕ by Observation 32. The proof then can be concluded. ◀

[11] We thank James B. Wilson for communicated this construction to us.

12 A quantum variant of the theory

One way to extend isotropic spaces and isotropic decompositions to the quantum information setting is as follows. Briefly speaking, firstly we restrict to the complex number field $\mathbb{C}$. Secondly, instead of tuples of alternating matrices, we will consider tuples of matrices which represent a so-called irreducible quantum channels. Thirdly, instead of general linear groups, we will consider unitary groups.

For detailed explanations, we need some notation. For $A \in \mathrm{M}(n, \mathbb{C})$ we use $A \succeq 0$ to denote that A is positive semi-definite, and $A \succ 0$ to denote that A is positive definite. For $\mathsf{B} = \{B_1, \ldots, B_m\} \subseteq \mathrm{M}(n, \mathbb{C})$, we let $\tilde{\mathsf{B}} : \mathrm{M}(n, \mathbb{C}) \to \mathrm{M}(n, \mathbb{C})$ be the function sending $A \in \mathrm{M}(n, \mathbb{C})$ to $\sum_{i=1}^{m} B_i A B_i^\dagger$. It is clear that $\tilde{\mathsf{B}}$ can be represented as an $n^2 \times n^2$ matrix $\sum_{i=1}^{m} B_i \otimes B_i^*$, where B_i^* stands for the entry-wise complex conjugation of B_i.

Let $\mathrm{D}(n, \mathbb{C}) \subseteq \mathrm{M}(n, \mathbb{C})$ be the set of $n \times n$ semi-positive definite matrices with unit trace over $\mathbb{C}$, and let $\mathrm{D}^+(n, \mathbb{C}) \subseteq \mathrm{D}(n, \mathbb{C})$ consist of those positive definite matrices in $\mathrm{D}(n, \mathbb{C})$. Elements from $\mathrm{D}(n, \mathbb{C})$ are known as quantum states.

Let $\mathrm{QC}(n, \mathbb{C})$ be the set of sets of matrices $\mathsf{B} = \{B_1, \ldots, B_m\} \subseteq \mathrm{M}(n, \mathbb{C})$ satisfying $\sum_{i=1}^{m} B_i^\dagger B_i = I$. Functions of the form $\tilde{\mathsf{B}}$ for $\mathsf{B} \in \mathrm{QC}(n, \mathbb{C})$ are known as quantum channels, as they are completely positive and trace preserving.

We then define isotropic spaces and decompositions in the quantum setting. To define isotropic spaces, we essentially follow the same pattern as in the alternating matrix space setting. For isotropic decompositions, we shall require that the direct sum decomposition is also an orthogonal one, as the underlying spaces of quantum channels are Hilbert spaces which come with a norm.

▶ **Definition 38.** *Let* $\mathsf{B} = \{B_1, \ldots, B_m\} \in \mathrm{QC}(n, \mathbb{C})$. *An isotropic space of* B *is a subspace* $U \leq \mathbb{C}^n$, *such that for any* $u, u' \in U$, *and any* B_i, *we have* $u^\dagger B_i u' = 0$. *An isotropic c-decomposition of* B *is an orthogonal direct sum decomposition of* $\mathbb{C}^n = U_1 \oplus U_2 \oplus \cdots \oplus U_c$ *such that each* U_i *is a non-zero isotropic space of* B.

12.1 From connected graphs to irreducible quantum channels

In this subsection, we establish a connection between independent sets and vertex colorings of connected graphs, and isotropic spaces and decompositions of a particular type of quantum channels, called irreducible channels (defined below).

We obtain two main results. The first result, Proposition 40, reduces certain problems for connected graphs to the corresponding ones for irreducible quantum channels. This result corresponds to Theorem 3. The second result, Theorem 42, gives an efficient algorithm for isotropic 2-decomposition in this setting. This result corresponds to Theorem 5, but the techniques are completely different.

Let $\mathrm{IQC}(n, \mathbb{C}) \subseteq \mathrm{QC}(n, \mathbb{C})$ consist of those $\mathsf{B} \in \mathrm{QC}(n, \mathbb{C})$ satisfying the following: there exists a unique fixed $\rho \in \mathrm{D}(n, \mathbb{C})$ of $\tilde{\mathsf{B}}$, and further $\rho \in \mathrm{D}^+(n, \mathbb{C})$, where ρ is said to be fixed of $\tilde{\mathsf{B}}$ if $\tilde{\mathsf{B}}(\rho) = \rho$. Such B and $\tilde{\mathsf{B}}$ are called *irreducible*. Irreducible quantum channels have been studied in e.g. [36] and [114, Sec. 6.2]. In particular, the definition of irreducible quantum channels follows from [36, Theorem 13]. Furthermore, given $\mathsf{B} \in \mathrm{QC}(n, \mathbb{C})$, let M be the $n^2 \times n^2$ matrix representation of $\tilde{\mathsf{B}}$. Then $\mathsf{B} \in \mathrm{IQC}(n, \mathbb{C})$ if and only if the algebraic and geometric multiplicities of the eigenvalue 1 of M are both 1, and any 1-eigenvector is of full-rank.

We first observe that a simple and connected graph can be realized as an irreducible quantum channel as follows. This is classical, but for completeness we spell out the details. Let $G = ([n], E)$ be a connected graph. For each $i \in [n]$, let d_i be the degree of i. We construct the following set of matrices $\mathsf{B}_G = \{\frac{1}{\sqrt{d_j}} \cdot E_{i,j}, \frac{1}{\sqrt{d_i}} \cdot E_{j,i} : \{i,j\} \in E\}$. Note that $|\mathsf{B}_G| = 2|E|$.

▶ **Proposition 39.** *Let G and B_G be as above. Then $\mathsf{B}_G \in \mathrm{IQC}(n, \mathbb{C})$.*

Proof. We first verify that $\tilde{\mathsf{B}}_G$ is a quantum channel. For this, observe that $(\frac{1}{\sqrt{d_j}} \cdot E_{i,j})^\dagger \frac{1}{\sqrt{d_j}} \cdot E_{i,j} = \frac{1}{d_j} E_{j,i} E_{i,j} = \frac{1}{d_j} E_{j,j}$. Since each vertex i connecting to j contributes one such term, and it follows that $\sum_{E \in \mathsf{B}_G} E^\dagger E = I$. We then verify that $\tilde{\mathsf{B}}_G$ is irreducible. For this, consider $P = (p_{i,j})$ where $p_{i,j} = 1/d_i$, which represents the transition matrix of the Markov chain naturally associated with G. Since G is connected, this Markov chain is irreducible, so there exists a unique probability distribution, e.g. a row vector $s = (s_1, \ldots, s_n)$ satisfying $s_i > 0$, $\sum_i s_i = 1$, such that $sP = s$ (see e.g. [74, Corollary 1.17]). It can then be verified that the matrix $S = \mathrm{diag}(s_1, \ldots, s_n) \in \mathrm{D}^+(n, \mathbb{C})$ is fixed by $\tilde{\mathsf{B}}_G$. To see that this is the unique fixed state, we represent $\tilde{\mathsf{B}}_G$ as an $n^2 \times n^2$ matrix M_G. It is not hard to see that by conjugating with a permutation matrix, M_G is of the form $\begin{bmatrix} P & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}$. Therefore, the algebraic and geometric multiplicities of the eigenvalue 1 of M_G are the same as those for P, which are 1 by the Perron-Frobenius theory. It follows that B is irreducible. ◀

▶ **Proposition 40.** *Let $G = ([n], E)$ be a connected graph, and let $\mathsf{B}_G \in \mathrm{IQC}(n, \mathbb{C})$ be as above.*

1. *G has a size-s independent set if and only if B_G has a dimension-s isotropic space;*
2. *G has a vertex c-coloring if and only if B_G has an isotropic c-decomposition.*

Proof. (1) The only if direction is trivial. For the if direction, let U be a dimension-s isotropic space of B_G. Then U is also a dimension-s isotropic space of the alternating matrix space $\langle E_{i,j} - E_{j,i} : \{i,j\} \in E\rangle$, because it is a subspace of $\langle \mathsf{B}_G \rangle$. We can then conclude by resorting to Theorem 3.

(2) The only if direction is trivial; observe that the direct sum decomposition obtained from a vertex coloring as in Theorem 3 is also an orthogonal direct sum decomposition. For the if direction, we observe that, an orthogonal direct sum decomposition into isotropic spaces for B_G is also one for the alternating matrix space $\langle E_{i,j} - E_{j,i} : \{i,j\} \in E\rangle$. We can then conclude by resorting to Theorem 3. ◀

Since the maximum independent set problem and the vertex 3-coloring problem on connected graphs are also NP-hard, we have the following.

▶ **Corollary 41.** *The maximum isotropic space problem and the isotropic 3-decomposition problem for $\mathsf{B} \in \mathrm{IQC}(n, \mathbb{C})$ are NP-hard.*

This also leaves the isotropic 2-decomposition problem an interesting question. For this, we can resort to the techniques developed for quantum Markov chains, mostly notably, based on recent works of periodicity of quantum channels [53].

▶ **Theorem 42.** *Suppose we are given $\mathsf{B} \in \mathrm{IQC}(n, \mathbb{C})$ such that every matrix in B are over $\mathbb{Q}$. There exists an algorithm that decides whether B admits an isotropic 2-decomposition in polynomial time.*

Proof. The key observation is to characterize isotropic 2-decompositions using the *periodicity* of irreducible quantum channels.

▶ **Definition 43** ([45]). *Given* $\mathsf{B} \in \mathrm{IQC}(n, \mathbb{C})$*, the* period *of* B *is the maximum integer* m *for which there exists an orthogonal direct sum decomposition* $\mathbb{C}^n = U_1 \oplus \cdots \oplus U_m$ *such that for any* $i \in [m]$*, and any* $B \in \mathsf{B}$*, we have* $B(U_{i \boxminus 1}) \leq U_i$*, where* $\boxminus$ *indicates subtraction modulo* m *in the range of* $[m]$*.*

The following lemma relates isotropic 2-decompositions with periodicity.

▶ **Lemma 44.** *Given* $\mathsf{B} \in \mathrm{IQC}(n, \mathbb{C})$*,* B *admits an isotropic* 2*-decomposition if and only if the period of* B *is* $2k$ *for some integer* k*.*

Proof. For the if direction, let $\mathbb{C}^n = U_1 \oplus U_2 \oplus \cdots \oplus U_{2k}$ be the orthogonal direct sum decomposition corresponding to the period of B. Let $V_1 = \langle U_i : i = 2j - 1, j \in [k] \rangle$, and $V_2 = \langle U_i : i = 2j, j \in [k] \rangle$. Then $V_1 \oplus V_2$ is an orthogonal direct sum decomposition, and for any $B \in \mathsf{B}$, $B(V_1) \leq V_2$, and $B(V_2) \leq V_1$. By the orthogonal condition, $v_1^\dagger v_2 = 0$ for any $v_1 \in V_1, v_2 \in V_2$. We then have for any $i = 1, 2$, any $v_i, v_i' \in V_i$, and any $B \in \mathsf{B}$, we have $v_i^\dagger B v_i' = 0$. That is, V_1 and V_2 are isotropic spaces.

For the only if direction, let $\mathbb{C}^n = V_1 \oplus V_2$ be an isotropic 2-decomposition. Let P_1 be the projection into V_1 along V_2, and P_2 the projection into V_2 along V_1. We have $P_1 + P_2 = I$, and $P_i^\dagger = P_i$. Since V_1 and V_2 are isotropic spaces, for any $B \in \mathsf{B}$, and any $i = 1, 2$, $P_i B P_i = \mathbf{0}$. Using $P_1 + P_2 = I$, it follows that $P_2 B = B P_1$, and $P_1 B = B P_2$. We are then in the position to apply [45, Lemma 4.2], to conclude that the period of B is $2k$ for some integer k. ◀

Given Lemma 44, it is enough to compute the period of B, and this can be done by resorting to the algorithm in [53]. For completeness, we give a brief sketch of the idea. By Lemma 13 of [53], the period of irreducible quantum channel is equivalent to be the number of eigenvalues with magnitude one of the quantum channel. Using the terminologies in the present article, we have the following lemma.

▶ **Lemma 45** ([53, Lemma 13]). *Given* $\mathsf{B} \in \mathrm{IQC}(n, \mathbb{C})$*, the period of* B *is equal to the number of eigenvalues of* $\tilde{\mathsf{B}}$ *with magnitude one.*

Given this lemma, we can explicitly write out the form of $\tilde{\mathsf{B}}$ as an $n^2 \times n^2$ matrix, and compute its eigenvalues using e.g. [26] in the exact model (Section 2.1). Therefore, each eigenvalue α is represented by an irreducible polynomial $f(x)$ and a separating rectangle in the complex plane. To decide whether α has magnitude 1 can be done efficiently by resorting to techniques from [80]. ◀

Finally, we remark that the investigation in this subsection is not completely satisfactory. It would be more satisfying to consider isotropic spaces and isotropic decompositions for arbitrary quantum channels, not just the irreducible ones. We adopt the current strategy, partly because for irreducible channels, the periodicity is well-studied and well-connected with isotropic 2-decomposition. We leave it a future work to study isotropic spaces and decompositions in the general setting.

12.2 Quantum gate subspace-fidelity and isotropic spaces

We provide one quantum information theoretic interpretation for isotropic spaces, by relating it to quantum gate (state) fidelity [88, Section 9] and noiseless subspaces in quantum error correction [68, 77]. For the sake of readers who have little quantum information knowledge, we shall proceed by introducing all the necessary notions from quantum information, even though most of them are standard.

In quantum information theory, the fidelity is a measure of the "closeness" of two quantum states, generalizing the fidelity of two distributions over finite events. It expresses the probability that one state will pass a test (quantum measurements) to identify as the other. Formally, the fidelity of two quantum states $\rho, \sigma \in \mathrm{D}(n, \mathbb{C})$ is defined by

$$F(\rho, \sigma) = [\mathrm{tr}(\sqrt{\sqrt{\rho}\sigma\sqrt{\rho}})]^2.$$

It is worth noting that $0 \leq F(\rho, \sigma) \leq 1$. Furthermore,

- $F(\rho, \sigma) = 0$ if and only if ρ and σ are orthogonal, i.e., $\mathrm{tr}(\rho\sigma) = 0$;
- $F(\rho, \sigma) = 1$ if and only if $\rho = \sigma$.

Quantum state fidelity induces quantum gate fidelity. Unitary channels (i.e. channels of the form $\tilde{V}(A) = VAV^\dagger$ for some unitary matrix $V \in M(n, \mathbb{C})$) are exactly the channels that do not introduce mixedness (i.e., decoherence) into states. Therefore, in experimental settings, they are considered to be the ideal type of channels to be implemented [88, Section 8]. However, no implementation of a channel is perfect, as there is no closed (isolated) system, so environment errors are unavoidable, which cause the channel actually implemented to be not unitary. The gate fidelity is a tool for comparing how well the implemented quantum channel $\tilde{\mathsf{B}}$ approximates the desired unitary channel $\tilde{V}$. Specifically, the gate fidelity on a pure state ($uu^\dagger$ for a normalized vector $u \in \mathbb{C}^n$) is a function defined as follows:

$$F_{\tilde{\mathsf{B}},\tilde{V}}(u) = F(\tilde{\mathsf{B}}(uu^\dagger), Vuu^\dagger V^\dagger) = u^\dagger V^\dagger \tilde{\mathsf{B}}(uu^\dagger) Vu = u^\dagger [\tilde{V}^\dagger \circ \tilde{\mathsf{B}}(uu^\dagger)]u,$$

where $\tilde{V}^\dagger(A) = V^\dagger AV$. In particular, $F_{\tilde{\mathsf{B}},\tilde{V}}(u) = F_{\tilde{V}^\dagger \circ \tilde{\mathsf{B}},\tilde{I}}(u)$, where $\tilde{I}$ is the identity channel. Then the gate fidelity on all states is defined as follows:

$$F(\tilde{\mathsf{B}}) = \min_{u \in \mathbb{C}^n} F_{\tilde{\mathsf{B}},\tilde{I}}(u) = \min_{\rho \in D(n,\mathbb{C})} F_{\tilde{\mathsf{B}},\tilde{I}}(\rho)$$

The second equation in the above follows from the joint concavity property of the state fidelity F (see [88, Equation 9.121]).

As we can see, quantum gate fidelity is a global property over $\mathbb{C}^n$. But in some cases, we only need a subspace of $\mathbb{C}^n$ as the state space of quantum information processing. This consideration motivates the following notions, which we call them quantum gate maximum and minimum subspace-fidelities, respectively. For a subspace $U \subseteq \mathbb{C}^n$,

$$F_U^{min}(\tilde{\mathsf{B}}) = \min_{u \in U} F_{\tilde{\mathsf{B}},\tilde{I}}(u), \qquad \text{and } F_U^{max}(\tilde{\mathsf{B}}) = \max_{u \in U} F_{\tilde{\mathsf{B}},\tilde{I}}(u).$$

Note that $F_U^{min}(\tilde{\mathsf{B}})$ and $F_U^{max}(\tilde{\mathsf{B}})$ quantify the worst-case and best-case behavior of the system by minimizing and maximizing over all possible initial states, respectively. Obviously, $0 \leq F_U^{min}(\tilde{\mathsf{B}}) \leq F_U^{max}(\tilde{\mathsf{B}}) \leq 1$ as $0 \leq F(\rho, \sigma) \leq 1$. We also have $F(\tilde{\mathsf{B}}) = \min_{U \subseteq \mathbb{C}^n} F_U^{min}(\tilde{\mathsf{B}}) = \min_{U \subseteq \mathbb{C}^n} F_U^{max}(\tilde{\mathsf{B}})$, where the second equation follows by examining one-dimensional subspaces.

One key notion in quantum error correction is that of noiseless subspaces, which have been intensively discussed in the setting where $\tilde{\mathsf{B}}$ as a noise model [68, 77]. Intuitively, noiseless subspaces are shelters under quantum noise $\tilde{\mathsf{B}}$, as they perfectly preserve quantum states under $\tilde{\mathsf{B}}$.

▶ **Definition 46.** *Let* $\mathsf{B} = \{B_1, \ldots, B_m\} \in \mathrm{QC}(n, \mathbb{C})$. *An noiseless subspace of* B *is a non-zero subspace* $U \leq \mathbb{C}^n$, *such that for any* $u \in U$, *and any* B_i, *we have* $B_i u = u$.

The following result formally shows that noiseless subspaces and isotropic subspaces are totally opposite from the viewpoint of the two quantum gate subspace-fidelities.

▶ **Proposition 47.** *Let* $\mathsf{B} = \{B_1, \ldots, B_m\} \in \mathrm{QC}(n, \mathbb{C})$.

- $F_U^{max}(\tilde{\mathsf{B}}) = 0$ *if and only if* U *is an isotropic space;*
- $F_U^{max}(\tilde{\mathsf{B}}) = 1$ *if and only if there is a noiseless subspace in* U*;*
- $F_U^{min}(\tilde{\mathsf{B}}) = 1$ *if and only if* U *is a noiseless subspace;*
- $F_U^{min}(\tilde{\mathsf{B}}) = 0$ *if and only if there is an isotropic space in* U.

Proof. These four claims are directly from the definitions of isotropic spaces, noiseless subspaces, quantum gate minimum subspace-fidelity and maximum subspace-fidelity. ◀

Knill devised an efficient algorithm to find all noiseless subspaces for a given $\tilde{\mathsf{B}}$ [67]. So we have a quite good understanding on $F_U^{min}(\tilde{\mathsf{B}}) = 1$. On the other hand, isotropic subspaces fully characterize $F_U^{max}(\tilde{\mathsf{B}}) = 0$. Therefore, isotropic spaces reveal the structure of the worst-case behavior of the channel.

Let us further point out another potential application of isotropic spaces in quantum control. A basic task of controlling quantum systems is to transfer all unknown quantum states into some targeting subspace [32, 102]. So designing a control scheme as a quantum channel with a non-trivial isotropic space (the dimension greater than 1) can turn all quantum states residing in the isotropic space into the orthogonal complement of it.

References

1. J. F. Adams. Vector fields on spheres. *Annals of Mathematics*, pages 603–632, 1962.
2. J. F. Adams, Peter D. Lax, and Ralph S. Phillips. On Matrices Whose Real Linear Combinations are Nonsingular. *Proceedings of the American Mathematical Society*, 16(2):318–322, 1965.
3. A. A. Albert. *Structure of Algebras*. Number v. 24 in American Mathematical Society colloquium publications. American Mathematical Society, 1939. URL: `https://books.google.com.au/books?id=1GOHcOcoJ1cC`.
4. J. L. Alperin. Large abelian subgroups of p-groups. *Transactions of the American Mathematical Society*, 117:10–20, 1965.
5. J. L. Alperin and R. B. Bell. *Groups and representations*. Number 162 in Graduate texts in mathematics. Springer, 1995. URL: `https://books.google.com.au/books?id=q2MPAQAAMAAJ`.
6. VA Antonov. Finite groups with a modular lattice of centralizers. *Algebra and Logic*, 26(6):403–422, 1987.
7. J. Gary Augustson and Jack Minker. An analysis of some graph theoretical cluster techniques. *Journal of the ACM*, 17(4):571–588, 1970.
8. László Babai. Trading Group Theory for Randomness. In Robert Sedgewick, editor, *Proceedings of the 17th Annual ACM Symposium on Theory of Computing, May 6-8, 1985, Providence, Rhode Island, USA*, pages 421–429. ACM, 1985. `doi:10.1145/22145.22192`.
9. László Babai. Graph isomorphism in quasipolynomial time [extended abstract]. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 684–697, 2016. arXiv:1512.03547, version 2. `doi:10.1145/2897518.2897542`.
10. László Babai, Robert Beals, and Ákos Seress. Polynomial-time theory of matrix groups. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 55–64, 2009. `doi:10.1145/1536414.1536425`.

11 László Babai and Endre Szemerédi. On the Complexity of Matrix Group Problems I. In *25th Annual Symposium on Foundations of Computer Science, West Palm Beach, Florida, USA, 24-26 October 1984*, pages 229–240, 1984. doi:10.1109/SFCS.1984.715919.

12 Reinhold Baer. Groups with abelian central quotient group. *Transactions of the American Mathematical Society*, 44(3):357–386, 1938.

13 Robert Beals. Towards polynomial time algorithms for matrix groups. In Larry Finkelstein and William M. Kantor, editors, *Groups and Computation, Proceedings of a DIMACS Workshop, New Brunswick, New Jersey, USA, June 7-10, 1995*, volume 28 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 31–54. DIMACS/AMS, 1995. doi:10.1090/dimacs/028/03.

14 Stuart J. Berkowitz. On Computing the Determinant in Small Parallel Time Using a Small Number of Processors. *Inf. Process. Lett.*, 18(3):147–150, 1984. doi:10.1016/0020-0190(84)90018-8.

15 Andreas Björklund and Thore Husfeldt. Exact algorithms for exact satisfiability and number of perfect matchings. *Algorithmica*, 52(2):226–249, 2008.

16 Andreas Björklund, Thore Husfeldt, Petteri Kaski, Mikko Koivisto, Jesper Nederlof, and Pekka Parviainen. Fast Zeta Transforms for Lattices with Few Irreducibles. *ACM Trans. Algorithms*, 12(1):4:1–4:19, 2016. doi:10.1145/2629429.

17 Andreas Björklund, Thore Husfeldt, and Mikko Koivisto. Set partitioning via inclusion-exclusion. *SIAM Journal on Computing*, 39(2):546–563, 2009.

18 Béla Bollobás. *Random Graphs*. Number 73 in Cambridge Studies in Advanced Mathematics. Cambridge University Press, second edition, 2001.

19 Ada Boralevi, Daniele Faenzi, and Emilia Mezzetti. Linear spaces of matrices of constant rank and instanton bundles. *Advances in Mathematics*, 248:895–920, 2013.

20 Peter A. Brooksbank, Joshua Maglione, and James B. Wilson. A fast isomorphism test for groups whose Lie algebra has genus 2. *Journal of Algebra*, 473:545–590, 2017.

21 Peter A. Brooksbank and James B. Wilson. Computing isometry groups of Hermitian maps. *Trans. Amer. Math. Soc.*, 364:1975–1996, 2012.

22 Joe Buhler, Ranee Gupta, and Joe Harris. Isotropic subspaces for skewforms and maximal abelian subgroups of p-groups. *Journal of Algebra*, 108(1):269–279, 1987.

23 W. Burnside. On some properties of groups whose orders are powers of primes. *Proceedings of the London Mathematical Society*, 2(1):225–245, 1913.

24 Jonathan F Buss, Gudmund S Frandsen, and Jeffrey O Shallit. The computational complexity of some problems of linear algebra. *Journal of Computer and System Sciences*, 58(3):572–596, 1999.

25 Jesper Makholm Byskov. Enumerating maximal independent sets with applications to graph colouring. *Operations Research Letters*, 32(6):547–556, 2004.

26 Jin-yi Cai. Computing Jordan Normal Forms Exactly for Commuting Matrices in Polynomial Time. *Int. J. Found. Comput. Sci.*, 5(3/4):293–302, 1994. doi:10.1142/S0129054194000165.

27 John F. Canny. Some Algebraic and Geometric Computations in PSPACE. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, pages 460–467, 1988. doi:10.1145/62212.62257.

28 Marco Carmosino, Russell Impagliazzo, Valentine Kabanets, and Antonina Kolokolova. Tighter Connections between Derandomization and Circuit Lower Bounds. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2015, August 24-26, 2015, Princeton, NJ, USA*, pages 645–658, 2015. doi:10.4230/LIPIcs.APPROX-RANDOM.2015.645.

29 Fabrizio Catanese. Moduli and classification of irregular Kaehler manifolds (and algebraic varieties) with Albanese general type fibrations. *Inventiones mathematicae*, 104(1):263–289, 1991.

30 Andrea Causin and Gian Pietro Pirola. A note on spaces of symmetric matrices. *Linear Algebra and Its Applications*, 2(426):533–539, 2007.

31 Alexander L. Chistov. Fast parallel calculation of the rank of matrices over a field of arbitrary characteristic. In Lothar Budach, editor, *Fundamentals of Computation Theory, FCT '85, Cottbus, GDR, September 9-13, 1985*, volume 199 of *Lecture Notes in Computer Science*, pages 63–69. Springer, 1985. doi:10.1007/BFb0028792.

32 Giuseppe Ilario Cirillo and Francesco Ticozzi. Decompositions of Hilbert spaces, stability analysis and convergence probabilities for discrete-time quantum dynamical semigroups. *Journal of Physics A: Mathematical and Theoretical*, 48(8):085302, 2015.

33 Henry Cohn, Robert D. Kleinberg, Balázs Szegedy, and Christopher Umans. Group-theoretic Algorithms for Matrix Multiplication. In *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2005), 23-25 October 2005, Pittsburgh, PA, USA, Proceedings*, pages 379–388. IEEE Computer Society, 2005. doi:10.1109/SFCS.2005.39.

34 P. M. Cohn. The Word Problem for Free Fields: A Correction and an Addendum. *J. Symbolic Logic*, 40(1):69–74, March 1975. URL: http://projecteuclid.org/euclid.jsl/1183739310.

35 P. M. Cohn and C. Reutenauer. On the construction of the free field. *International Journal of Algebra and Computation*, 9(3-4):307–323, 1999.

36 E. B. Davies. Quantum stochastic processes II. *Communications in Mathematical Physics*, 19(2):83–105, 1970.

37 Clément de Seguins Pazzis. Large affine spaces of non-singular matrices. *Transactions of the American Mathematical Society*, 365(5):2569–2596, 2013.

38 H. Derksen. Polynomial bounds for rings of invariants. *Proceedings of the American Mathematical Society*, 129(4):955–964, 2001.

39 H. Derksen and V. Makam. Polynomial degree bounds for matrix semi-invariants. *Advances in Mathematics*, 310:44–63, 2017. doi:10.1016/j.aim.2017.01.018.

40 Reinhard Diestel. *Graph Theory.* Number 173 in Springer Graduate Texts in Mathematics. Springer, 5th edition, 2017.

41 Alexandru Dimca. On the isotropic subspace theorems. *Bulletin mathématique de la Société des Sciences Mathématiques de Roumanie*, pages 307–324, 2008.

42 John D. Dixon. Maximal Abelian Subgroups of the Symmetric Groups. *Canadian Journal of Mathematics*, 23(3):426–438, 1971. doi:10.4153/CJM-1971-045-7.

43 Wayne Eberly. Decompositions of Algebras Over R and C. *Computational Complexity*, 1:211–234, 1991. doi:10.1007/BF01200061.

44 David Eppstein. Small maximal independent sets and faster exact graph coloring. *J. Graph Algorithms Appl.*, 7(2):131–140, 2003.

45 Franco Fagnola and Rely Pellicer. Irreducible and periodic positive maps. *Commun. Stoch. Anal*, 3(3):407–418, 2009.

46 Stephen A. Fenner, Rohit Gurjar, and Thomas Thierauf. Bipartite perfect matching is in quasi-NC. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 754–763, 2016. doi:10.1145/2897518.2897564.

47 M. Fortin and C. Reutenauer. Commutative/Noncommutative Rank of Linear Matrices and Subspaces of Matrices of Low Rank. *Séminaire Lotharingien de Combinatoire*, 52:B52f, 2004.

48 Ankit Garg, Leonid Gurvits, Rafael Mendes de Oliveira, and Avi Wigderson. A Deterministic Polynomial Time Algorithm for Non-commutative Rational Identity Testing. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 109–117, 2016. doi:10.1109/FOCS.2016.95.

49 Ankit Garg, Leonid Gurvits, Rafael Mendes de Oliveira, and Avi Wigderson. Algorithmic and optimization aspects of Brascamp-Lieb inequalities, via operator scaling. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 397–409, 2017. doi:10.1145/3055399.3055458.

50 I. Gelbukh. Isotropy index for the connected sum and the direct product of manifolds. *Publicationes Mathematicae Debrecen*, 90(3-4):287–310, 2017.

51 Daniel Goldstein and Robert M. Guralnick. Alternating forms and self-adjoint operators. *Journal of Algebra*, 308(1):330–349, 2007.

52 R. Gow and T. J. Laffey. Pairs of alternating forms and products of two skew-symmetric matrices. *Linear algebra and its applications*, 63:119–132, 1984.

53 Ji Guan, Yuan Feng, and Mingsheng Ying. Decomposition of quantum Markov chains and its applications. *Journal of Computer and System Sciences*, 95:55–68, 2018.

54 Leonid Gurvits. Classical complexity and quantum entanglement. *J. Comput. Syst. Sci.*, 69(3):448–484, 2004. doi:10.1016/j.jcss.2004.06.003.

55 P. Hall. On Representatives of Subsets. *Journal of the London Mathematical Society*, 1(1):26–30, 1935.

56 Pavel Hrubeš and Avi Wigderson. Non-Commutative Arithmetic Circuits with Division. *Theory of Computing*, 11:357–393, 2015. doi:10.4086/toc.2015.v011a014.

57 Bo Ilic and J. M. Landsberg. On symmetric degeneracy loci, spaces of symmetric matrices of constant rank and dual varieties. *Mathematische Annalen*, 314(1):159–174, 1999.

58 Gábor Ivanyos, Marek Karpinski, Youming Qiao, and Miklos Santha. Generalized Wong sequences and their applications to Edmonds' problems. *J. Comput. Syst. Sci.*, 81(7):1373–1386, 2015. doi:10.1016/j.jcss.2015.04.006.

59 Gábor Ivanyos and Youming Qiao. Algorithms Based on *-Algebras, and Their Applications to Isomorphism of Polynomials with One Secret, Group Isomorphism, and Polynomial Identity Testing. *SIAM Journal on Computing*, 48(3):926–963, 2019. doi:10.1137/18M1165682.

60 Gábor Ivanyos, Youming Qiao, and K. V. Subrahmanyam. Non-commutative Edmonds' problem and matrix semi-invariants. *Computational Complexity*, 26(3):717–763, 2017. doi:10.1007/s00037-016-0143-x.

61 Gábor Ivanyos, Youming Qiao, and K. V. Subrahmanyam. Constructive non-commutative rank computation is in deterministic polynomial time. *Computational Complexity*, 27(4):561–593, 2018. doi:10.1007/s00037-018-0165-7.

62 Gábor Ivanyos and Lajos Rónyai. Computations in associative and Lie algebras. In *Some tapas of computer algebra*, pages 91–120. Springer, 1999.

63 Tommy R. Jensen and Bjarne Toft. *Graph coloring problems*, volume 39 of *Wiley-Interscience series in discrete mathematics and optimization*. John Wiley & Sons, 1995.

64 David S. Johnson, Christos H. Papadimitriou, and Mihalis Yannakakis. On Generating All Maximal Independent Sets. *Inf. Process. Lett.*, 27(3):119–123, 1988. doi:10.1016/0020-0190(88)90065-8.

65 Valentine Kabanets and Russell Impagliazzo. Derandomizing Polynomial Identity Tests Means Proving Circuit Lower Bounds. *Computational Complexity*, 13(1-2):1–46, 2004. doi:10.1007/s00037-004-0182-6.

66 Richard M. Karp, Eli Upfal, and Avi Wigderson. Constructing a perfect matching is in random NC. *Combinatorica*, 6(1):35–48, 1986. doi:10.1007/BF02579407.

67 Emanuel Knill. Protected realizations of quantum information. *Physical Review A*, 74(4):042301, 2006.

68 Emanuel Knill and Raymond Laflamme. Theory of quantum error-correcting codes. *Physical Review A*, 55(2):900, 1997.

69 Pascal Koiran. Hilbert's Nullstellensatz Is in the Polynomial Hierarchy. *J. Complexity*, 12(4):273–286, 1996. doi:10.1006/jcom.1996.0019.

70 Kee Yuen Lam and Paul Yiu. Linear spaces of real matrices of constant rank. *Linear algebra and its applications*, 195:69–79, 1993.

71 Serge Lang. *Algebra*. Number 211 in Graduate Texts in Mathematics. Springer-Verlag, New York, third enlarged edition, 2002.

72 Eugene L. Lawler. A note on the complexity of the chromatic number problem. *Inf. Proc. Lett.*, 5:66–67, 1976.

73 Eugene L. Lawler, Jan Karel Lenstra, and A. H. G. Rinnooy Kan. Generating all Maximal Independent Sets: NP-Hardness and Polynomial-Time Algorithms. *SIAM J. Comput.*, 9(3):558–565, 1980. doi:10.1137/0209042.

74 David A. Levin and Yuval Peres. *Markov chains and mixing times*, volume 107. American Mathematical Soc., 2017.

75 Mark L. Lewis and James B. Wilson. Isomorphism in expanding families of indistinguishable groups. *Groups - Complexity - Cryptology*, 4(1):73–110, 2012.

76 Yinan Li and Youming Qiao. Linear Algebraic Analogues of the Graph Isomorphism Problem and the Erdős-Rényi Model. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 463–474, 2017. arXiv:1708.04501, v2. doi:10.1109/FOCS.2017.49.

77 Daniel A Lidar. Review of decoherence free subspaces, noiseless subsystems, and dynamical decoupling. *Adv. Chem. Phys.*, 154:295–354, 2014.

78 Nathan Linial, Alex Samorodnitsky, and Avi Wigderson. A Deterministic Strongly Polynomial Algorithm for Matrix Scaling and Approximate Permanents. *Combinatorica*, 20(4):545–568, 2000. doi:10.1007/s004930070007.

79 László Lovász. On determinants, matchings, and random algorithms. In *FCT*, pages 565–574, 1979.

80 László Lovász. *An Algorithmic Theory of Numbers, Graphs, and Convexity*, volume 50 of *CBMS-NSF Regional Conference Series in Applied Mathematics*. SIAM, 1986.

81 László Lovász. Singular spaces of matrices and their application in combinatorics. *Boletim da Sociedade Brasileira de Matemática-Bulletin/Brazilian Mathematical Society*, 20(1):87–99, 1989.

82 Roy Meshulam. On k-spaces of real matrices. *Linear and multilinear algebra*, 26(1-2):39–41, 1990.

83 Gary L. Miller. On the $n \log n$ isomorphism technique (A Preliminary Report). In *STOC*, pages 51–58, New York, NY, USA, 1978. ACM. doi:10.1145/800133.804331.

84 John W. Moon and Leo Moser. On cliques in graphs. *Israel Journal of Mathematics*, 3(1):23–28, 1965.

85 Ketan Mulmuley. Geometric Complexity Theory V: Equivalence between Blackbox Derandomization of Polynomial Identity Testing and Derandomization of Noether's Normalization Lemma. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012*, pages 629–638, 2012. doi:10.1109/FOCS.2012.15.

86 Ketan Mulmuley. Geometric complexity theory V: Efficient algorithms for Noether normalization. *Journal of the American Mathematical Society*, 30(1):225–309, 2017.

87 Ketan Mulmuley, Umesh V. Vazirani, and Vijay V. Vazirani. Matching is as easy as matrix inversion. *Combinatorica*, 7(1):105–113, 1987. doi:10.1007/BF02579206.

88 Michael A Nielsen and Isaac Chuang. Quantum computation and quantum information, 2002.

89 A. Yu Ol'shanskii. The number of generators and orders of abelian subgroups of finite p-groups. *Mathematical notes of the Academy of Sciences of the USSR*, 23(3):183–185, 1978.

90 Youming Qiao. Matrix spaces as a linear algebraic analogue of graphs. Under preparation, 2019.

91 James Renegar. On the Computational Complexity and Geometry of the First-Order Theory of the Reals, Part I: Introduction. Preliminaries. The Geometry of Semi-Algebraic Sets. The Decision Problem for the Existential Theory of the Reals. *J. Symb. Comput.*, 13(3):255–300, 1992. doi:10.1016/S0747-7171(10)80003-3.

92 Lajos Rónyai. Simple Algebras Are Difficult. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*, pages 398–408, 1987. doi:10.1145/28395.28438.

93 Lajos Rónyai. Computing the structure of finite algebras. *Journal of Symbolic Computation*, 9(3):355–373, 1990.

94 Lajos Rónyai. A Deterministic Method for Computing Splitting Elements in Simple Algebras over Q. *J. Algorithms*, 16(1):24–32, 1994. doi:10.1006/jagm.1994.1002.

95 Tobias Rossmann. *Algorithms for Nilpotent Linear Groups*. PhD thesis, National University of Ireland, Galway, 2011.

96 Rudolf Scharlau. Pairs of alternating forms. *Mathematische Zeitschrift*, 147(1):13–19, 1976.

97 Jacob T. Schwartz. Fast Probabilistic Algorithms for Verification of Polynomial Identities. *J. ACM*, 27(4):701–717, 1980. doi:10.1145/322217.322225.

98 John Sheekey. *On rank problems for subspaces of matrices over finite fields*. PhD thesis, University College Dublin, 2011.

99 Richard Sinkhorn. A relationship between arbitrary positive matrices and doubly stochastic matrices. *The annals of mathematical statistics*, 35(2):876–879, 1964.

100 Ola Svensson and Jakub Tarnawski. The Matching Problem in General Graphs Is in Quasi-NC. In Chris Umans, editor, *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 696–707. IEEE Computer Society, 2017. doi:10.1109/FOCS.2017.70.

101 John Sylvester. On the dimension of spaces of linear transformations satisfying rank conditions. *Linear Algebra and its Applications*, 78:1–10, 1986.

102 Francesco Ticozzi and Lorenza Viola. Quantum Markovian subsystems: invariance, attractivity, and control. *IEEE Transactions on Automatic Control*, 53(9):2048–2063, 2008.

103 Shuji Tsukiyama, Mikio Ide, Hiromu Ariyoshi, and Isao Shirakawa. A new algorithm for generating all the maximal independent sets. *SIAM Journal on Computing*, 6(3):505–517, 1977.

104 Paul Turán. Egy gráfelméleti szélsőértékfeladatról (On an extremal problem in graph theory). *Mat. Fiz. Lapok*, 48:436–452, 1941.

105 William T. Tutte. The factorization of linear graphs. *Journal of the London Mathematical Society*, 1(2):107–111, 1947.

106 Leslie G. Valiant. Completeness Classes in Algebra. In *Proceedings of the 11h Annual ACM Symposium on Theory of Computing, April 30 - May 2, 1979, Atlanta, Georgia, USA*, pages 249–261, 1979. doi:10.1145/800135.804419.

107 Christiaan van de Woestijne. Deterministic equation solving over finite fields. In *Proceedings of the 2005 international symposium on Symbolic and algebraic computation*, pages 348–353. ACM, 2005.

108 Vincent Vatter. Maximal independent sets and separating covers. *The American Mathematical Monthly*, 118(5):418–423, 2011.

109 EP Vdovin. The number of subgroups with trivial unipotent radicals in finite groups of Lie type. *Journal of Group Theory*, 7(1):99–112, 2004.

110 R. Westwick. Spaces of matrices of fixed rank. *Linear and Multilinear Algebra*, 20(2):171–174, 1987.

111 James B. Wilson. Decomposing p-groups via Jordan algebras. *Journal of Algebra*, 322(8):2642–2679, 2009.

112 James B. Wilson. Finding central decompositions of p-groups. *Journal of Group Theory*, 12(6):813–830, 2009.

113 James B. Wilson. Optimal algorithms of Gram–Schmidt type. *Linear Algebra and its Applications*, 438(12):4573–4583, 2013.

114 Michael M. Wolf. Quantum channels & operations: Guided tour, May 2012. Lecture notes available at https://www-m5.ma.tum.de/foswiki/pub/M5/Allgemeines/MichaelWolf/QChannelLecture.pdf.

115 David Wood. On the number of maximal independent sets in a graph. *Discrete Mathematics and Theoretical Computer Science*, 13(3):17, 2011.

116 Jay A Wood. Spinor groups and algebraic coding theory. *Journal of Combinatorial Theory, Series A*, 51(2):277–313, 1989.

117 Richard Zippel. Probabilistic algorithms for sparse polynomials. In Edward W. Ng, editor, *Symbolic and Algebraic Computation, EUROSAM '79, An International Symposiumon Symbolic and Algebraic Computation, Marseille, France, June 1979, Proceedings*, volume 72 of *Lecture Notes in Computer Science*, pages 216–226. Springer, 1979. doi:10.1007/3-540-09519-5_73.

A Breadth-first search in the alternating matrix space setting

Indeed, suppose $\mathcal{A} \leq \Lambda(n, \mathbb{F})$ admits an isotropic 2-decomposition as $\mathbb{F}^n = U_1 \oplus U_2$. Note that U_1 and U_2 are not known to us. To follow the idea of breadth-first search, we would start from a vector $v \in \mathbb{F}^n$, and then find its neighbours, and then its neighbours' neighbours, etc.. Intuitively, for $v \in \mathbb{F}^n$, we can view the linear span of Av, $A \in \mathcal{A}$ as those neighbours of v, denoted by $V_1 \leq \mathbb{F}^n$. Then the linear span of AV_1, $A \in \mathcal{A}$, may be considered as the neighbours of V_1. Continuing this way, if $v \in U_1$, we do see that V_i's alternates between subspaces of U_1 and U_2. It follows that $V_i \cap V_{i+1} = \mathbf{0}$, from which we can compute U_1 and U_2 after this sequence stabilizes. However, if v is neither in U_1 nor in U_2, it is not clear how to read any information about U_1 and U_2. In fact, it is possible that the linear span of Av is the hyperplane orthogonal to v, so it is impossible to tell whether such U_1 and U_2 exist.

B The relation between alternating bilinear maps and alternating matrix spaces

We first recall the relation between alternating bilinear maps and alternating matrix tuples. Let $\phi : U \times U \to V$ be an alternating bilinear map, that is, for any $u \in U$, $\phi(u, u) = \mathbf{0}$. Fix bases of U and V, so that $U \cong \mathbb{F}^n$ and $V \cong \mathbb{F}^m$. Then ϕ can be represented by an m-tuple of alternating matrices $(A_1, \ldots, A_m) \in \Lambda(n, \mathbb{F})^m$, such that $\phi(u, u') = (u^t A_1 u', \ldots, u^t A_m u')^t$. Conversely, given an m-tuple of alternating matrices, one can define an alternating bilinear map as such. Two alternating bilinear maps $\phi, \psi : U \times U \to V$ are isometric, if there exist $A \in \mathrm{GL}(U)$, $B \in \mathrm{GL}(V)$, such that $\phi = B \circ \psi \circ A$. (Some authors prefer to call this isometric as pseudo-isometric [21].)

Let $\mathcal{A} \leq \Lambda(n, \mathbb{F})$. Let $\mathbf{A} = (A_1, \ldots, A_m) \in \Lambda(n, \mathbb{F})^m$ be an ordered basis of $\mathcal{A}$. Then $\mathbf{A}$ defines an alternating bilinear map $\phi_{\mathbf{A}} : \mathbb{F}^n \times \mathbb{F}^n \to \mathbb{F}^m$ as above. While difference choices of ordered bases give different alternating bilinear maps, it is easy to see that ordered bases from isometric alternating matrix spaces give isometric alternating bilinear maps.

Hard Properties with (Very) Short PCPPs and Their Applications

Omri Ben-Eliezer
Tel Aviv University, Israel
omrib@mail.tau.ac.il

Eldar Fischer
Technion – Israel Institute of Technology, Haifa, Israel
eldar@cs.technion.ac.il

Amit Levi
University of Waterloo, Canada
amit.levi@uwaterloo.ca

Ron D. Rothblum
Technion – Israel Institute of Technology, Haifa, Israel
rothblum@cs.technion.ac.il

Abstract

We show that there exist properties that are maximally hard for testing, while still admitting PCPPs with a proof size very close to linear. Specifically, for every fixed ℓ, we construct a property $\mathcal{P}^{(\ell)} \subseteq \{0,1\}^n$ satisfying the following: Any testing algorithm for $\mathcal{P}^{(\ell)}$ requires $\Omega(n)$ many queries, and yet $\mathcal{P}^{(\ell)}$ has a constant query PCPP whose proof size is $O(n \cdot \log^{(\ell)} n)$, where $\log^{(\ell)}$ denotes the ℓ times iterated log function (e.g., $\log^{(2)} n = \log \log n$). The best previously known upper bound on the PCPP proof size for a maximally hard to test property was $O(n \cdot \text{polylog } n)$.

As an immediate application, we obtain stronger separations between the standard testing model and both the tolerant testing model and the erasure-resilient testing model: for every fixed ℓ, we construct a property that has a constant-query tester, but requires $\Omega(n/\log^{(\ell)}(n))$ queries for every tolerant or erasure-resilient tester.

2012 ACM Subject Classification Theory of computation → Interactive proof systems

Keywords and phrases PCPP, Property testing, Tolerant testing, Erasure resilient testing, Randomized encoding, Coding theory

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.9

Related Version `https://arxiv.org/abs/1909.03255`

Funding *Amit Levi*: Supported by the David R. Cheriton Graduate Scholarship. Part of this work was done while the author was visiting the Technion.
Ron D. Rothblum: Supported in part by the Israeli Science Foundation (Grant No. 1262/18), a Milgrom family grant and the Technion Hiroshi Fujiwara cyber security research center and the Israel cyber directorate.

1 Introduction

Probabilistically checkable proofs (PCPs) are one of the landmark achievements in theoretical computer science. Loosely speaking, PCPs are proofs that can be verified by reading only a very small (i.e., constant) number of bits. Beyond the construction of highly efficient proof systems, PCPs have myriad applications, most notably within the field of hardness of approximation.

A closely related variant of PCPs, called *probabilistically checkable proofs of proximity* (PCPPs), was introduced independently by Ben-Sasson *et al.* [5] and Dinur and Reingold [13]. In the PCPP setting, a verifier is given oracle access to both an input x and a proof π. It should make a few (e.g., constant) number of queries to both oracles to ascertain whether $x \in \mathcal{L}$. Since the verifier can only read a few of the input bits, we only require that it rejects inputs that are *far* (in Hamming distance) from $\mathcal{L}$, no matter what proof π is provided. PCPPs are highly instrumental in the construction of standard PCPs. Indeed, using modern terminology, both the original algebraic construction of PCPs [1] (see also [5]) as well as Dinur's [12] combinatorial proof utilize PCPPs.

By combining the seminal works of Ben-Sasson and Sudan [7] and Dinur [12], one can obtain PCPs and PCPPs with only poly-logarithmic (multiplicative) overhead. More specifically, the usual benchmark for PCPPs is with respect to the `CircuitEval` problem, in which the verifier is given explicit access to a circuit C and oracle access to both an input x and a proof π, and needs to verify that x is close to the set $\{x' : C(x') = 1\}$. The works of [7, 12] yield a PCPP whose length is quasilinear in the size $|C|$ of the circuit C.[1]

Given the important connections both to constructions of efficient proof-systems, and to hardness of approximation, a central question in the area is whether this result can be improved: Do PCPPs with only a *constant* overhead exist? In a recent work, Ben Sasson *et al.* [6] construct PCPs with constant overhead, albeit with very large query complexity (as well as a non-uniform verification procedure).[2] To verify that $C(x) = 1$ the verifier needs to make $|C|^{\delta}$ queries, where $\delta > 0$ can be any fixed constant.

Given the lack of success (despite the significant interest) in constructing constant-query PCPPs with constant overhead, it may be the case that there exist languages that do not have such efficient PCPPs. A natural class of candidate languages for which such PCPPs may not exist are languages for which it is *maximally* hard to test whether $x \in \mathcal{L}$ or is far from such, *without* a PCPP proof. In other words, languages (or rather properties) that do not admit sub-linear query testers. Thus, we investigate the following question:

> *Supposing that $\mathcal{L}$ requires $\Omega(n)$ queries for every (property) tester, must any constant-query* PCPP *for $\mathcal{L}$ have proof length $n \cdot (\log n)^{\Omega(1)}$?*

1.1 Our Results

Our first main result answers the above question negatively, by constructing a property that is maximally hard for testing, while admitting a very short PCPP. For the exact theorem statement, we let $\log^{(\ell)}$ denote the ℓ times iterated log function. That is, $\log^{(\ell)}(n) = \log(\log^{(\ell-1)}(n))$ for $\ell \geq 1$ and $\log^{(0)} n = n$.

▶ **Theorem 1** (informal restatement of Theorem 30). *For every constant integer $\ell \in \mathbb{N}$, there exists a property $\mathcal{P} \subseteq \{0,1\}^n$ such that any testing algorithm for $\mathcal{P}$ requires $\Omega(n)$ many queries, while $\mathcal{P}$ admits a (constant query[3]) PCPP system with proof length $O(n \cdot \log^{(\ell)}(n))$.*

---

[1] Note that a PCPP for `CircuitEval` can be easily used to construct a PCP for `CircuitSAT` with similar overhead (see [5, Proposition 2.4]).

[2] Although it is not stated in [6], we believe that their techniques can also yield PCPPs with similar parameters.

[3] For detection radius (or proximity parameter) $\varepsilon > 0$ and constant soundness, the particular query complexity of the PCPP system is bounded by $(2^{\ell}/\varepsilon)^{O(\ell)}$.

We remark that all such maximally hard properties cannot have constant-query PCPP proof-systems with a *sub-linear* length proof string (see Proposition 13), leaving only a small gap of $\log^{(\ell)}(n)$ on the proof length in Theorem 1.

Beyond demonstrating that PCPPs with extremely short proofs exist for some hard properties, we use Theorem 1 to derive several applications. We proceed to describe these applications next.

Tolerant Testing

Recall that property testing (very much like PCPPs) deals with solving approximate decision problems. A tester for a property $\mathcal{P}$ is an algorithm that given a sublinear number of queries to its input x, should accept (with high probability) if $x \in \mathcal{P}$ and reject if x is *far* from $\mathcal{P}$ (where, unlike PCPPs, the tester is not provided with any proof).

The standard setting of property testing is arguably fragile, since the testing algorithm is only guaranteed to accept all functions that exactly satisfy the property. In various settings and applications, accepting only inputs that exactly have a certain property is too restrictive, and it is more beneficial to distinguish between inputs that are close to having the property, and those that are far from it. To address this question, Parnas, Ron and Rubinfeld [26] introduced a natural generalization of property testing, in which the algorithm is required to accept functions that are close to the property. Namely, for parameters $0 \leq \varepsilon_0 < \varepsilon_1 \leq 1$, an $(\varepsilon_0, \varepsilon_1)$-*tolerant testing algorithm* is given an oracle access to the input, and is required to determine (with high probability) whether a given input is ε_0-close to the property or whether it is ε_1-far from it. As observed in [26], any standard testing algorithm whose queries are uniformly (but not necessarily independently) distributed, is inherently tolerant to some extent. Nevertheless, for many problems, strengthening the tolerance requires applying advanced methods and devising new algorithms (see e.g., [16, 23, 10, 8, 9]).

It is natural to ask whether tolerant testing is strictly harder than standard testing. This question was explicitly studied by Fischer and Fortnow [15], who used PCPPs with polynomial size proofs to show that there exists a property $\mathcal{P} \subseteq \{0,1\}^n$ that admits a tester with constant query complexity, but such that every tolerant tester for $\mathcal{P}$ has query complexity $\Omega(n^c)$ for some $0 < c < 1$. Using modern quasilinear PCPPs [7, 12] in combination with the techniques of [15] it is possible to construct a property demonstrating a better separation, of constant query complexity for standard testing versus $\Omega(n/\text{polylog}\, n)$ for tolerant testing.

Using Theorem 1 we can obtain an improved separation between testing and tolerant testing:

▶ **Theorem 2** (informal restatement of Theorem 46). *For any constant integer $\ell \in \mathbb{N}$, there exist a property of boolean strings $\mathcal{P} \subseteq \{0,1\}^n$ and a constant $\varepsilon_1 \in (0,1)$ such that $\mathcal{P}$ is ε-testable for any $\varepsilon > 0$ with a number of queries[4] independent of n, but for any $\varepsilon_0 \in (0, \varepsilon_1)$, every $(\varepsilon_0, \varepsilon_1)$-tolerant tester for $\mathcal{P}$ requires $\Omega(n/\text{polylog}^{(\ell)} n)$ many queries.*

Erasure-Resilient Testing

Another variant of the property testing model is the *erasure-resilient testing* model. This model was defined by Dixit et. al. [14] to address cases where data cannot be accessed at some domain points due to privacy concerns, or when some of the values were adversarially erased. More precisely, an α-erasure-resilient ε-tester gets as input parameters $\alpha, \varepsilon \in (0,1)$,

[4] The query complexity of the intolerant ε-tester has the same asymptotic bound as in Theorem 1.

as well as oracle access to a function f, such that at most an α fraction of its values have been erased. The tester has to accept with high probability if there is a way to assign values to the erased points of f such that the resulting function satisfies the desired property. The tester has to reject with high probability if for every assignment of values to the erased points, the resulting function is still ε-far from the desired property.

Similarly to the tolerant testing scenario, PCPPs were also used in [14] to show that there exists a property of boolean strings of length n that has a tester with query complexity independent of n, but for any constant $\alpha > 0$, every α-erasure-resilient tester is required to query $\Omega(n^c)$ many bits for some $c > 0$, thereby establishing a separation between the models. Later, in [27] PCPP constructions were used to provide a separation between the erasure-resilient testing model and the tolerant testing model.

Similarly to the tolerant testing case, we use Theorem 1 to prove a stronger separation between the erasure-resilient testing model and the standard testing model.

▶ **Theorem 3** (informal restatement of Theorem 47). *For any constant integer $\ell \in \mathbb{N}$, there exist a property of boolean strings $\mathcal{P} \subseteq \{0,1\}^n$ and a constant $\varepsilon_1 \in (0,1)$ such that $\mathcal{P}$ is ε-testable for any $\varepsilon > 0$ with number of queries[5] independent of n, but for any $\alpha = \Omega(1/\log^{(\ell)} n)$ and $\varepsilon \in (0, \varepsilon_1)$ such that $\varepsilon < 1 - \alpha$, any α-erasure-resilient ε-tester is required to query $\Omega(n/\text{polylog}^{(\ell)} n)$ many bits.*

Secret Sharing applications

As an additional application of our techniques we also obtain a new type of *secret sharing scheme*. Recall that in a secret sharing scheme, a secret value $b \in \{0,1\}$ is shared between n parties in such a way that only an authorized subset of the users can recover the secret. We construct a secret sharing scheme in which no subset of $o(n)$ parties can recover the secret and yet it is possible for each one of the parties to recover the secret, if given access to a PCPP-like proof, with the guarantee that no matter what proof-string is given, most parties will either recover b or reject.

We obtain such a secret sharing scheme through a notion called *Probabilistically Checkable Unveiling of a Shared Secret (*PCUSS*)*, which will be central in our work. This notion is loosely described in Subsection 1.2 and formally defined in Section 4.

1.2 Techniques

Central to our construction are (univariate) polynomials over a finite field $\mathbb{F}$. A basic fact is that a random polynomial $p : \mathbb{F} \to \mathbb{F}$ of degree (say) $|\mathbb{F}|/2$, evaluated at any set of at most $|\mathbb{F}|/2$ points, looks exactly the same as a totally random function $f : \mathbb{F} \to \mathbb{F}$. This is despite the fact that a random function is very far (in Hamming distance) from the set of low degree polynomials. Indeed, this is the basic fact utilized by Shamir's secret sharing scheme [29].

Thus, the property of being a degree-$|\mathbb{F}|/2$ univariate polynomial is a hard problem to decide for any tester, in the sense that such a tester must make $\Omega(|\mathbb{F}|)$ queries to the truth table of the function in order to decide. Given that, it seems natural to start with this property in order to prove Theorem 1. Here we run into two difficulties. First, the property of being a low degree polynomial is defined over a large alphabet, whereas we seek a property over boolean strings. Second, the best known PCPPs for this property have quasi-linear length [7], which falls short of our goal.

[5] Again, the query complexity of the (non erasure resilient) ε-tester has the same asymptotic bound as in Theorem 1.

To cope with these difficulties, our approach is to use composition, or more accurately, an iterated construction. The main technical contribution of this paper lies in the mechanism enabling this iteration. More specifically, rather than having the property contain the explicit truth table of the low degree polynomial p, we would like to use a more redundant representation for encoding each value $p(\alpha)$. This encoding should have several properties:

- It must be the case that one needs to read (almost) the entire encoding to be able to decode $p(\alpha)$. This feature of the encoding, which we view as a secret-sharing type of property, lets us obtain a hard to test property over boolean strings.
- The encoding need not be efficient, and in fact it will be made long enough to eventually subsume the typical length of a PCPP proof-string for the low degree property, when calculated with respect to an *unencoded* input string.
- Last but not least, we need the value to be decodable using very few queries, when given access to an auxiliary PCP-like proof string. This would allow us to "propagate" the PCPP verification of the property across iterations.

In more detail, we would like to devise a (randomized) encoding of strings in $\{0,1\}^k$ by strings in $\{0,1\}^m$. The third requirement listed above can be interpreted as saying that given oracle access to $v \in \{0,1\}^m$ and explicit access to a value $w \in \{0,1\}^k$, it will be possible verify that v indeed encodes w using a PCPP-like scheme, i.e. by providing a proof that can be verified with a constant number of queries. We refer to this property as a *probabilistically checkable unveiling (*PCU*)*[6]. Note that in our setting a single value w may (and usually will) have more than one valid encoding.

Going back to the first requirement of the encoding, we demand that without a proof, one must query at least $\Theta(m)$ bits of v to obtain *any* information about the encoded w, or even discern that v is indeed a valid encoding of some value. Given this combination of requirements, we refer to the verification procedure as a *Probabilistically Checkable Unveiling of a Shared Secret (*PCUSS*)*.

Low degree polynomials can be used to obtain a PCUSS based on Shamir's secret sharing scheme. More specifically, to encode a k bit string w, we take a random polynomial whose values on a subset $H \subseteq \mathbb{F}$ are exactly equal to the bits of w. However, we provide the values of this polynomial only over the sub domain $\mathbb{F} \setminus H$. Then, the encoded value is represented by the (interpolated) values of g over H, which admit a PCU scheme. On the other hand, the "large independence" feature of polynomials makes the encoded value indiscernible without a a supplied proof string, unless too many of the values of g over $\mathbb{F} \setminus H$ are read, thus allowing for a PCUSS.

This construction can now be improved via iteration. Rather than explicitly providing the values of the polynomial, they will be provided by a PCUSS scheme. Note that the PCUSS scheme that we now need is for strings of a (roughly) exponentially smaller size. The high level idea is to iterate this construction ℓ times to obtain the ℓ iterated log function in our theorems.

At the end of the recursion, i.e., for the smallest blocks at the bottom, we utilize a linear-code featuring both high distance and high dual distance, for a polynomial size PCUSS of the encoded value. This is the only "non-constructive" part in our construction, but since the relevant block size will eventually be less than $\log\log(n)$, the constructed property will still be uniform with polynomial calculation time (the exponential time in $\text{poly}(\log\log(n))$, needed to construct the linear-code matrix, becomes negligible).

[6] In fact, we will use a stronger variant where the access to w is also restricted.

Our PCUSS in particular provides a property that is hard to test (due to its shared secret feature), and yet has a near-linear PCPP through its unveiling, thereby establishing Theorem 1. We utilize this property for separation results in a similar manner to [15] and [14], by considering a weighted version of a "PCPP with proof" property, where the proof part holds only a small portion of the total weight. The PCPP proof part enables a constant query test, whereas if the PCPP proof is deleted, efficient testing is no longer possible.

1.3 Related work

Short PCPPs

For properties which can be verified using a circuit of size n, [5] gave PCPP constructions with proof length $n \cdot \exp(\text{poly}(\log\log n))$ and with a query complexity of $\text{poly}(\log\log n)$, as well as slightly longer proofs with constant query complexity. Later, Ben-Sasson and Sudan [7] gave constructions with quasilinear size proofs, but with slightly higher query complexity. The state of the art construction is due to Dinur [12] who, building on [7], showed a PCPP construction with proof length that is quasilinear in the circuit size and with constant query complexity. In a recent work Ben Sasson *et al.* [3] constructed an *interactive* version of PCPPs [4, 28] of strictly linear length and constant query complexity.

Tolerant Testing

The tolerant testing framework has received significant attention in the past decade. Property testing of dense graphs, initiated by [19], is inherently tolerant by the canonical tests of Goldreich and Trevisan [21]. Later, Fischer and Newman [16] (see also [2]) showed that every testable (dense) graph property admits a tolerant testing algorithm for *every* $0 < \varepsilon_0 < \varepsilon_1 < 1$, which implies that $O(1)$ query complexity testability is equivalent to distance approximation in the dense graph model. Some properties of boolean functions were also studied recently in the tolerant testing setting. In particular, the properties of being a *k-junta* (i.e. a function that depends on k variables) and being *unate* (i.e., a function where each direction is either monotone increasing or monotone decreasing) [9, 24, 11].

Erasure-resilient Testing

For the erasure resilient model, in addition to the separation between that model and the standard testing model, [14] designed efficient erasure-resilient testers for important properties, such as monotonicity and convexity. Shortly after, in [27] a separation between the erasure-resilient testing model and the tolerant testing model was established. The last separation requires an additional construction (outside PCPPs), which remains an obstacle to obtaining better than polynomial separations.

2 Preliminaries

We start with some notation and central definitions. For a set A, we let 2^A denote the power-set of A. For two strings $x, y \in \{0,1\}^*$ we use $x \sqcup y$ to denote string concatenation.

For an integer k, a field $\mathbb{F} = \text{GF}(2^k)$ and $\alpha \in \mathbb{F}$, we let $\langle\langle \alpha \rangle\rangle \in \{0,1\}^k$ denote the binary representation of α in some canonical way.

For two sets of strings A and B we use $A \sqcup B$ to denote the set $\{a \sqcup b \mid a \in A,\ b \in B\}$. For a collection of sets $\{A(d)\}_{d \in D}$ we use $\bigsqcup_{d \in D} A(d)$ to denote the set of all possible concatenations $\bigsqcup_{d \in D} a_d$, where $a_d \in A(d)$ for every $d \in D$.

Throughout this paper we use boldface letters to denote random variables, and assume a fixed canonical ordering over the elements in all the sets we define. For a set D, we write $\boldsymbol{v} \sim D$ to denote a random variable resulting from a uniformly random choice of an element $v \in D$.

2.1 Error correcting codes and polynomials over finite fields

The relative Hamming distance of two strings $x, y \in \Sigma^n$ is defined as $\mathrm{dist}(x,y) = \frac{1}{n} \cdot |\{i \in [n] \mid x_i \neq y_i\}|$. For a string $x \in \Sigma^n$ and a non-empty set $S \subseteq \Sigma^n$, we define $\mathrm{dist}(x,S) = \min_{y \in S} \mathrm{dist}(x,y)$. The following plays a central role in many complexity-related works, including ours.

▶ **Definition 4.** *A* code *is an injective function $C : \Sigma^k \to \Sigma^n$. If Σ is a finite field and C is a linear function (over Σ), then we say that C is a* linear code. *The* rate *of C is defined as k/n, whereas the* minimum relative distance *is defined as the minimum over all distinct $x, y \in \Sigma^k$ of* $\mathrm{dist}(C(x), C(y))$.

*An ε-*distance code *is a code whose minimum relative distance is at least ε. When for a fixed $\varepsilon > 0$ we have a family of ε-distance codes (for different values of k), we refer to its members as* error correcting codes.

In this work we use the fact that efficient codes with constant rate and constant relative distance exist. Moreover, there exist such codes in which membership can be decided by a quasi-linear size Boolean circuit.

▶ **Theorem 5** (see e.g., [30]). *There exists a linear code $\mathsf{Spiel} : \{0,1\}^k \to \{0,1\}^{100k}$ with constant relative distance, for which membership can be decided by a $k \cdot \mathrm{polylog}\, k$ size Boolean circuit.*

Actually, the rate of the code in [30] is significantly better, but since we do not try to optimize constants, we use the constant 100 solely for readability. In addition, the code described in [30] is *linear time decodeable*, but we do not make use of this feature throughout this work.

We slightly abuse notation, and for a finite field $\mathbb{F}$ of size 2^k, view the encoding given in Theorem 5 as $\mathsf{Spiel} : \mathbb{F} \to \{0,1\}^{100k}$, by associating $\{0,1\}^k$ with $\mathbb{F}$ in the natural way. Note that for $f : \mathbb{F} \to \mathbb{F}$, it holds that $\langle\!\langle f(\beta) \rangle\!\rangle \in \{0,1\}^k$ for every $\beta \in \mathbb{F}$, and therefore $\mathsf{Spiel}(f(\beta)) \in \{0,1\}^{100k}$. We slightly abuse notation, and for a function $f : \mathbb{F} \to \mathbb{F}$ we write $\mathsf{Spiel}(f)$ to denote the length $100k \cdot 2^k$ bit string $\bigsqcup_{\beta \in \mathbb{F}} \mathsf{Spiel}(f(\beta))$ (where we use the canonical ordering over $\mathbb{F}$).

▶ **Definition 6.** *Let $\mathcal{C}_{\mathbb{F}}$ denote the set of polynomials $g : \mathbb{F} \to \mathbb{F}$ such that $\deg(g) \leq \frac{|\mathbb{F}|}{2}$.*

The following lemma of [22], providing a fast univariate interpolation, will be an important tool in this work.

▶ **Lemma 7** ([22]). *Given a set of pairs $\{(x_1, y_1), \ldots, (x_r, y_r)\}$ with all x_i distinct, we can output the coefficients of $p(x) \in \mathbb{F}[X]$ of degree at most $r-1$ satisfying $p(x_i) = y_i$ for all $i \in [r]$, in $O(r \cdot \log^3(r))$ additions and multiplications in $\mathbb{F}$.*

The next lemma states that a randomly chosen function $\boldsymbol{\lambda} : \mathbb{F} \to \mathbb{F}$ is far from any low degree polynomial with very high probability.

▶ **Lemma 8.** *With probability at least $1 - o(1)$, a uniformly random function $\boldsymbol{\lambda} : \mathbb{F} \to \mathbb{F}$ is $1/3$-far from $\mathcal{C}_{\mathbb{F}}$.*

Proof. Consider the size of a ball of relative radius $1/3$ around some function $\lambda : \mathbb{F} \to \mathbb{F}$ in the space of functions from $\mathbb{F}$ to itself. The number of points (i.e., functions from $\mathbb{F} \to \mathbb{F}$) contained in this ball is at most

$$\binom{|\mathbb{F}|}{|\mathbb{F}|/3} \cdot |\mathbb{F}|^{|\mathbb{F}|/3} \leq (3e|\mathbb{F}|)^{|\mathbb{F}|/3}.$$

By the fact that the size of $\mathcal{C}_{\mathbb{F}}$ is $|\mathbb{F}|^{|\mathbb{F}|/2+1}$, the size of the set of points that are at relative distance at most $1/3$ from any point in $\mathcal{C}_{\mathbb{F}}$ is at most

$$|\mathbb{F}|^{|\mathbb{F}|/2+1} \cdot (3e|\mathbb{F}|)^{|\mathbb{F}|/3} = o(|\mathbb{F}|^{|\mathbb{F}|}).$$

The lemma follows by observing that there are $|\mathbb{F}|^{|\mathbb{F}|}$ functions from $\mathbb{F}$ to itself. ◀

2.1.1 Dual distance of linear codes

We focus here specifically on a linear code $C : \mathbb{F}^k \to \mathbb{F}^n$, and consider the linear subspace of its image, $V_C = \{C(x) : x \in \mathbb{F}^k\} \subseteq \mathbb{F}^n$. We define the *distance* of a linear space as $\text{dist}(V) = \min_{v \in V \setminus \{0^n\}} \text{dist}(v, 0^n)$, and note that in the case of V being the image V_C of a code C, this is identical to $\text{dist}(C)$. For a linear code, it helps to investigate also *dual distances*.

▶ **Definition 9.** *Given two vectors $u, v \in \mathbb{F}^n$, we define their* scalar product *as $u \cdot v = \sum_{i \in [n]} u_i v_i$, where multiplication and addition are calculated in the field $\mathbb{F}$. Given a linear space $V \subseteq \mathbb{F}^n$, its* dual space *is the linear space $V^\perp = \{u : \forall v \in V, u \cdot v = 0\}$. In other words, it is the space of vectors who are orthogonal to all members of V. The* dual distance *of the space V is simply defined as $\text{dist}(V^\perp)$.*

For a code C, we define its *dual distance*, $\text{dist}^\perp(C)$, as the dual distance of its image V_C. We call C an *η-dual-distance* code if $\text{dist}^\perp(C) \geq \eta$. The following well-known lemma is essential to us, as it will relate to the "secret-sharing" property that we define later.

▶ **Lemma 10** (See e.g., [25, Chapter 1, Theorem 10])**.** *Suppose that $C : \mathbb{F}^k \to \mathbb{F}^n$ is a linear η-dual distance code, let $Q \subset [n]$ be any set of size less than $\eta \cdot n$, and consider the following random process for picking a function $\boldsymbol{u} : Q \to \mathbb{F}$: Let $\boldsymbol{w} \in \mathbb{F}^k$ be drawn uniformly at random, and set $\boldsymbol{u}$ be the restriction of $C(\boldsymbol{w})$ to the set Q. Then, the distribution of $\boldsymbol{u}$ is identical to the uniform distribution over the set of all functions from Q to $\mathbb{F}$.*

2.2 Probabilistically checkable proofs of proximity (PCPP)

As described briefly in the introduction, a PCPP verifier for a property $\mathcal{P}$ is given access to an input x and a proof π, as well as a *detection radius* $\varepsilon > 0$ and *soundness parameter* $\delta > 0$. The verifier should make a constant number of queries (depending only on ε, δ) to the input x and the proof π, and satisfy the following. If $x \in \mathcal{P}$, then there exists π for which the verifier should always accept x. If $\text{dist}(x, \mathcal{P}) > \varepsilon$, the verifier should reject x with probability at least δ, regardless of the contents of π. More formally, we define the following.

▶ **Definition 11** (PCPP)**.** *For $n \in \mathbb{N}$, let $\mathcal{P} \subset \{0,1\}^n$ be a property of n-bit Boolean strings, and let $t \in \mathbb{N}$. We say that $\mathcal{P}$ has a $q(\varepsilon, \delta)$-*query, length-t Probabilistically Checkable Proof of Proximity (PCPP) system *if the following holds: There exists a verification algorithm V that takes as input $\varepsilon, \delta > 0$ and $n \in \mathbb{N}$, makes a total of $q(\varepsilon, \delta)$ queries on strings $w \in \{0,1\}^n$ and $\pi \in \{0,1\}^t$, and satisfies the following:*

1. *(Completeness) If $w \in \mathcal{P}$, then there exists a proof $\pi = \boldsymbol{Proof}_{\mathcal{P}}(w) \in \{0,1\}^t$ such that for every $\varepsilon, \delta > 0$, the verifier V accepts with probability 1.*
2. *(Soundness) If $\mathrm{dist}(w, \mathcal{P}) > \varepsilon$, then for every alleged proof $\pi \in \{0,1\}^t$, the verifier V rejects with probability greater than δ.*

Note that soundness is easy to amplify: Given a PCPP as above with parameters ε, δ, t and query complexity $q(\varepsilon, \delta)$, one can increase the soundness parameter to $1 - \tau$ (for any $\tau > 0$) by simply running $\Theta(\log(1/\tau)/\delta)$ independent instances of the verification algorithm V, and rejecting if at least one of them rejected; the query complexity then becomes $\Theta\left(q(\varepsilon, \delta) \cdot \log(1/\tau)/\delta\right)$, while the parameters ε and t remain unchanged.

The following lemma, establishing the existence of a quasilinear PCPP for any property $\mathcal{P}$ that is verifiable in quasilinear time, will be an important tool throughout this work.

▶ **Lemma 12** (Corollary 8.4 in [12], see also [20]). *Let $\mathcal{P}$ be a property of Boolean strings which is verifiable by a size t Boolean circuit. Then, there exists a length-t' PCPP system $\mathcal{P}$ with parameters $\varepsilon, \delta > 0$, that makes at most $q(\varepsilon, \delta)$ queries, where $t' = t \cdot \mathrm{polylog}\, t$.*

Specifically, $q(\varepsilon, \delta) = O(\varepsilon^{-1})$ suffices for any $\delta < 0.99$.

As described briefly in the introduction, maximally hard properties cannot have a constant query PCPP proof systems with a sublinear length proof string.

▶ **Proposition 13.** *Let $\mathcal{P} \subseteq \{0,1\}^n$ and $\varepsilon > 0$ be such that any ε-tester for $\mathcal{P}$ has to make $\Omega(n)$ many queries. Then, any constant query PCPP system for $\mathcal{P}$ (where e.g. $\delta = 1/3$) must have proof length of size $\Omega(n)$.*

Proof. Suppose that there exists a PCPP for $\mathcal{P}$ with $O(1)$ queries and proof length $t = o(n)$. Since the PCPP verifier has constant query complexity, we may assume that it is non adaptive and uses $q = O(1)$ queries. By an amplification argument as above, we can construct an amplified verifier that makes $O(q \cdot t) = o(n)$ queries, with soundness parameter $1 - 2^{-t}/3$. By the fact that the verifier is non-adaptive, it has the same query distribution regardless of the proof string. Therefore, we can run 2^t amplified verifiers in parallel while reusing queries, one verifier for each of the 2^t possible proof strings. If any of the 2^t amplified verifiers accept, we accept the input. If the input belongs to $\mathcal{P}$, one of the above 2^t verifiers will accept (the one that used the correct proof). If the input was ε-far from $\mathcal{P}$, then by a union bound, the probability that there was any accepting amplified verifier is at most $1/3$. This yields an $o(n)$ tester for $\mathcal{P}$, which contradicts our assumption. ◀

2.3 Testing, tolerant testing and erasure-resilient testing

In this subsection we define notions related to the property testing framework. We also formally define a few variants of the original testing model that will be addressed in this work. A *property* $\mathcal{P}$ of n-bit boolean strings is a subset of all those strings, and we say that a string x *has* the property $\mathcal{P}$ if $x \in \mathcal{P}$.

Given $\varepsilon \geq 0$ and a property $\mathcal{P}$, we say that a string $x \in \{0,1\}^n$ is *ε-far* from $\mathcal{P}$ if $\mathrm{dist}(x, \mathcal{P}) > \varepsilon$, and otherwise it is *$\varepsilon$-close* to $\mathcal{P}$. We next define the notion of a *tolerant* tester of which standard (i.e. intolerant) testers are a special case.

▶ **Definition 14** (Intolerant and tolerant testing). *Given $0 \leq \varepsilon_0 < \varepsilon_1 \leq 1$, a q-query $(\varepsilon_0, \varepsilon_1)$-*testing *algorithm T for a property $\mathcal{P} \subseteq \{0,1\}^n$ is a probabilistic algorithm (possibly adaptive) making q queries to an input $x \in \{0,1\}^n$ that outputs a binary verdict satisfying the following two conditions.*

1. *If* $\text{dist}(x, \mathcal{P}) \leq \varepsilon_0$*, then* T *accepts* x *with probability at least* $2/3$.
2. *If* $\text{dist}(x, \mathcal{P}) > \varepsilon_1$*, then* T *rejects* x *with probability at least* $2/3$.

When $\varepsilon_0 = 0$*, we say that* T *is an* ε_1-testing algorithm *for* $\mathcal{P}$*, and otherwise we say that* T *is an* $(\varepsilon_0, \varepsilon_1)$-tolerant testing algorithm *for* $\mathcal{P}$.

Next, we define the erasure-resilient testing model. We start with some terminology. A string $x \in \{0,1,\perp\}^n$ is α-erased if x_i is equal to $\perp$ on at most αn coordinates. A string $x' \in \{0,1\}^n$ that differs from x only on coordinates $i \in [n]$ for which $x_i = \perp$ is called a *completion* of x. The (pseudo-)distance $\text{dist}(x, \mathcal{P})$ of an α-erased string x from a property $\mathcal{P}$ is the minimum, over every completion x' of x, of the relative Hamming distance of x' from $\mathcal{P}$. Note that for a string with no erasures, this is simply the Hamming distance of x from $\mathcal{P}$. As before, x is ε-far from $\mathcal{P}$ if $\text{dist}(x, \mathcal{P}) > \varepsilon$, and ε-close otherwise.

▶ **Definition 15** (Erasure-resilient tester)**.** *Let* $\alpha \in [0,1)$ *and* $\varepsilon \in (0,1)$ *be parameters satisfying* $\alpha + \varepsilon < 1$*. A* q*-query* α*-erasure-resilient* ε*-tester* T *for* $\mathcal{P}$ *is a probabilistic algorithm making* q *queries to an* α*-erased string* $x \in \{0,1,\perp\}^n$*, that outputs a binary verdict satisfying the following two conditions.*

1. *If* $\text{dist}(x, \mathcal{P}) = 0$ *(i.e., if there exists a completion* x' *of* x*, such that* $x' \in \mathcal{P}$*), then* T *accepts* x *with probability at least* $2/3$.
2. *If* $\text{dist}(x, \mathcal{P}) > \varepsilon$ *(i.e., if every completion of* x' *of* x *is* ε*-far from* $\mathcal{P}$*), then* T *rejects* x *with probability at least* $2/3$.

The next lemma will be useful to prove that some properties are hard to test. The lemma states that if we have two distributions whose restrictions to any set of queries of size at most q are identical, then no (possibly adaptive) algorithm making at most q queries can distinguish between them.

▶ **Definition 16** (Restriction)**.** *Given a distribution* $\mathcal{D}$ *over functions* $f : D \to \{0,1\}$ *and a subset* $Q \subseteq D$*, we define the* restriction $\mathcal{D}|_Q$ of $\mathcal{D}$ to Q *to be the distribution over functions* $g : Q \to \{0,1\}$*, that results from choosing a function* $f : D \to \{0,1\}$ *according to* $\mathcal{D}$*, and setting* g *to be* $f|_Q$*, the restriction of* f *to* Q.

▶ **Lemma 17** ([17], special case)**.** *Let* $\mathcal{D}_1$ *and* $\mathcal{D}_2$ *be two distributions of functions over some domain* D*. Suppose that for any set* $Q \subset D$ *of size at most* q*, the restricted distributions* $\mathcal{D}_1|_Q$ *and* $\mathcal{D}_2|_Q$ *are identically distributed. Then, any (possibly adaptive) algorithm making at most* q *queries cannot distinguish* $\mathcal{D}_1$ *from* $\mathcal{D}_2$ *with any positive probability.*

3 Code Ensembles

It will be necessary for us to think of a generalized definition of an encoding, in which each encoded value has multiple legal encodings.

▶ **Definition 18** (Code ensemble)**.** *A* code ensemble *is a function* $\mathcal{E} : \Sigma^k \to 2^{\Sigma^m}$*. Namely, every* $x \in \Sigma^k$ *has a set of its valid encodings from* Σ^m*. We define the distance of the code ensemble as*

$$\min_{x \neq x' \in \{0,1\}^k} \min_{(v,u) \in \mathcal{E}(x) \times \mathcal{E}(x')} \text{dist}(v, u).$$

It is useful to think of a code ensemble $\mathcal{E} : \Sigma^k \to 2^{\Sigma^m}$ as a *randomized mapping*, that given $x \in \Sigma^k$, outputs a uniformly random element from the set of encodings $\mathcal{E}(x)$. Using the above we can define a *shared secret* property. In particular, we use a strong information

theoretic definition of a shared secret, in which $o(m)$ bits do not give *any information at all* about the encoded value. Later on, we construct code ensembles with a shared secret property.

▶ **Definition 19** (Shared Secret). *For $m, k \in \mathbb{N}$ and a constant $\zeta > 0$, we say that a code ensemble $\mathcal{C} : \{0,1\}^k \to 2^{(\{0,1\}^m)}$ has a* ζ-shared secret *property if it satisfies the following. For any $Q \subseteq [m]$ of size $|Q| \leq \zeta m$, any $w, w' \in \{0,1\}^k$ such that $w \neq w'$, and any $t \in \{0,1\}^{|Q|}$ it holds that*

$$\underset{\boldsymbol{v} \sim \mathcal{C}(w)}{\boldsymbol{Pr}}[\boldsymbol{v}|_Q = t] = \underset{\boldsymbol{v}' \sim \mathcal{C}(w')}{\boldsymbol{Pr}}[\boldsymbol{v}'|_Q = t].$$

Namely, for any $w \neq w'$ and any $Q \subseteq [m]$ of size at most ζm, the distribution obtained by choosing a uniformly random member of $\mathcal{C}(w)$ and considering its restriction to Q, is identical to the distribution obtained by choosing a uniformly random member of $\mathcal{C}(w')$ and considering its restriction to Q.

3.1 A construction of a hard code ensemble

We describe a construction of a code ensemble for which a linear number of queries is necessary to verify membership or to decode the encoded value. This code will be our *base* code in the iterative construction. The existence of such a code ensemble is proved probabilistically, relying on the following simple lemma.

▶ **Lemma 20.** *Fix constant $\alpha, \beta > 0$ where $\beta \log(e/\beta) < \alpha$. Let $s, t \in \mathbb{N}$ so that $s \leq (1-\alpha)t$. Then, with probability $1 - o(1)$, a sequence of s uniformly random vectors $\{v_1, \ldots, v_s\}$ from $\{0,1\}^t$ is linearly independent, and corresponds to a β-distance linear code.*

Proof. The proof follows from a straightforward counting argument. If we draw s uniformly random vectors $v_1, \ldots, v_s \in \{0,1\}^t$, then each non-trivial linear combination of them is in itself a uniformly random vector from $\{0,1\}^t$, and hence has weight less than β with probability at most

$$2^{-t} \cdot \binom{t}{\beta t} \leq 2^{-t} \left(\frac{et}{\beta t}\right)^{\beta t} = 2^{-t} \cdot 2^{\beta \log(e/\beta) t} = 2^{(\gamma - 1)t},$$

where we set $\gamma = \beta \log(e/\beta) < \alpha$.

By a union bound over all $2^s \leq 2^{(1-\alpha)t}$ possible combinations, the probability that there exists a linear combination with weight less than β is at most $2^{(\gamma-\alpha)t} = o(1)$. If this is not the case, then $v_1, \ldots, v_s$ are linearly independent, and moreover, $\{v_1, \ldots, v_s\}$ corresponds to a β-distance linear code (where we use the fact that the distance of a linear code is equal to the minimal Hamming weight of a non-zero codeword). ◀

Our construction makes use of a sequence of vectors that correspond to a high-distance and high-dual distance code, as described below.

▶ **Definition 21** (Hard code ensemble $\mathcal{H}_k$). *Let $k \in \mathbb{N}$ and let $\{v_1, \ldots, v_{3k}\}$ be a sequence of vectors in $\{0,1\}^{4k}$ such that* $\mathrm{Span}\{v_1, \ldots, v_{3k}\}$ *is a* $1/30$*-distance code, and that* $\mathrm{Span}\{v_{k+1}, \ldots, v_{3k}\}$ *is a* $1/10$*-dual distance code. Let*

$$A = \begin{bmatrix} | & & | \\ v_1 & \cdots & v_{3k} \\ | & & | \end{bmatrix}.$$

We define the code ensemble $\mathcal{H}_k : \{0,1\}^k \to 2^{\{0,1\}^{4k}}$ *as*

$$\mathcal{H}_k(w) = \{Au \ : \ u \in \{0,1\}^{3k} \text{ where } u|_{\{1,\ldots,k\}} = w\},$$

where all operations are over GF(2)*.*

The next lemma states that a collection of random vectors $\{v_1, \ldots, v_{3k}\}$ in $\{0,1\}^{4k}$ satisfies the basic requirements of a code ensemble $\mathcal{H}_k$ with high probability (that is, with probability tending to one as $k \to \infty$), and hence such a code ensemble exists.

▶ **Lemma 22.** *A set* $\{v_1, \ldots, v_{3k}\}$ *of random vectors in* $\{0,1\}^{4k}$ *satisfies with high probability the following two conditions:* $\mathrm{Span}\{v_1, \ldots, v_{3k}\}$ *is a* 1/30*-distance code, and furthermore,* $\mathrm{Span}\{v_{k+1}, \ldots, v_{3k}\}$ *is a* 1/10*-dual distance code. In particular, for all* k *large enough the code ensemble* $\mathcal{H}_k$ *exists.*

Proof. We apply Lemma 20 multiple times. First, picking $t = 4k$, $s = 3k$, $\alpha = 1/4$, and $\beta = 1/30$, we conclude that $v_1, \ldots, v_{3k}$ with high probability correspond to a 1/30-distance code.

To show that with high probability the code spanned by the last $2k$ vectors has high dual distance, we compare the following two processes, whose output is a linear subspace of $(\mathrm{GF}(2))^{4k}$, that we view as a code: (i) Choose $2k$ vectors and return their span. (ii) Choose $4k - 2k = 2k$ vectors and return the dual of their span. Conditioning on the chosen $2k$ vectors being linearly independent, the output distributions of these two processes are identical. Indeed, by a symmetry argument it is not hard to see that under the conditioning, the linear subspace generated by Process (i) is uniformly distributed among all rank-$2k$ subspaces V of $(\mathrm{GF}(2))^{4k}$. Now, since we can uniquely couple each such V with its dual $V^\perp$ (also a rank-$2k$ subspace) and since $V = (V^\perp)^\perp$, this means that the output distribution of Process (ii) is uniform as well.

However, it follows again from Lemma 20 (with $t = 4k$, $s = 2k$, $\alpha = 1/2$, and any $\beta > 0$ satisfying the conditions of the lemma) that the chosen $2k$ vectors are independent with high probability. This means that (without the conditioning) the output distributions of Process (i) and Process (ii) are $o(1)$-close in variation distance. Applying Lemma 20 with $t = 4k$, $s = 2k$, $\alpha = 1/2$, and $\beta = 1/10$ we get that the distance of the code generated by Process (i) is at least $\beta = 1/10$ with high probability. However, the latter distance equals by definition to the dual distance of the code generated by Process (ii). By the closeness of the distributions, we conclude that the dual distance of Process (i) is also at least 1/10 with high probability. ◀

We next state a simple but important observation regarding membership verification.

▶ **Observation 23.** *Once a matrix* A *with the desired properties is constructed (which may take* $\exp(k^2)$ *time if we use brute force), given* $w \in \{0,1\}^k$*, the membership of* v *in* $\mathcal{H}_k(w)$ *can be verified in* $\mathrm{poly}(k)$ *time (by solving a system of linear equations over* GF(2)*).*

4 PCUs and PCUSSs

Next, we define the notion of Probabilistically Checkable Unveiling (PCU). This notion is similar to PCPP, but here instead of requiring our input to satisfy a given property, we require our input to encode a value $w \in \{0,1\}^k$ (typically using a large distance code ensemble). We then require that given the encoded value w, it will be possible to prove in a PCPP-like fashion that the input is indeed a valid encoding of w.

▶ **Definition 24** (PCU). *Fix $m, t, k \in \mathbb{N}$, and let $\mathcal{C} : \{0,1\}^k \to 2^{\{0,1\}^m}$ be a code ensemble. We say that $\mathcal{C}$ has a $q(\varepsilon,\delta)$-query, length-t* PCU *if the following holds. There exists a verification algorithm V that takes as inputs $\varepsilon, \delta > 0$, $m \in \mathbb{N}$, and $w \in \{0,1\}^k$, makes at most $q(\varepsilon,\delta)$ queries to the strings $v \in \{0,1\}^m$ and $\pi \in \{0,1\}^t$, and satisfies the following:*

1. *If $v \in \mathcal{C}(w)$, then there exists a proof $\pi = \boldsymbol{Proof}_{\mathcal{C}}(v) \in \{0,1\}^t$ such that for every $\varepsilon, \delta > 0$, the verifier V accepts with probability 1.*
2. *If $\mathrm{dist}(v, \mathcal{C}(w)) > \varepsilon$, then for every alleged proof $\pi \in \{0,1\}^t$, the verifier V rejects v with probability greater than δ.*

In order to facilitate the proof of the main theorem, we utilize a more stringent variant of the above PCU definition. Instead of supplying $w \in \{0,1\}^k$ to the algorithm, we supply oracle access to a a string $\tau \in \{0,1\}^{100k}$ that is supposed to represent $\mathsf{Spiel}(w)$, along with the proof π, and the algorithm only makes $q(\varepsilon,\delta)$ queries to the proof string π, the original encoding v *and* the string τ. For cases where $v \in \mathcal{C}(w)$, we use $\mathbf{Value}(v)$ to denote $\mathsf{Spiel}(w)$.

▶ **Definition 25** (Spiel-PCU). *Fix $m, t, k \in \mathbb{N}$, and let $\mathcal{C} : \{0,1\}^k \to 2^{\{0,1\}^m}$ be a code ensemble. We say that $\mathcal{C}$ has a $q(\varepsilon,\delta)$-query, length-t* Spiel-PCU *if the following holds. There exists a verification algorithm V that takes as inputs $\varepsilon, \delta > 0$, $m \in \mathbb{N}$, makes at most $q(\varepsilon,\delta)$ queries to the strings $v \in \{0,1\}^m$, $\tau \in \{0,1\}^{100k}$ and $\pi \in \{0,1\}^t$, and satisfies the following:*

1. *If there exists $w \in \{0,1\}^k$ for which $v \in \mathcal{C}(w)$ and $\tau = \boldsymbol{Value}(v) = \mathsf{Spiel}(w)$, then there exists a proof $\pi = \boldsymbol{Proof}_{\mathcal{C}}(v) \in \{0,1\}^t$ such that for every $\varepsilon, \delta > 0$, the verifier V accepts with probability 1.*
2. *If for every $w \in \{0,1\}^k$ either $\mathrm{dist}(\tau, \mathsf{Spiel}(w)) > \varepsilon$ or $\mathrm{dist}(v, \mathcal{C}(w)) > \varepsilon$, then for every alleged proof $\pi \in \{0,1\}^t$, the verifier V rejects v with probability greater than δ.*

Note that a code ensemble admitting a Spiel-PCU automatically admits a PCU. Indeed, given the string w, an oracle for $\mathsf{Spiel}(w)$ can be simulated.

The following lemma states the existence of Spiel-PCU for efficiently computable code ensembles, and will be used throughout this work. The proof follows from Lemma 12 together with a simple concatenation argument.

▶ **Lemma 26.** *Let $k, m, t \in \mathbb{N}$ be such that $t \geq m$, and let $\mathcal{C} : \{0,1\}^k \to 2^{\{0,1\}^m}$ be a code ensemble. If given $w \in \{0,1\}^k$ and $v \in \{0,1\}^m$, it is possible to verify membership of v in $\mathcal{C}(w)$ using a circuit of size t, then there is a $q(\varepsilon,\delta)$-query, length-t'* Spiel-PCU *for $\mathcal{C}$ where $t' = t \cdot \mathrm{polylog}\, t$.*

Proof. Assume without loss of generality that $m \geq |\mathsf{Spiel}(0^k)|$. Let $\xi = \left\lfloor \frac{m}{|\mathsf{Spiel}(0^k)|} \right\rfloor$ (note that $\xi \geq 1$), and define

$$\mathcal{C}_{eq} \stackrel{\text{def}}{=} \left\{ v \sqcup (\mathsf{Spiel}(w))^{\xi} \;\middle|\; \exists w \in \{0,1\}^k \text{ for which } v \in \mathcal{C}(w) \right\},$$

where $(\mathsf{Spiel}(w))^{\xi}$ denotes the ξ-times concatenation of $\mathsf{Spiel}(w)$.

For any string u it is possible to check, using a quasilinear size circuit (see [30]), that the substring that corresponds to the domain of $(\mathsf{Spiel}(w))^{\xi}$ is a ξ-times repetition of $\mathsf{Spiel}(w)$ for some w. After doing so, we decode w using a quasilinear size circuit (as in [30]), and then, by the premise of the lemma, we can verify membership in $\mathcal{C}(w)$ using a circuit of size t. Therefore, membership in $\mathcal{C}_{eq}$ can be decided using a $O(t)$ size boolean circuit, and therefore by Lemma 12 admits a PCPP system whose proof length is quasilinear in t.

Given an input v to Spiel-PCU, let $v' = v \sqcup (\mathsf{Spiel}(w))^{\xi}$ and use the PCPP system for $\mathcal{C}_{eq}$, with detection radius $\varepsilon/3$ and soundness parameter δ, where each query to v' is emulated by a corresponding query to v or $\mathsf{Spiel}(w)$. Note that if $v \in \mathcal{C}(w)$, then $v' \in \mathcal{C}_{eq}$, so the PCPP system for $\mathcal{C}_{eq}$ will accept with probability 1.

Next, suppose that $\mathrm{dist}(v, \mathcal{C}(w)) > \varepsilon$, and observe that this implies that v' is at least $\varepsilon/3$-far from $\mathcal{C}_{eq}$. Thus, by the soundness property of the PCPP for $\mathcal{C}_{eq}$, the verifier rejects with probability at least δ, regardless of the contents of the alleged proof π it is supplied with. ◀

Next we define Probabilistically Checkable Unveiling of a Shared Secret (PCUSS).

▶ **Definition 27.** *For $m, k, t \in \mathbb{N}$, we say that a function $\mathcal{C}\colon \{0,1\}^k \to 2^{(\{0,1\}^n)}$ has a $q(\varepsilon,\delta)$-query, length-t* PCUSS*, if $\mathcal{C}$ has a shared secret property, as well as $\mathcal{C}$ has a $q(\varepsilon,\delta)$-query, length-t* PCU*. Similarly, when $\mathcal{C}$ has a shared secret property (for constant ζ), as well as $\mathcal{C}$ has a $q(\varepsilon,\delta)$-query, length-t* Spiel-PCU*, we say that $\mathcal{C}$ has a $q(\varepsilon,\delta)$-query, length-t* Spiel-PCUSS*.*

Note that $\mathcal{C}$ admitting a Spiel-PCUSS directly implies that it admits a PCUSS with similar parameters.

The following lemma establishes the existence of a Spiel-PCUSS for $\mathcal{H}_k$, where $\mathcal{H}_k$ is the code ensemble from Definition 21.

▶ **Lemma 28.** *For any $k \in \mathbb{N}$, $\mathcal{H}_k$ has a $q(\varepsilon,\delta)$-query, length-t'* Spiel-PCUSS *where $t' = \mathrm{poly}(k)$.*

Proof. By Observation 23, given w, membership in $\mathcal{H}_k(w)$ can be checked in $\mathrm{poly}(k)$ time, which means that there exists a polynomial size circuit that decides membership in $\mathcal{H}_k(w)$. Combining the above with Lemma 26 implies a $q(\varepsilon,\delta)$-query, length-t' Spiel-PCU where $t' = \mathrm{poly}(k)$. By Lemma 10, the large dual distance property of $\mathcal{H}_k$ implies its shared secret property for some constant ζ, which concludes the proof of the lemma. ◀

5 PCUSS construction

In this section we give a construction of code ensembles that admit a PCUSS. First we show that our code ensemble has a PCU with a short proof. Specifically,

▶ **Lemma 29.** *For any fixed $\ell \in \mathbb{N}$ and any $k \in \mathbb{N}$, there exists $n_0(\ell, k)$ and a code ensemble $\mathcal{E}^{(\ell)} : \{0,1\}^k \to 2^{(\{0,1\}^n)}$, such that for all $n > n_0(\ell, k)$, the code ensemble $\mathcal{E}^{(\ell)}$ has a $q(\varepsilon,\delta)$-query length-t* PCU*, for $t = O(n \cdot \mathrm{polylog}^{(\ell)} n)$.*

Later, we prove that our code ensemble has a shared secret property, which implies that it has a PCUSS (which implies Theorem 1, as we shall show).

▶ **Theorem 30.** *For any fixed $\ell \in \mathbb{N}$ and any $k \in \mathbb{N}$, there exists $n_0(\ell, k)$ and a code ensemble $\mathcal{E}^{(\ell)} : \{0,1\}^k \to 2^{(\{0,1\}^n)}$, such that for all $n > n_0(\ell, k)$, the code ensemble $\mathcal{E}^{(\ell)}$ has a $q(\varepsilon,\delta)$-query length-t* PCUSS*, for $t = O(n \cdot \mathrm{polylog}^{(\ell)} n)$.*

Specifically, by the discussion before Lemma 37, for any fixed soundness parameter $0 < \delta < 1$ it suffices to take

$$q(\varepsilon,\delta) \leq \left(2^{\ell}/\varepsilon\right)^{O(\ell)},$$

and for the high soundness regime where $\delta = 1 - \tau$ (and $\tau > 0$ is small), it suffices to have

$$q(\varepsilon,\delta) \leq \left(2^{\ell}/\varepsilon\right)^{O(\ell)} \log(1/\tau).$$

5.1 The iterated construction

Our iterative construction uses polynomials over a binary finite field $\mathrm{GF}(2^t)$. In our proof we will need to be able to implement arithmetic operations over this field efficiently (i.e., in $\mathrm{poly}(t)$ time). This can be easily done given a suitable representation of the field: namely, a degree t irreducible polynomial over $\mathrm{GF}(2)$. It is unclear in general whether such a polynomial can be found in $\mathrm{poly}(t)$ time. Fortunately though, for $t = 2 \cdot 3^r$ where $r \in \mathbb{N}$, it is known that the polynomial $x^t + x^{t/2} + 1$ is irreducible over $\mathrm{GF}(2)$ (see, e.g., [18, Appendix G]). We will therefore restrict our attention to fields of this form. At first glance this seems to give us a property that is defined only on a sparse set of input lengths. However, towards the end of this section, we briefly describe how to bypass this restriction.

We next formally define our iterated construction, starting with the "level-0" construction as a base case. The constants c, d in the definition will be explicitly given in the proof of Lemma 36. Additionally, for any $\ell \in \mathbb{N}$, we shall pick a large enough constant c_ℓ that satisfies several requirements for the "level-ℓ" iteration of the construction.

▶ **Definition 31** (Iterated coding ensemble). *For $k \in \mathbb{N}$ and $w \in \{0,1\}^k$, we define the* base code ensemble of w *(i.e., level-ℓ code ensemble of w for $\ell = 0$) as*

$$\mathcal{E}_k^{(0)}(w) = \mathcal{H}_k(w).$$

Let $c, d \in \mathbb{N}$ be large enough global constants, fix $\ell > 0$, let c_ℓ be large enough, and let $\mathbb{F}$ be a finite field for which $|\mathbb{F}| \geq \max\{c_\ell, c \cdot k\}$.

We define the level-ℓ code ensemble of $w \in \{0,1\}^k$ over $\mathbb{F}$ *as follows. Let $r \in \mathbb{N}$ be the smallest integer such that $(\log|\mathbb{F}|)^d \leq 2^{2 \cdot 3^r}$, set $\mathbb{F}' = \mathrm{GF}\left(2^{2 \cdot 3^r}\right)$ and $k' = \log|\mathbb{F}|$. Note that these satisfy the recursive requirements of a level-$(\ell-1)$ code ensemble provided that c_ℓ is large enough (specifically we require $(\log|\mathbb{F}|)^{d-1} > c$, so that $|\mathbb{F}'| \geq ck'$). Finally, let $H \subseteq \mathbb{F}$ be such that $|H| = k$, and define*

$$\mathcal{E}_{\mathbb{F},k}^{(\ell)}(w) = \bigcup_{g \in \mathcal{C}_\mathbb{F}:\ g|_H = w} \bigsqcup_{\beta \in \mathbb{F} \setminus H} \mathcal{E}_{\mathbb{F}',k'}^{(\ell-1)}(\langle\!\langle g(\beta) \rangle\!\rangle).$$

(Note that for $\ell = 1$ we just use $\mathcal{E}_{\mathbb{F},k}^{(1)}(w) = \bigcup_{g \in \mathcal{C}_\mathbb{F}:\ g|_H = w} \bigsqcup_{\beta \in \mathbb{F} \setminus H} \mathcal{E}_{k'}^{(0)}(\langle\!\langle g(\beta) \rangle\!\rangle)$).

That is, $v \in \mathcal{E}_{\mathbb{F},k}^{(\ell)}(w)$ if there exists a polynomial $g \in \mathcal{C}_\mathbb{F}$ such that $v = \bigsqcup_{\beta \in \mathbb{F} \setminus H} v_\beta$, where $v_\beta \in \mathcal{E}_{\mathbb{F}',k'}^{(\ell-1)}(\langle\!\langle g(\beta) \rangle\!\rangle)$ for every $\beta \in \mathbb{F} \setminus H$ and $g|_H = w$ (where we identify the 0 and 1 elements of $\mathbb{F}$ with 0 and 1 bits respectively). When the context is clear, we sometimes omit the subscripts.

Our choice of the constants c, d, c_ℓ needs to satisfy the following conditions. The constant c is chosen such that H will not be an overly large portion of $\mathbb{F}$ (this requirement is used in Lemma 42). The constant d is needed to subsume the length of PCPP proof string which is part of the construction (this requirement is used in Lemma 36). Finally, the constant c_ℓ needs to be large enough to enable iteration (as explained in Definition 31 itself).

Let $\ell \geq 0$ be some fixed iteration. The following simple observation follows by a simple inductive argument using the definition of the level-ℓ coding ensemble, and in particular that $|\mathbb{F}'| = \mathrm{polylog}\,|\mathbb{F}|$.

▶ **Observation 32.** *For $\ell > 0$, let $n = |\mathbb{F}|$ and $w \in \{0,1\}^k$. If $v \in \mathcal{E}^{(\ell)}(w)$, then $m_\mathbb{F}^{(\ell)} \stackrel{\text{def}}{=} |v| = n \cdot \mathrm{poly}(\log n) \cdot \mathrm{poly}(\log\log n) \cdots \mathrm{poly}(\log^{(\ell)} n)$, where $\log^{(\ell)} n$ is the* log *function iterated ℓ times.*

When the field $\mathbb{F}$ is clear from context, we shall usually write $m^{(\ell)}$ as a shorthand for $m^{(\ell)}_{\mathbb{F}}$. The following lemma, proved in the next subsection, establishes the existence of short length Spiel-PCUs for our code ensembles.

▶ **Lemma 33.** *For any $\ell \geq 0$, the code ensemble $\mathcal{E}^{(\ell)}_{\mathbb{F},k}$ admits a $q(\varepsilon,\delta)$-query, length-t* Spiel-PCU *for $t = O(m^{(\ell)} \cdot \mathrm{polylog}^{(\ell)} m^{(\ell)})$.*

5.2 Proof of Lemma 33

We start by defining the PCU proof string for a given $v \in \mathcal{E}^{(\ell)}_{\mathbb{F},k}(w)$ for some $w \in \{0,1\}^k$.

▶ **Definition 34** (The PCU Proof String)**.** *For $\ell = 0$, let $v \in \mathcal{E}^{(0)}_k(w)$ and $\boldsymbol{Value}^{(0)}(v) = \mathsf{Spiel}(w)$. We define the proof string for v, $\boldsymbol{Proof}^{(0)}(v)$, as the one guaranteed by Lemma 28 (note that the length of $\boldsymbol{Proof}^{(0)}(v)$ is $\mathrm{poly}(k)$).*

For $\ell > 0$, let $g \in \mathcal{C}_{\mathbb{F}}$ and $w \in \{0,1\}^k$ be such that $v \in \bigsqcup_{\beta\in\mathbb{F}\setminus H} \mathcal{E}^{(\ell-1)}(\langle\!\langle g(\beta)\rangle\!\rangle)$, $\boldsymbol{Value}^{(\ell)}(v) = \mathsf{Spiel}(w)$ and $g|_H = w$. In addition, set $S_v \stackrel{\text{def}}{=} \bigsqcup_{\beta\in\mathbb{F}\setminus H} \boldsymbol{Value}^{(\ell-1)}(v_\beta) = \bigsqcup_{\beta\in\mathbb{F}\setminus H} \mathsf{Spiel}(g(\beta))$. The proof string for $v \in \mathcal{E}^{(\ell)}_{\mathbb{F},k}$ is defined as follows.

$$\boldsymbol{Proof}^{(\ell)}(v) = S_v \sqcup \bigsqcup_{\beta\in\mathbb{F}\setminus H} \boldsymbol{Proof}^{(\ell-1)}(v_\beta) \sqcup \boldsymbol{Proof}_{\mathcal{L}}(S_v)$$

where the code ensemble $\mathcal{L} : \{0,1\}^k \to 2^{\{0,1\}^{O(|\mathbb{F}|\cdot\log|\mathbb{F}|)}}$ is defined as follows. Given $w \in \{0,1\}^k$, $S \in \mathcal{L}(w)$ if and only if there exists a polynomial $g \in \mathcal{C}_{\mathbb{F}}$ such that the following conditions are satisfied.

1. $g|_H = w$.
2. $S = \bigsqcup_{\beta\in\mathbb{F}\setminus H} \mathsf{Spiel}(g(\beta))$.

The following lemma establishes the existence of a Spiel-PCU for $\mathcal{L}$.

▶ **Lemma 35.** *$\mathcal{L}$ has a $q(\varepsilon,\delta)$-query length-t* Spiel-PCU *for $t = O(|\mathbb{F}| \cdot \mathrm{polylog}\,|\mathbb{F}|)$.*

Proof. By Theorem 5, there exists a quasilinear size circuit that decodes $\mathsf{Spiel}(\alpha)$. Using such a circuit, we can decode $g(\beta)$ from S for every $\beta \in \mathbb{F}$. Then, using all the values $g(\beta)$ and w (where the i-th bit of w correspond to the value of the i-th element in H according to the ordering), we use Theorem 7 to interpolate the values and achieve a representation of a polynomial $g : \mathbb{F} \to \mathbb{F}$. If $g \in \mathcal{C}_{\mathbb{F}}$ we accept S and otherwise we reject. Since deciding if $S \in \mathcal{L}(w)$ has a quasilinear size circuit, by Lemma 26, there is a quasilinear length Spiel-PCU for $\mathcal{L}$. ◀

Having defined $\mathbf{Proof}^{(\ell)}$, we first provide an upper bound on the bit length of the prescribed proof string. For $\ell > 0$, let $z^{(\ell)}_{\mathbb{F},k}$ denote the bit length of the proof for membership in $\mathcal{E}^{(\ell)}$ as defined in Definition 34, where for $\ell = 0$ we replace the (nonexistent) field $\mathbb{F}$ with $|w|$.

The following lemma, establishing the proof string's length, relies on our choice of the constant d in Definition 31. In particular, d needs to be large enough to subsume the size of $\mathbf{Proof}_{\mathcal{L}}(\cdot)$

▶ **Lemma 36.** *For any $\ell \geq 0$, we have that $z^{(\ell)}_{\mathbb{F},k} = O\big(m^{(\ell)} \cdot \mathrm{polylog}^{(\ell)} m^{(\ell)}\big)$.*

Proof. The proof follows by induction on ℓ. The base case ($\ell = 0$) follows directly from the definition of $\mathcal{P}^{(0)}$ by our convention that $\log^{(0)}|w| = |w|$.

Consider $\ell > 0$, and note that since the size of S_v is $O(|\mathbb{F}| \log |\mathbb{F}|)$, the size of $\mathbf{Proof}_{\mathcal{L}}(S_v)$ is $O(|\mathbb{F}| \cdot \text{polylog } |\mathbb{F}|)$. By combining the above with the definition of the proof string we have

$$z^{(\ell)}_{\mathbb{F},k} \leq |\mathbb{F}| \cdot \text{polylog } |\mathbb{F}| + |\mathbb{F}| \cdot z^{(\ell-1)}_{\mathbb{F}',k'}.$$

Now, assume that $z^{(\ell-1)}_{\mathbb{F}',k'} = O(m^{(\ell-1)} \cdot \text{polylog}^{(\ell-1)}|\mathbb{F}'|)$. Note that since the global constant d was chosen so that $|\mathbb{F}| \cdot |\mathbb{F}'| \geq |\mathbf{Proof}_{\mathcal{L}}(S_v)|$, we have that $|\mathbb{F}| \cdot z^{(\ell-1)}_{\mathbb{F}',k'} \geq |\mathbb{F}| \cdot |\mathbb{F}'| \geq |\mathbf{Proof}_{\mathcal{L}}(S_v)|$. Therefore,

$$m^{(\ell)} = \Theta(|\mathbb{F}| \cdot m^{(\ell-1)}) = \Omega(|\mathbb{F}| \cdot |\mathbb{F}'|) = \Omega(|\mathbb{F}| \cdot \text{polylog } |\mathbb{F}|),$$

so that $|\mathbb{F}| \cdot \text{polylog } |\mathbb{F}| = O(m^{(\ell)})$, and

$$z^{(\ell)}_{\mathbb{F},k} = O(|\mathbb{F}| \cdot z^{(\ell-1)}_{\mathbb{F}'}).$$

In addition, by the fact that $m_\ell = \Theta(|\mathbb{F}| \cdot m^{(\ell-1)})$ and the induction hypothesis we obtain

$$|\mathbb{F}| \cdot z^{(\ell-1)}_{\mathbb{F}',k'} = O(|\mathbb{F}| \cdot m^{(\ell-1)} \cdot \text{polylog}^{(\ell-1)}|\mathbb{F}'|) = O(m^{(\ell)} \cdot \text{polylog}^{(\ell)}|\mathbb{F}|) = O(m^{(\ell)} \cdot \text{polylog}^{(\ell)} m_\ell).$$

So overall, we get that $z^{(\ell)}_{\mathbb{F},k} = O(m^{(\ell)} \cdot \text{polylog}^{(\ell)} m^{(\ell)})$ as required. ◀

Next, for an alleged proof $\pi = \mathbf{Proof}^{(\ell)}(v)$, we use the notation $\pi|_{\text{Dom}(X)}$ to denote the restriction of π to the bits that correspond to X in π as defined in Definition 34. For example, $\pi|_{\text{Dom}(\mathbf{Value}^{(\ell-1)}(v_\beta))}$ refers to the bits that represent $\mathbf{Value}^{(\ell-1)}(v_\beta)$.

We introduce the verifier procedure for $\mathcal{E}^{(\ell)}_{\mathbb{F},k}$ (see Figure 1), and prove its completeness and soundness. For technical considerations, the verifier procedure is only defined when the soundness parameter δ is small enough (as a function of ℓ); the soundness amplification argument from Subsection 2.2 can easily take care of the situation where δ is larger, by running sufficiently many independent instances of the verification step.

Before proceeding to the completeness and soundness proofs, let us analyze the query complexity. Denote by $Q_\ell(\varepsilon, \delta)$ the query complexity of the verifier in the above procedure for a given $\ell \geq 0$. It follows from the recursive description of `Verifier-Procedure` $_{\mathcal{E}^{(\ell)}}$ and the proof of Lemmas 26 and 35 that the query complexity satisfies the recurrence relation $Q_\ell(\varepsilon, \delta) \leq O(1/\varepsilon) \cdot Q_{\ell-1}(\varepsilon/O(1)), \delta \cdot O(1)) + q^*(\Theta(\varepsilon), \Theta(\delta))$, where $q^*(\varepsilon^*, \delta^*) = O((\varepsilon^*)^{-1})$ is the query complexity of Dinur's PCP [12] with detection radius ε^* and soundness parameter $\delta^* \leq 1/2$; and furthermore, that $Q_0(\varepsilon, \delta) \leq q^*(\Theta(\varepsilon), \Theta(\delta))$. Thus, we conclude by induction that, provided that $\delta \leq 2^{-\ell-1}$,

$$Q_\ell(\varepsilon, \delta) \leq \frac{C}{\varepsilon} \cdot \frac{C^2}{\varepsilon} \cdot \ldots \cdot \frac{C^\ell}{\varepsilon} \cdot q^*(\varepsilon/2^{O(\ell)}, \delta \cdot 2^\ell) = 2^{O(\ell^2)} \varepsilon^{-O(\ell)},$$

where $C > 0$ is a large enough absolute constant. To achieve any given soundness $\delta > 1/2$, we can amplify by repeating the verifier procedure with parameter $\delta' = 2^{-\ell-1}$ a total of $2^{O(\ell)} \cdot \log\left((1-\delta)^{-1}\right)$ times and rejecting if any of these instances rejected. The query complexity is bounded by

$$2^{O(\ell^2)} \varepsilon^{-O(\ell)} \cdot 2^{O(\ell)} \cdot \log\left((1-\delta)^{-1}\right) = (2^\ell/\varepsilon)^{O(\ell)} \cdot \log\left((1-\delta)^{-1}\right),$$

as desired. The next two lemmas establish the completeness and soundeness of the verifier procedure, respectively.

`Verifier-Procedure` $_{\mathcal{E}^{(\ell)}}$

Input: Parameters $\varepsilon, \delta \in (0,1)$ where $\delta \leq 2^{-\ell-1}$, an input $v \in \{0,1\}^{m^{(\ell)}}$, an alleged value $\tau \in \{0,1\}^{100k}$ of v, and an alleged proof $\pi \in \{0,1\}^{z^{(\ell)}_{\mathbb{F},k}}$ for v.

1. If $\ell = 0$, use the PCU for $\mathcal{E}^{(0)}$ with parameters ε and δ.
2. If $\ell > 0$:
   a. Use the PCU verifier for $\mathcal{L}$ with radius $\varepsilon/300$ and soundness δ, to verify the unveiling of $\pi|_{\mathrm{Dom}(S_v)}$, using τ as the value oracle and $\pi|_{\mathrm{Dom}(\mathbf{Proof}_{\mathcal{L}}(S_v))}$ as the proof oracle.
   b. For $6/\varepsilon$ many times:
      i. Pick $\boldsymbol{\beta} \in \mathbb{F} \setminus H$ uniformly at random.
      ii. Use the PCU verifier procedure for $\mathcal{E}^{(\ell-1)}$ with parameters $\varepsilon/3$ and 2δ, to verify the unveiling of $v_{\boldsymbol{\beta}}$, using $\pi|_{\mathrm{Dom}(\mathbf{Value}^{(\ell-1)}(v_\beta))}$ as the value oracle and $\pi|_{\mathrm{Dom}(\mathbf{Proof}^{(\ell-1)}(v_{\boldsymbol{\beta}}))}$ as the proof oracle.

If any of the stages rejected then **Reject**, and otherwise **Accept**.

Figure 1 Description of `Verifier-Procedure` $_{\mathcal{E}^{(\ell)}}$.

▶ **Lemma 37.** *If there exist $w \in \{0,1\}^k$ for which $v \in \mathcal{E}^{(\ell)}_{\mathbb{F},k}(w)$, then* `Verifier-Procedure` $_{\mathcal{E}^{(\ell)}}$ *accepts v with probability 1 when supplied with oracle access to the corresponding $\boldsymbol{Proof}^{(\ell)}(v)$ and $\tau = \boldsymbol{Value}^{(\ell)}(v) = \mathsf{Spiel}(w)$.*

Proof. The proof follows by induction on ℓ. The base case follows directly from Lemma 28. Hence, the verifier for $\mathcal{E}^{(0)}$ supplied with $\mathbf{Proof}^{(0)}(v)$ as the proof oracle and $\mathbf{Value}^{(\ell)}(v)$ as the value oracle, will accept v with probability 1.

Assume that `Verifier-Procedure` $_{\mathcal{E}^{(\ell-1)}}$ accepts with probability 1 any valid encoding v' when supplied with the corresponding oracles for $\mathbf{Value}^{(\ell-1)}(v')$ and $\mathbf{Proof}^{(\ell-1)}(v')$. Let $v \in \mathcal{E}^{(\ell)}$ and write $v = \bigsqcup_{\beta \in \mathbb{F} \setminus H} v_\beta$, where there exist $w \in \{0,1\}^k$ and $g \in \mathcal{C}_{\mathbb{F}}$ such that for all $\beta \in \mathbb{F} \setminus H$, $v_\beta \in \mathcal{E}^{(\ell-1)}(g(\beta))$, where $g|_H = w$ and $\tau = \mathbf{Value}^{(\ell)}(v) = \mathsf{Spiel}(w)$. Then, by the definition of the language $\mathcal{L}$ and the first two components of $\mathbf{Proof}^{(\ell)}(v)$, Step (2a) of `Verifier-Procedure` $_{\mathcal{E}^{(\ell)}}$ will always accept. In addition, for every $\beta \in \mathbb{F} \setminus H$, we have that $v_\beta \in \mathcal{E}^{(\ell-1)}$, and therefore by the induction hypothesis, Step (2b) of `Verifier-Procedure` $_{\mathcal{E}^{(\ell)}}$ will accept the corresponding unveiling for any picked $\beta \in \mathbb{F} \setminus H$. ◀

▶ **Lemma 38.** *If for every $w \in \{0,1\}^k$ either $\mathrm{dist}(\tau, \mathsf{Spiel}(w)) > \varepsilon$ or $\mathrm{dist}(v, \mathcal{E}^{(\ell)}(w)) > \varepsilon$ (or both), then with probability greater than δ,* `Verifier-Procedure` $_{\mathcal{E}^{(\ell)}}$ *will reject v regardless of the contents of the supplied proof string.*

Proof. Let $\tau \in \{0,1\}^{100k}$ be an alleged value for v, and $\pi \in \{0,1\}^{z^{(\ell)}_{\mathbb{F},k}}$ be an alleged proof string for v. We proceed by induction on ℓ. For $\ell = 0$ we use the PCU verifier for $\mathcal{E}^{(0)}$ with error ε and soundness δ to check that v is a member of the code ensemble $\mathcal{E}^{(0)}$ and τ is its value. If the PCU verifier for $\mathcal{E}^{(0)}$ rejects with probability at most δ, then there exist $w \in \{0,1\}^k$ such that $\mathrm{dist}(v, \mathcal{E}^{(0)}(w)) \leq \varepsilon$ and $\mathrm{dist}(\tau, \mathsf{Spiel}(w)) \leq \varepsilon$, and the base case is complete.

Next assume that the lemma holds for $\ell - 1$. If the PCU verifier for $\mathcal{L}$ in Step (2a) rejects with probability at most δ, then there exist a function $g \in \mathcal{C}_{\mathbb{F}}$ and $w \in \{0,1\}^k$ for which $g|_H = w$ so that

$$\mathrm{dist}(\pi|_{\mathrm{Dom}(S_v)}, \mathsf{Spiel}(g|_{\mathbb{F}\setminus H})) \leq \varepsilon/300 \qquad \text{and} \qquad \mathrm{dist}(\tau, \mathsf{Spiel}(w)) \leq \varepsilon/300.$$

In particular, the leftmost inequality means that for at most $\frac{\varepsilon}{3}|\mathbb{F}\setminus H|$ of the elements $\beta \in \mathbb{F}\setminus H$, it holds that

$$\mathrm{dist}(\pi|_{\mathrm{Dom}(\mathbf{Value}^{(\ell-1)}(v_\beta))}, \mathsf{Spiel}(g(\beta)) > 1/100.$$

We refer to elements $\beta \in \mathbb{F} \setminus H$ satisfying the above inequality as *bad* elements, and to the rest as *good* elements. Let G denote the set of good elements.

Next, we show that if the loop that uses the PCU verifier for $\mathcal{E}^{(\ell-1)}$ in Step (2b) rejects with probability at most δ, then for at most an $\varepsilon/3$ fraction of the good $\beta \in \mathbb{F} \setminus H$, it holds that

$$\mathrm{dist}\left(v_\beta, \mathcal{E}^{(\ell-1)}(\langle\!\langle g(\beta)\rangle\!\rangle)\right) > \varepsilon/3.$$

Assume that there are more than $\frac{\varepsilon}{3} \cdot |G|$ good elements such that $\mathrm{dist}\left(v_\beta, \mathcal{E}^{(\ell-1)}(\langle\!\langle g(\beta)\rangle\!\rangle)\right) > \varepsilon/3$. Then, by our induction hypothesis, each of them will be rejected by the PCU verifier for $\mathcal{E}^{(\ell-1)}$ with probability more than 2δ. In addition, with probability at least $1/2$ we sample at least one such good β, and then during this iteration the verifier in Step (2b(ii)) rejects with conditional probability more than 2δ, and hence the verifier will reject with overall probability more than δ. Summing everything up, when the input is rejected with probability at most δ,

$$\mathrm{dist}\left(v, \bigsqcup_{\beta\in\mathbb{F}\setminus H} \mathcal{E}^{(\ell-1)}(\langle\!\langle g(\beta)\rangle\!\rangle)\right) \leq \varepsilon/3 + (1-\varepsilon/3)\cdot\varepsilon/3 + (1-\varepsilon/3)^2\cdot\varepsilon/3 \leq \varepsilon,$$

where the three summands are respectively the contribution to the distance of the bad elements, the good elements with v_β being far from any level $\ell - 1$ encoding of $\langle\!\langle g(\beta)\rangle\!\rangle$, and all the other elements. ◀

The proof of Lemma 33 follows directly by combining Lemma 36, Lemma 37 and Lemma 38.

The following corollary follows directly from Lemma 33 and the definition of Spiel-PCU (Definition 25), and implies Lemma 29.

▶ **Corollary 39.** *Let $\mathbb{F}$ be a finite field and $k \in \mathbb{N}$ which satisfy the requirements in Definition 31. Then, for every $\ell \geq 0$ the coding ensemble $\mathcal{E}^{(\ell)}_{\mathbb{F},k} : \{0,1\}^k \to 2^{\left(\{0,1\}^{m^{(\ell)}}\right)}$ has a $q(\varepsilon,\delta)$-query, length-t Spiel-PCU for $t = O(m^{(\ell)}\mathrm{polylog}^{(\ell)}m^{(\ell)})$.*

5.3 The Lower Bound

We turn to prove the linear query lower bound for the testability of our property. We start by defining distributions over strings of length $m^{(\ell)}$.

Distribution $\mathcal{D}^{(\ell)}_{\mathsf{yes}}(w)$: Given $w \in \{0,1\}^k$, we define the distribution $\mathcal{D}^{(\ell)}_{\mathsf{yes}}(w)$ to be the uniform distribution over elements in $\mathcal{E}^{(\ell)}(w)$.

Distribution $\mathcal{D}^{(\ell)}_{\mathsf{no}}$: An element v from $\mathcal{D}^{(\ell)}_{\mathrm{no}}$ is drawn by the following process. For $\ell = 0$, $\boldsymbol{v}$ is a uniformly random string in $\{0,1\}^{4k}$. For $\ell > 0$, we pick a uniformly random function $\boldsymbol{\lambda} : \mathbb{F} \setminus H \to \mathbb{F}$, and let v be a uniformly random element of $\bigsqcup_{\beta \in \mathbb{F} \setminus H} \mathcal{E}^{(\ell-1)}(\langle\!\langle \boldsymbol{\lambda}(\beta) \rangle\!\rangle)$

▶ **Lemma 40.** *For any $\ell \geq 0$, every $w \in \{0,1\}^k$ and $q = o(m^{(\ell)}/10^{\ell})$, any algorithm making at most q queries cannot distinguish (with constant probability) between $\boldsymbol{v} \sim \mathcal{D}^{(\ell)}_{yes}(w)$ and $\boldsymbol{u}$ which is drawn according to any of the following distributions:*
1. *$\mathcal{D}^{(\ell)}_{yes}(w')$ for any $w' \neq w$.*
2. *$\mathcal{D}^{(\ell)}_{no}$.*

Note that Item (1) in the above follows immediately from Item (2). Additionally, the first item implies the shared secret property of the code ensemble $\mathcal{E}^{(\ell)}$. Furthermore, we remark that that above lemma implies a more stringent version of PCUSS. In addition to the shared secret property, Item (2) implies that the ensemble $\mathcal{E}^{(\ell)}$ is indistinguishable from strings that are mostly far from any encoding (i.e., drawn from $\mathcal{D}^{(\ell)}_{\mathrm{no}}$).

The proof of Lemma 40 follows by induction over ℓ. Before we continue, we introduce some useful lemmas that will be used in the proof.

▶ **Lemma 41.** *For any $\ell \geq 0$ and $w, w' \in \{0,1\}^k$ for which $w \neq w'$ it holds that*

$$\min_{(v,v') \in \mathcal{E}^{(\ell)}(w) \times \mathcal{E}^{(\ell)}(w')} \mathrm{dist}(v, v') = \Theta\left(1/4^{\ell+1}\right).$$

Proof. The proof follows by induction over ℓ. The base case for $\ell = 0$ follows directly by the fact that the code from Definition 21 has high distance, and in particular $\mathrm{dist}(\mathcal{E}^{(0)}(w), \mathcal{E}^{(0)}(w')) > 1/10$. Assume that the lemma holds for $\ell - 1$. Namely, for $w, w' \in \{0,1\}^{k'}$ for which $w \neq w'$ it holds that

$$\min_{(v,v') \in \mathcal{E}^{(\ell-1)}(w) \times \mathcal{E}^{(\ell-1)}(w')} \mathrm{dist}(v, v') = \Theta\left(1/4^{\ell}\right).$$

Let $\tilde{w}, \tilde{w}' \in \{0,1\}^k$ be such that $\tilde{w}' \neq \tilde{w}$. Then we can write $(\tilde{v}, \tilde{v}') \in \mathcal{E}^{(\ell)}(\tilde{w}) \times \mathcal{E}^{(\ell)}(\tilde{w}')$ as

$$\tilde{v} = \bigsqcup_{\beta \in \mathbb{F} \setminus H} \mathcal{E}^{(\ell-1)}(\langle\!\langle g(\beta) \rangle\!\rangle) \qquad \text{and} \qquad \tilde{v}' = \bigsqcup_{\beta \in \mathbb{F} \setminus H} \mathcal{E}^{(\ell-1)}(\langle\!\langle g'(\beta) \rangle\!\rangle),$$

for some $g, g' \in \mathcal{C}_{\mathbb{F}}$ such that $g|_H = \tilde{w}$ and $g'|_H = \tilde{w}'$. By the fact that g and g' are degree $|\mathbb{F}|/2$ polynomials (which are not identical), we have that g and g' disagree on at least $|\mathbb{F} \setminus H|/4$ of the elements $\beta \in \mathbb{F} \setminus H$. By applying the induction hypothesis on the minimum distance between $\mathcal{E}^{(\ell)}(\langle\!\langle g(\beta) \rangle\!\rangle)$ and $\mathcal{E}^{(\ell)}(\langle\!\langle g'(\beta) \rangle\!\rangle)$, for all β such that $g(\beta) \neq g'(\beta)$, we have that

$$\min_{(\tilde{v},\tilde{v}') \in \mathcal{E}^{(\ell)}(\tilde{w}) \times \mathcal{E}^{(\ell)}(\tilde{w}')} \mathrm{dist}(\tilde{v}, \tilde{v}') > \frac{1}{4} \cdot \Theta\left(\frac{1}{4^{\ell}}\right) = \Theta\left(1/4^{\ell+1}\right). \qquad \blacktriangleleft$$

▶ **Lemma 42.** *For any $\ell \geq 0$, with probability at least $1 - o(1)$, a string v drawn from $\mathcal{D}^{(\ell)}_{no}$ satisfies $\mathrm{dist}(v, \mathcal{E}^{(\ell)}(w)) = \Theta\left(1/4^{\ell+1}\right)$ for all $w \in \{0,1\}^k$.*

Proof. The proof follows by induction over ℓ. For $\ell = 0$, fix some $w \in \{0,1\}^k$. Consider the size of a ball of relative radius $1/40$ around some $v \in \mathcal{E}^{(0)}(w)$ in the space of all strings $\{0,1\}^{4k}$. The number of strings contained in this ball is at most

$$\binom{4k}{k/10} \leq (40e)^{k/10} = 2^{k/10 \cdot \log(40e)}.$$

Thus, the size of the set of strings which are at relative distance $1/40$ from any legal encoding of some word $w \in \{0,1\}^k$ is at most

$$2^{3k} \cdot 2^{k/10 \cdot \log(40e)} = o(2^{4k}).$$

This implies that with probability at least $1 - o(1)$, a random string from $\{0,1\}^{4k}$ is $1/40$-far from $\mathcal{E}^{(0)}(w)$ for any $w \in \{0,1\}^k$.

For any $\ell > 0$, consider v' sampled according to $\mathcal{D}^{(\ell)}_{\text{no}}$. Then, v' can be written as

$$v' = \bigsqcup_{\beta \in \mathbb{F} \setminus H} \mathcal{E}^{(\ell-1)}(\langle\langle \boldsymbol{\lambda}(\beta) \rangle\rangle),$$

where $\boldsymbol{\lambda} : \mathbb{F} \setminus H \to \mathbb{F}$ is a uniformly random function. On the other hand, each member $\tilde{v}$ of $\mathcal{P}^{(\ell)}$ can be written as

$$\tilde{v} = \bigsqcup_{\beta \in \mathbb{F} \setminus H} \mathcal{E}^{(\ell-1)}(\langle\langle g(\beta) \rangle\rangle),$$

for some $g \in \mathcal{C}_{\mathbb{F}}$ such that $g|_H = w$ for some $w \in \{0,1\}^k$. Note that by Lemma 41, whenever $\boldsymbol{\lambda}(\beta) \neq g(\beta)$, we have that the minimum distance between any $\tilde{v} \in \mathcal{E}^{(\ell-1)}(\langle\langle g(\beta) \rangle\rangle)$ and $v' \in \mathcal{E}^{(\ell-1)}(\langle\langle \boldsymbol{\lambda}(\beta) \rangle\rangle)$ is at least $\Theta(1/4^\ell)$. In addition, by Lemma 8, we have that that with probability at least $1 - o(1)$, a uniformly random function $\boldsymbol{\lambda} : \mathbb{F} \to \mathbb{F}$ is $1/3$-far from any $g \in \mathcal{C}_{\mathbb{F}}$. By the restrictions on k in Definition 31, which implies that $|H| \leq |F|/c$, we can ensure (by the choice of c) that with probability at least $1 - o(1)$, that a uniformly random $\boldsymbol{\lambda} : \mathbb{F} \setminus H \to \mathbb{F}$ is at least $1/4$-far from the restriction $g|_{\mathbb{F} \setminus H}$. This implies that for at least $|\mathbb{F} \setminus H|/4$ of the elements $\beta \in \mathbb{F} \setminus H$, we have that $\boldsymbol{\lambda}(\beta) \neq g(\beta)$. Therefore, we have that $\text{dist}(v', \mathcal{E}^{(\ell)}(w)) = \frac{1}{4} \cdot \Theta\left(\frac{1}{4^\ell}\right) = \Theta\left(1/4^{\ell+1}\right)$ for all $w \in \{0,1\}^k$, and the proof is complete. ◀

▶ **Lemma 43.** *Fix any $\ell > 0$, and suppose that for any $w' \in \{0,1\}^{k'}$, and for any set Q' of at most $q^{(\ell-1)}_{\mathbb{F}',k'}$ queries (where $\mathbb{F}'$ and k' are picked according to the recursive definition of the level ℓ-encoding, and for $q^{(0)}$ we substitute k' for the nonexistent $\mathbb{F}'$) the restricted distributions $\mathcal{D}^{(\ell-1)}_{yes}(w')|_{Q'}$ and $\mathcal{D}^{(\ell-1)}_{no}|_{Q'}$ are identical. Then, for any $w \in \{0,1\}^k$, and any set Q of at most $\frac{|\mathbb{F} \setminus H|}{10} \cdot q^{(\ell-1)}_{\mathbb{F}',k'}$ queries, the restricted distributions $\mathcal{D}^{(\ell)}_{yes}(w)|_Q$ and $\mathcal{D}^{(\ell)}_{no}|_Q$ are identical.*

Proof. Let $Q \subset [m^{(\ell)}]$ be the set of queries, and fix a canonical ordering over the elements in $\mathbb{F} \setminus H$. Let $\boldsymbol{v}$ be an element drawn according to distribution $\mathcal{D}^{(\ell)}_{\text{yes}}(w)$, and let $\boldsymbol{v}'$ be an element drawn according to distribution $\mathcal{D}^{(\ell)}_{\text{no}}$. The sampling process from $\mathcal{D}^{(\ell)}_{\text{yes}}(w)$ can be thought of as first drawing a uniformly random function $\boldsymbol{g} \in \mathcal{C}_{\mathbb{F}}$ such that $\boldsymbol{g}|_H = w$, and for every $\beta \in \mathbb{F} \setminus H$, letting $\boldsymbol{v}_\beta$ be a uniformly random element in $\mathcal{E}^{(\ell-1)}(\langle\langle \boldsymbol{g}(\beta) \rangle\rangle)$.

For each $\beta \in \mathbb{F} \setminus H$ we set $Q_\beta = Q \cap \text{Dom}(v_\beta)$, and define the set of *big clusters*

$$I = \left\{ \beta \in \mathbb{F} \setminus H \ : \ |Q_\beta| \geq q^{(\ell-1)}_{\mathbb{F}',k'} \right\}.$$

Note that since $|Q| \leq |\mathbb{F} \setminus H| \cdot q^{(\ell-1)}_{\mathbb{F}',k'}/10$, we have that $|I| \leq |\mathbb{F} \setminus H|/10$.

By the fact that $\boldsymbol{g}$ is a uniformly random polynomial of degree $|\mathbb{F}|/2 > |I|$, we have that $\boldsymbol{g}|_I$ is distributed exactly as $\boldsymbol{\lambda}|_I$ (both are a sequence of $|I|$ independent uniformly random values), which implies that $\boldsymbol{v}|_{\bigcup_{j \in I} Q_j}$ is distributed exactly as $\boldsymbol{v}'|_{\bigcup_{j \in I} Q_j}$.

Next, let $\mathbb{F} \setminus (I \cup H) = \{i_1, \ldots, i_{|\mathbb{F} \setminus (I \cup H)|}\}$ be a subset ordered according to the canonical ordering over $\mathbb{F}$. We proceed by showing that $\boldsymbol{v}|_{\bigcup_{j \in I \cup \{i_1, \ldots, i_t\}} Q_j}$ is distributed identically to $\boldsymbol{v}'|_{\bigcup_{j \in I \cup \{i_1, \ldots, i_t\}} Q_j}$ by induction over t.

The base case ($t = 0$) corresponds to the restriction over $\bigcup_{j \in I} Q_j$, which was already proven above. For the induction step, let $T = \{i_1, \ldots, i_{t-1}\} \subseteq \mathbb{F} \setminus (I \cup H)$ be an ordered subset that agrees with the canonical ordering on $\mathbb{F}$, and let $i_t \in \mathbb{F} \setminus (H \cup T \cup I)$ be the successor of i_{t-1} according to the ordering. We now prove that for each $x \in \{0,1\}^{m^{(\ell)}}$ for which $\boldsymbol{v}|_{\bigcup_{j \in I \cup T} Q_j}$ has a positive probability of being equal to $x|_{\bigcup_{j \in I \cup T} Q_j}$, conditioned on the above event taking place (and its respective event for v'), $\boldsymbol{v}|_{Q_{i_t}}$ is distributed exactly as $\boldsymbol{v'}|_{Q_{i_t}}$.

Observe that conditioned on the above event, $\boldsymbol{v}|_{Q_{i_t}}$ is distributed exactly as a uniformly random element in $\mathcal{E}^{(\ell-1)}(\boldsymbol{\rho})$ for some $\boldsymbol{\rho} \in \{0,1\}^{k'}$ (which follows some arbitrary distribution, possibly depending on $x|_{\bigcup_{j \in I \cup T} Q_j}$), while $\boldsymbol{v'}|_{Q_{i_t}}$ is distributed exactly as a uniformly random element in $\mathcal{E}^{(\ell-1)}(\boldsymbol{y})$ for a uniformly random $\boldsymbol{y} \in \{0,1\}^{k'}$. By the fact that $|Q_{i_t}| \leq q^{(\ell-1)}_{\mathbb{F}',k'}/10$, we can apply the induction hypothesis and conclude that $\boldsymbol{v}|_{Q_{i_t}}$ is distributed exactly as $\boldsymbol{v'}|_{Q_{i_t}}$, because by our hypothesis both are distributed identically to the corresponding restriction of $\mathcal{D}^{(\ell-1)}_{\text{no}}$, regardless of the values picked for $\boldsymbol{\rho}$ and $\boldsymbol{y}$. This completes the induction step for t. The lemma follows by setting $t = |\mathbb{F} \setminus H \cup I|$. ◀

▶ **Lemma 44.** *For any $\ell \geq 0$, $w \in \{0,1\}^k$ and any set of queries $Q \subset [m^{(\ell)}]$ such that $|Q| = O\left(\frac{m^{(\ell)}}{10^\ell}\right)$, the restricted distributions $\mathcal{D}^{(\ell)}_{yes}(w)|_Q$ and $\mathcal{D}^{(\ell)}_{no}|_Q$ are identically distributed.*

Proof. By induction on ℓ. For $\ell = 0$ and any $w \in \{0,1\}^k$, by the fact that our base encoding $\mathcal{E}^{(0)}(w)$ is a high dual distance code, we can select (say) $q^{(0)} = k/c$ (for some constant $c > 0$), making the assertion of the lemma trivial.

Assume that for any $w' \in \{0,1\}^{k'}$, and any set of queries Q' of size up to $O(m^{(\ell-1)}/10^{\ell-1})$ the conditional distributions $\mathcal{D}^{(\ell-1)}_{\text{yes}}(w')|_{Q'}$ and $\mathcal{D}^{(\ell-1)}_{\text{no}}|_{Q'}$ are identically distributed. Then, by Lemma 43, we have that for any $w \in \{0,1\}^k$ and any set of queries Q of size at most

$$O\left(\frac{|\mathbb{F} \setminus H|}{10^\ell} \cdot m^{(\ell-1)}\right),$$

the restricted distributions $\mathcal{D}^{(\ell)}_{\text{yes}}(w)|_Q$ and $\mathcal{D}^{(\ell)}_{\text{no}}|_Q$ are identically distributed. Note that by definition of the level ℓ-encoding, $m^{(\ell)} = |\mathbb{F} \setminus H| \cdot m^{(\ell-1)}$, which implies the conclusion of the lemma. ◀

Proof of Lemma 40. Lemma 40 follows directly by combining Lemma 17, and Lemma 44. ◀

Combining Lemma 40 with the definition of Spiel-PCU (Definition 25) establishes that we have constructed a Spiel-PCUSS, which implies Theorem 30.

▶ **Corollary 45.** *Let $\mathbb{F}$ be a finite field and $k \in \mathbb{N}$ which satisfy the requirements in Definition 31. Then, for every $\ell \geq 0$, the coding ensemble $\mathcal{E}^{(\ell)}_{\mathbb{F},k} : \{0,1\}^k \to 2^{\left(\{0,1\}^{m^{(\ell)}}\right)}$ has $q(\varepsilon,\delta)$-query length-t Spiel-PCUSS for $t = O(m^{(\ell)}\text{polylog}^{(\ell)}m^{(\ell)})$.*

5.4 Handling arbitrary input lengths

As mentioned in the beginning of this section, our construction of code ensembles relies on the fact that operations over a finite field $\text{GF}(2^t)$ can be computed efficiently. In order to do so we need to have an irreducible polynomial of degree t over $\text{GF}(2)$, so that we have a representation $\text{GF}(2^t)$. Given such a polynomial, operations over the field can be

implemented in polylogarithmic time in the size of the field. By [18] (Appendix G), we know that for $t = 2 \cdot 3^r$ where $r \in \mathbb{N}$, we do have such a representation. However, the setting of t restricts the sizes of the fields that we can work with, which will limit our input size length.

We show here how to extend our construction to a set of sizes that is "log-dense". For a global constant c', our set of possible input sizes includes a member of $[m', c'm']$ for every m'. Moving from this set to the set of all possible input sizes now becomes a matter of straightforward padding.

For any $n \in \mathbb{N}$, let r be the smallest integer such that $n < 2^{2 \cdot 3^r}$ and let $\mathbb{F} = \mathrm{GF}(2^{2 \cdot 3^r})$. We make our change only at the level-ℓ construction. First, we use $4d$ instead of d in the calculation of the size of $\mathbb{F}'$. Then, instead of using $\mathbb{F} \setminus H$ as the domain for our input, we use $E \setminus H$, for any arbitrary set $E \subseteq \mathbb{F}$ of size $n \geq \max\{4k, |\mathbb{F}|^{1/4}, c_\ell\}$ that contains H. Then, for the level-ℓ, instead of considering polynomials of degree $|\mathbb{F}|/2$, we consider polynomials of degree $|E|/2$. The rest of the construction follows the same lines as the one defined above. This way, all of our operations can be implemented in polylogarithmic time in $|E|$.

6 Separations of testing models

In this section we use Theorem 30 to prove a separation between the standard testing model, and both the tolerant and the erasure resilient testing models. Specifically, we prove the following.

▶ **Theorem 46** (Restatement of Theorem 2). *For every constant $\ell \in \mathbb{N}$, there exist a property $\mathcal{Q}^{(\ell)}$ and $\varepsilon_1 = \varepsilon_1(\ell) \in (0,1)$ such that the following hold.*

1. *For every $\varepsilon \in (0,1)$, the property $\mathcal{Q}^{(\ell)}$ can be ε-tested using a number of queries depending only on ε (and ℓ).*
2. *For every $\varepsilon_0 \in (0, \varepsilon_1)$, any $(\varepsilon_0, \varepsilon_1)$-tolerant tester for $\mathcal{Q}^{(\ell)}$ needs to make $\Omega(N/10^\ell \cdot \mathrm{polylog}^{(\ell)} N)$ many queries on inputs of length N.*

▶ **Theorem 47** (Restatement of Theorem 3). *For every constant $\ell \in \mathbb{N}$, there exist a property $\mathcal{Q}^{(\ell)}$ and $\varepsilon_1 = \varepsilon_1(\ell) \in (0,1)$ such that the following hold.*

1. *For every $\varepsilon \in (0,1)$, the property $\mathcal{Q}^{(\ell)}$ can be ε-tested using a number of queries depending only on ε (and ℓ).*
2. *For every $\varepsilon \in (0, \varepsilon_1)$ and any $\alpha = \Omega(1/\log^{(\ell)} N)$ satisfying $\varepsilon + \alpha < 1$, any α-erasure resilient ε-tester for $\mathcal{Q}^{(\ell)}$ needs to make $\Omega(N/10^\ell \cdot \mathrm{polylog}^{(\ell)} N)$ many queries on inputs of length N.*

In order to prove the separation we use the code ensemble $\mathcal{E}^{(\ell)}_{\mathbb{F},k}$ where k is set to 0. Namely, we consider $\mathcal{E}_{\mathbb{F},0}(\emptyset)$. Note that in this case, the code ensemble becomes a property (i.e. a subset of the set of all possible strings).

Next, we define the property that exhibits the separation between the standard testing model and both the tolerant testing model and the erasure resilient model. We prove Theorem 46 and mention the small difference between the proof of Theorem 46 and the proof of Theorem 47.

▶ **Definition 48.** *Fix a finite field $\mathbb{F}$ and a constant integer $\ell \in \mathbb{N}$ and let $\varepsilon(\ell) = \Theta(1/4^\ell)$. Let $n \stackrel{\text{def}}{=} m^{(\ell)}_{\mathbb{F}}$, $z^{(\ell)}_{\mathbb{F},0} \leq n \cdot \mathrm{polylog}^{(\ell)} n$ denote the length of the proof for the* PCUSS *from Theorem 30, and let $N = (\log^{(\ell)} n + 1) \cdot z^{(\ell)}_{\mathbb{F},0}$. Let $\mathcal{Q}^{(\ell)} \subseteq \{0,1\}^N$ be defined as follows. A string $x \in \{0,1\}^N$ satisfies $\mathcal{Q}^{(\ell)}$ if the following hold.*

1. *The first $z^{(\ell)}_{\mathbb{F},0} \cdot \log^{(\ell)} n$ bits of x consist of $s = \frac{z^{(\ell)}_{\mathbb{F},0} \cdot \log^{(\ell)} n}{n}$ copies of $y \in \mathcal{E}^{(\ell)}_{\mathbb{F},0}$.*
2. *The remaining $z^{(\ell)}_{\mathbb{F},0}$ bits of x consist of a proof string $\pi \in \{0,1\}^{z^{(\ell)}_{\mathbb{F},0}}$, for which the* `Verifier-Procedure` $_{\mathcal{E}^{(\ell)}_{\mathbb{F},0}}$ *in Figure 1 accepts y given oracle access to y and π.*

We first show that $\mathcal{Q}^{(\ell)}$ can be tested using a constant number of queries in the standard testing model.

`Testing Algorithm for` $\mathcal{Q}^{(\ell)}$

Input: Parameter $\varepsilon \in (0,1)$, an oracle access to $x \in \{0,1\}^N$.

1. Set $s \stackrel{\text{def}}{=} \frac{z^{(\ell)}_{\mathbb{F},0} \cdot \log^{(\ell)} n}{n}$.
2. Repeat $4/\varepsilon$ times:
   a. Sample $\boldsymbol{j} \in [n]$ and $\boldsymbol{i} \in [s] \setminus \{1\}$ uniformly at random.
   b. If $x_{\boldsymbol{j}} \neq x_{(\boldsymbol{i}-1)\cdot n+\boldsymbol{j}}$, then **Reject**.
3. Let $v = (x_1, \ldots, x_n)$, $\pi = (x_{z^{(\ell)}_{\mathbb{F},0} \cdot \log^{(\ell)} n+1}, \ldots, x_{(\log^{(\ell)} n+1) z^{(\ell)}_{\mathbb{F},0}})$ and τ be the empty string.
4. Run the PCU verifier for $\mathcal{E}^{(\ell)}_{\mathbb{F},0}$ with parameters $\varepsilon/3$ and $\delta = 2/3$ on v, using π as the alleged proof for v, and τ as the alleged value for v.
5. If the PCU verifier rejects, then **Reject**; otherwise **Accept**.

Figure 2 Description of `Testing Algorithm for` $\mathcal{Q}^{(\ell)}$.

For Item 4 in Figure 2, recall that running the PCU verifier with parameter $\delta = 2/3$ actually involves running multiple instances of the verifier with smaller δ, as discussed in Subsection 5.2.

▶ **Lemma 49.** *The property $\mathcal{Q}^{(\ell)}$ has a tester with query complexity depending only on ε.*

Proof. We show that the algorithm described in Figure 2 is a testing algorithm for $\mathcal{Q}^{(\ell)}$. We assume that n is large enough so that $\log^{(\ell)} n > 6/\varepsilon$.

Assume that $x \in \mathcal{Q}^{(\ell)}$. Then, there exists a string $y \in \mathcal{E}^{(\ell)}_{\mathbb{F},0}$ with $x_1, \ldots, x_{z^{(\ell)}_{\mathbb{F},0} \log^{(\ell)} n} = (y)^s$ (where $(y)^s$ is the concatenation of s copies of y), and $x_{z^{(\ell)}_{\mathbb{F},0} \cdot \log^{(\ell)} n+1}, \ldots, x_{(\log^{(\ell)}+1) z^{(\ell)}_{\mathbb{F},0}} = \pi \in \{0,1\}^{z^{(\ell)}_{\mathbb{F},0}}$, where π is a proof that makes the PCU verifier for $\mathcal{E}^{(\ell)}_{\mathbb{F},0}$ accept when given oracle access to y and π. Therefore, the algorithm in Figure 2 accepts x.

Next, assume that x is ε-far from $\mathcal{Q}^{(\ell)}$, and let $y' = x_1, \ldots, x_n$. Note that if the string $x_1, \ldots, x_{z^{(\ell)}_{\mathbb{F},0} \cdot \log^{(\ell)} n}$ is $\varepsilon/2$-far from being $(z')^s$, then the loop in Step 2 rejects x with probability at least $2/3$, and we are done. If $x_1, \ldots, x_{z^{(\ell)}_{\mathbb{F},0} \cdot \log^{(\ell)} n}$ is $\varepsilon/2$-close to $(y')^s$, then y' must be $\varepsilon/3$-far from $\mathcal{E}^{(\ell)}_{\mathbb{F},0}$. To see this, assume toward a contradiction that y' is $\varepsilon/3$-close to $\mathcal{E}^{(\ell)}_{\mathbb{F},0}$. Then, by modifying at most $\frac{\varepsilon \cdot z^{(\ell)}_{\mathbb{F},0} \cdot \log^{(\ell)} n}{2}$ bits, we can make $x_1, \ldots, x_{z^{(\ell)}_{\mathbb{F},0} \cdot \log^{(\ell)} n}$ equal to $(y')^s$. Since, by our assumption y' is $\varepsilon/3$-close to $\mathcal{E}^{(\ell)}_{\mathbb{F},0}$, we can further modify the string $(y')^s$ to $(\tilde{y})^s$, where $\tilde{y} \in \mathcal{E}^{(\ell)}_{\mathbb{F},0}$, by changing at most $\frac{\varepsilon \cdot z^{(\ell)}_{\mathbb{F},0} \cdot \log^{(\ell)} n}{3}$ bits. Finally, by changing at most $z^{(\ell)}_{\mathbb{F},0}$ bits from π, we can get a proof string $\tilde{\pi}$ which will make the PCPP verifier accept $\tilde{y}$. By our assumption that $6/\varepsilon < \log^{(\ell)} n$, the total number of changes to the input string x is at most

$$\frac{\varepsilon \cdot z^{(\ell)}_{\mathbb{F},0} \cdot \log^{(\ell)} n}{2} + \frac{\varepsilon \cdot z^{(\ell)}_{\mathbb{F},0} \cdot \log^{(\ell)} n}{3} + z^{(\ell)}_{\mathbb{F},0} \leq \varepsilon \cdot (\log^{(\ell)} n + 1) \cdot z^{(\ell)}_{\mathbb{F},0} = \varepsilon N,$$

which is a contradiction to the fact that x is ε-far from $\mathcal{E}^{(\ell)}_{\mathbb{F},0}$.

Finally, having proved that y' is $\varepsilon/3$-far from $\mathcal{E}^{(\ell)}_{\mathbb{F},0}$, the PCU verifier for $\mathcal{E}^{(\ell)}_{\mathbb{F},0}$ (when called with parameters $\varepsilon/3$ and $\delta = 2/3$) rejects with probability at least $2/3$. ◀

▶ **Lemma 50.** *For every constant $\ell \in \mathbb{N}$, there exists $\varepsilon_1 \stackrel{\text{def}}{=} \Theta(1/4^\ell)$ such that for every $\varepsilon_0 < \varepsilon_1$, any $(\varepsilon_0, \varepsilon_1)$-tolerant tester for $\mathcal{Q}^{(\ell)}$ needs to make at least $\Omega\left(\frac{N}{10^\ell \cdot \text{polylog}^{(\ell)} N}\right)$ many queries.*

Proof. Fix some constant $\ell \in \mathbb{N}$. The proof follows by a reduction from $2\varepsilon_1$-testing of $\mathcal{E}^{(\ell)}_{\mathbb{F},0}$. Given oracle access to a string $y \in \{0,1\}^n$ which we would like to $2\varepsilon_1$-test for $\mathcal{E}^{(\ell)}_{\mathbb{F},0}$, we construct an input string $x \in \{0,1\}^N$ where $N = (\log^{(\ell)} n + 1) \cdot z^{(\ell)}_{\mathbb{F},0}$ as follows.

$$x \stackrel{\text{def}}{=} (y)^{\frac{z^{(\ell)}_{\mathbb{F},0} \cdot \log^{(\ell)} n}{n}} \sqcup (0)^{z^{(\ell)}_{\mathbb{F},0}}.$$

That is, we concatenate $z^{(\ell)}_{\mathbb{F},0} \cdot \log^{(\ell)} n / n$ copies of y, and set the last $z^{(\ell)}_{\mathbb{F},0}$ bits to 0. Note that a single query to the new input string x can be simulated using at most one query to the string y.

If $y \in \mathcal{E}^{(\ell)}_{\mathbb{F},0}$, then for large enough n we have that x is ε_0-close to $\mathcal{Q}^{(\ell)}$, since the last $z^{(\ell)}_{\mathbb{F},0}$ bits that are set to 0 are less than an ε_0-fraction of the input length.

On the other hand, if $\text{dist}(x, \mathcal{E}^{(\ell)}_{\mathbb{F},0}) > 2\varepsilon_1$, since each copy of y in x is $2\varepsilon_1$-far from $\mathcal{E}^{(\ell)}_{\mathbb{F},0}$, then x is $\frac{2\varepsilon_1 \cdot \log^{(\ell)} n}{\log^{(\ell)} n + 1}$-far from $\mathcal{Q}^{(\ell)}$ (note that $\frac{\log^{(\ell)} n}{\log^{(\ell)} n + 1} > 1/2$). Therefore, an $(\varepsilon_0, \varepsilon_1)$-tolerant tester for $\mathcal{Q}^{(\ell)}$ would imply an $2\varepsilon_1$-tester for $\mathcal{E}^{(\ell)}_{\mathbb{F},0}$ with the same query complexity. By Lemma 40, since for some $\varepsilon_1 = \Theta(1/4^\ell)$, every $2\varepsilon_1$-tester for $\mathcal{E}^{(\ell)}_{\mathbb{F},0}$ requires $\Omega(n/10^\ell)$ queries on inputs of length n, any $(\varepsilon_0, \varepsilon_1)$-tolerant tester for $\mathcal{Q}^{(\ell)}$ requires to make $\Omega\left(\frac{N}{10^\ell \cdot \text{polylog}^{(\ell)} N}\right)$ many queries. ◀

Proof of Theorem 46. The proof follows by combining Lemma 49 and Lemma 50. ◀

Proof of Theorem 47. The proof of Theorem 47 is almost identical to the proof of Theorem 46. The only difference is that we replace Lemma 50 with a counterpart for erasure resilient testing, where instead of setting the last $z^{(\ell)}_{\mathbb{F},0}$ bits of x to $(0)^{z^{(\ell)}_{\mathbb{F},0}}$, we use $(\perp)^{z^{(\ell)}_{\mathbb{F},0}}$, noting that the relative size of this part of the input is $1/(s+1) = \Theta(1/\log^{(\ell)}(N))$. ◀

References

1. Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof Verification and the Hardness of Approximation Problems. *J. ACM*, 45(3):501–555, 1998.
2. Omri Ben-Eliezer and Eldar Fischer. Earthmover Resilience and Testing in Ordered Structures. In *Proceedings of the 33rd Conference on Computational Complexity (CCC)*, pages 18:1–18:35, 2018.
3. Eli Ben-Sasson, Alessandro Chiesa, Ariel Gabizon, Michael Riabzev, and Nicholas Spooner. Interactive Oracle Proofs with Constant Rate and Query Complexity. In *Proceedings of the 44th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 40:1–40:15, 2017.
4. Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive Oracle Proofs. In *Theory of Cryptography - 14th International Conference TCC Proceedings, Part II*, pages 31–60, 2016.
5. Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil P. Vadhan. Robust PCPs of Proximity, Shorter PCPs, and Applications to Coding. *SIAM Journal on Computing*, 36(4):889–974, 2006.
6. Eli Ben-Sasson, Yohay Kaplan, Swastik Kopparty, Or Meir, and Henning Stichtenoth. Constant Rate PCPs for Circuit-SAT with Sublinear Query Complexity. *Journal of the ACM*, 63(4):32:1–32:57, 2016.

7 Eli Ben-Sasson and Madhu Sudan. Short PCPs with polylog query complexity. *SIAM Journal on Computing*, 38(2):551–607, 2008.

8 Piotr Berman, Meiram Murzabulatov, and Sofya Raskhodnikova. Tolerant Testers of Image Properties. In *Proceedings of the 43rd International Colloquium on Automata, Languages and Programming (ICALP)*, pages 90:1–90:14, 2016.

9 Eric Blais, Clément L. Canonne, Talya Eden, Amit Levi, and Dana Ron. Tolerant junta testing and the connection to submodular optimization and function isomorphism. In *Proceedings of the 29th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2113–2132, 2018.

10 Andrea Campagna, Alan Guo, and Ronitt Rubinfeld. Local reconstructors and tolerant testers for connectivity and diameter. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 411–424. Springer, 2013.

11 Anindya De, Elchanan Mossel, and Joe Neeman. Junta correlation is testable. In *Proceedings of the 60th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 1549–1563, 2019.

12 Irit Dinur. The PCP theorem by gap amplification. *Journal of the ACM*, 54(3):12, 2007.

13 Irit Dinur and Omer Reingold. Assignment Testers: Towards a Combinatorial Proof of the PCP Theorem. *SIAM Journal on Computing*, 36(4):975–1024, 2006.

14 Kashyap Dixit, Sofya Raskhodnikova, Abhradeep Thakurta, and Nithin M. Varma. Erasure-Resilient Property Testing. *SIAM Journal on Computing*, 47(2):295–329, 2018.

15 Eldar Fischer and Lance Fortnow. Tolerant versus intolerant testing for Boolean properties. *Theory of Computing*, 2(9):173–183, 2006.

16 Eldar Fischer and Ilan Newman. Testing versus estimation of graph properties. *SIAM Journal on Computing*, 37(2):482–501, 2007.

17 Eldar Fischer, Ilan Newman, and Jiří Sgall. Functions that have read-twice constant width branching programs are not necessarily testable. *Random Structures & Algorithms*, 24(2):175–193, 2004.

18 Oded Goldreich. *Computational complexity - A conceptual perspective*. Cambridge University Press, 2008.

19 Oded Goldreich, Shafi Goldwasser, and Dana Ron. Property testing and its connection to learning and approximation. *Journal of the ACM*, 45(4):653–750, 1998.

20 Oded Goldreich and Or Meir. A small gap in the gap amplification of assignment testers, 2007. In ECCC, 2007, TR05-46, Comment 3.

21 Oded Goldreich and Luca Trevisan. Three theorems regarding testing graph properties. *Random Structures & Algorithms*, 23(1):23–57, 2003.

22 Ellis Horowitz. A fast method for interpolation using preconditioning. *Information Processing Letters*, 1(4):157–163, 1972.

23 Swastik Kopparty and Shubhangi Saraf. Tolerant linearity testing and locally testable codes. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM)*, pages 601–614. Springer, 2009.

24 Amit Levi and Erik Waingarten. Lower Bounds for Tolerant Junta and Unateness Testing via Rejection Sampling of Graphs. In *Proceedings of the 10th Innovations in Theoretical Computer Science Conference (ITCS)*, pages 52:1–52:20, 2019.

25 Florence Jessie MacWilliams and Neil James Alexander Sloane. *The theory of error-correcting codes*, volume 16. Elsevier, 1977.

26 Michal Parnas, Dana Ron, and Ronitt Rubinfeld. Tolerant property testing and distance approximation. *Journal of Computer and System Sciences*, 72(6):1012–1042, 2006.

27 Sofya Raskhodnikova, Noga Ron-Zewi, and Nithin M. Varma. Erasures vs. Errors in Local Decoding and Property Testing. In *Proceedings of the 10th Innovations in Theoretical Computer Science Conference (ITCS)*, pages 63:1–63:21, 2019.

28 Omer Reingold, Guy N. Rothblum, and Ron D. Rothblum. Constant-round interactive proofs for delegating computation. In *Proceedings of the 48th ACM Symposium on the Theory of Computing (STOC)*, pages 49–62, 2016.

29 Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.
30 Daniel A. Spielman. Linear-time encodable and decodable error-correcting codes. *IEEE Transactions on Information Theory*, 42(6):1723–1731, 1996.

Kronecker Powers of Tensors and Strassen's Laser Method

Austin Conner
Department of Mathematics, Texas A&M University, College Station, TX 77843-3368, USA
connerad@math.tamu.edu

Joseph M. Landsberg
Department of Mathematics, Texas A&M University, College Station, TX 77843-3368, USA
https://www.math.tamu.edu/~jml/
jml@math.tamu.edu

Fulvio Gesmundo
QMATH, University of Copenhagen, Universitetsparken 5, 2100 Copenhagen O., Denmark
fulges@math.ku.dk

Emanuele Ventura
Department of Mathematics, Texas A&M University, College Station, TX 77843-3368, USA
eventura@math.tamu.edu

Abstract

We answer a question, posed implicitly in [18, §11], [11, Rem. 15.44] and explicitly in [9, Problem 9.8], showing the border rank of the Kronecker square of the little Coppersmith-Winograd tensor is the square of the border rank of the tensor for all $q > 2$, a negative result for complexity theory. We further show that when $q > 4$, the analogous result holds for the Kronecker cube. In the positive direction, we enlarge the list of explicit tensors potentially useful for the laser method. We observe that a well-known tensor, the 3×3 determinant polynomial regarded as a tensor, $\det_3 \in \mathbb{C}^9 \otimes \mathbb{C}^9 \otimes \mathbb{C}^9$, could potentially be used in the laser method to prove the exponent of matrix multiplication is two. Because of this, we prove new upper bounds on its Waring rank and rank (both 18), border rank and Waring border rank (both 17), which, in addition to being promising for the laser method, are of interest in their own right. We discuss "skew" cousins of the little Coppersmith-Winograd tensor and indicate why they may be useful for the laser method. We establish general results regarding border ranks of Kronecker powers of tensors, and make a detailed study of Kronecker squares of tensors in $\mathbb{C}^3 \otimes \mathbb{C}^3 \otimes \mathbb{C}^3$.

2012 ACM Subject Classification Theory of computation → Algebraic complexity theory

Keywords and phrases Matrix multiplication complexity, Tensor rank, Asymptotic rank, Laser method

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.10

Funding *Joseph M. Landsberg*: supported by NSF grant AF-1814254.
Fulvio Gesmundo: acknowledges support VILLUM FONDEN via the QMATH Centre of Excellence (Grant no. 10059).

1 Introduction

The exponent ω of matrix multiplication is defined as

$$\omega := \inf\{\tau \mid \mathbf{n} \times \mathbf{n} \text{ matrices may be multiplied using } O(\mathbf{n}^\tau) \text{ arithmetic operations}\}.$$

The exponent is a fundamental constant governing the complexity of the basic operations in linear algebra. It is conjectured that $\omega = 2$. There was steady progress in the research for upper bounds from 1968 to 1988: after Strassen's famous $\omega < 2.81$ [39], Bini et al. [8], using border rank (see below), showed $\omega < 2.78$, then a major breakthrough by Schönhage [36] (the asymptotic sum inequality) was used to show $\omega < 2.55$, then Strassen's *laser method*

was introduced and used by Strassen to show $\omega < 2.48$, and refined by Coppersmith and Winograd to show $\omega < 2.3755$ [18]. Then there was no progress until 2011 when a series of improvements by Stothers, Williams, and Le Gall [38, 45, 33] lowered the upper bound to the current state of the art $\omega < 2.373$.

Strassen's 1968 result is obtained by an explicit algorithm for multiplying matrices. This algorithm is more efficient than the standard one in practical implementation as soon as the size of the matrices is around 1000×1000, see [6]. Bini et al. exhibited a matrix multiplication algorithm that is in principle implementable exactly (at a cost of a constant size blow-up which does not effect the exponent) but as presented is only a sequence of algorithms that limits to an exact one. This gave rise to the notion of *border rank* to describe this phenomenon. To explain border rank, it is best to adopt the language of *tensors*.

A bilinear map $b : \mathbb{C}^{\mathbf{a}} \times \mathbb{C}^{\mathbf{b}} \to \mathbb{C}^{\mathbf{c}}$ may be regarded as a trilinear form $\widehat{b} : \mathbb{C}^{\mathbf{a}} \times \mathbb{C}^{\mathbf{b}} \times \mathbb{C}^{\mathbf{c}} \to \mathbb{C}$ defined by $\widehat{b}(X, Y, \alpha) = \alpha \cdot b(X, Y)$ where $b(X, Y)$ is regarded as a column vector of $\mathbb{C}^{\mathbf{c}}$, α is regarded as a row vector and $\cdot$ is the row-column multiplication. In this language, matrix multiplication, as a trilinear map, becomes $M_{\mathbf{l},\mathbf{m},\mathbf{n}}(X, Y, Z) = \mathrm{trace}(XYZ)$, where X, Y, Z are matrices of size $\mathbf{l} \times \mathbf{m}$, $\mathbf{m} \times \mathbf{n}$ and $\mathbf{n} \times \mathbf{l}$, respectively. It is known [11, §14.1] that the complexity of performing a bilinear map is captured, up to a factor of four, by the *tensor rank* of the corresponding tensor. Thus, this geometric quantity may be used to determine ω.

Let A, B, C be fixed vector spaces. A tensor $T \in A \otimes B \otimes C$ has *rank one* if $T = a \otimes b \otimes c$ for some $a \in A$, $b \in B$, $c \in C$. The *rank* of T, denoted $\mathbf{R}(T)$, is the smallest r such that T is sum of r rank one tensors. The *border rank* of T, denoted $\underline{\mathbf{R}}(T)$, is the smallest r such that T is the limit of a sequence of rank r tensors. One has $\underline{\mathbf{R}}(T) \leq \mathbf{R}(\mathbf{T})$ and the inequality can be strict: Let $T = a_1 \otimes b_1 \otimes c_2 + a_1 \otimes b_2 \otimes c_1 + a_2 \otimes b_1 \otimes c_1$, then $\mathbf{R}(\mathbf{T}) = 3$ and $\underline{\mathbf{R}}(T) = 2$ as

$$T = \lim_{t \to 0} \frac{1}{t}[(a_1 + ta_2) \otimes (b_1 + tb_2) \otimes (c_1 + tc_2) - a_1 \otimes b_1 \otimes c_1].$$

Bini [7] proved that the border rank of matrix multiplication also captures its complexity. More precisely,

$$\omega = \inf\{\tau : \underline{\mathbf{R}}(M_{\langle \mathbf{n} \rangle}) \in O(\mathbf{n}^{\tau})\}.$$

Schönhage's advance comes from his discovery that it can be more efficent to perform two matrix multiplications together than one at a time. For tensors $T \in A \otimes B \otimes C$ and $T' \in A' \otimes B' \otimes C'$, define a new tensor $T \oplus T' \in (A \oplus A') \otimes (B \oplus B') \otimes (C \otimes C')$ whose computation is equivalent to computing T and T'. He gave explicit examples of matrix multiplication tensors where $\underline{\mathbf{R}}(T \oplus T') \ll \underline{\mathbf{R}}(T) + \underline{\mathbf{R}}(T')$. To explain how he exploited this we need some more definitions:

Given $T \in A \otimes B \otimes C$ and $T' \in A' \otimes B' \otimes C'$, the *Kronecker product* of T and T' is the tensor $T \boxtimes T' := T \otimes T' \in (A \otimes A') \otimes (B \otimes B') \otimes (C \otimes C')$, regarded as 3-way tensor. Given $T \in A \otimes B \otimes C$, the *Kronecker powers* of T are $T^{\boxtimes N} \in A^{\otimes N} \otimes B^{\otimes N} \otimes C^{\otimes N}$, defined iteratively. We have $\mathbf{R}(T \boxtimes T') \leq \mathbf{R}(T)\mathbf{R}(T')$, and similarly for border rank. The matrix multiplication tensor has the following important self-reproducing property: $M_{\langle \mathbf{l},\mathbf{m},\mathbf{n} \rangle} \boxtimes M_{\langle \mathbf{l}'\mathbf{,m}'\mathbf{n}' \rangle} = M_{\langle \mathbf{ll}'\mathbf{,mm}'\mathbf{,nn}' \rangle}$.

Given $T, T' \in A \otimes B \otimes C$, we say that T *degenerates* to T' if $T' \in \overline{GL(A) \times GL(B) \times GL(C) \cdot T}$, the closure of the orbit of T under the natural action of $GL(A) \times GL(B) \times GL(C)$ on $A \otimes B \otimes C$. Here $GL(A)$ denote the general linear group of invertible linear maps $A \to A$. Border rank is upper semi-continuous under degeneration: if T' is a degeneration of T, then $\underline{\mathbf{R}}(T') \leq \underline{\mathbf{R}}(T)$.

Schönhage observed that if one takes a high Kronecker power of $(M_{\langle \mathbf{l},\mathbf{m},\mathbf{n}\rangle} \oplus M_{\langle \mathbf{l}',\mathbf{m}',\mathbf{n}'\rangle})$, that because of the reproducing property, it will be a sum of matrix multiplication tensors, some of them quite large. One can then perform a degeneration to obtain a single very large matrix multiplication tensor and exploit the strict sub-additivity to get an upper bound on this large matrix multiplication tensor. This is his celebrated *asymptotic sum inequality*.

After Schönhage, Strassen realized that the starting tensor need not be a sum of matrix multiplication tensors, as long as some high power of it degenerates to a large matrix multiplication tensor. This gave rise to his *laser method*, where the starting tensor "resembles" the sum of disjoint matrix multiplication tensors. All upper bounds since 1984 are obtained via Strassen's laser method. The best starting tensor for Strassen's method (so far) was discovered by Coppersmith and Winograd, the *big Coppersmith-Winograd tensor*.

In 2014 [4] gave an explanation for the limited progress since 1988, followed by further explanations in [3, 2, 13, 1]: there are limitations to the laser method applied to the big Coppersmith-Winograd tensor and other auxiliary tensors. These limitations are referred to as *barriers*. Our main motivation is to eventually overcome these barriers via auxilary tensors that avoid them, or, failing that, to prove structural results explaining the failure. We deal with the *little Coppersmith-Winograd tensor*, which was known to potentially avoid the barriers and a new series of tensors that are skew versions of the little Coppersmith-Winograd tensor that we show also potentially avoid the barriers. We are interested in two kinds of barriers: to proving the exponent is two, and barriers to proving the exponent is less than 2.3.

▶ Remark 1. A different approach to upper bounds was introduced by Cohn and Umans [15] using the Fourier-transform on finite groups. One can show $\omega < 2.41$ by this method [13, 14].

Definitions and notation

Let A, B, C be complex vector spaces. We will work with tensors in $A \otimes B \otimes C$. Let $GL(A)$ denote the general linear group of invertible linear maps $A \to A$. Unless stated otherwise, we write $\{a_i\}$ for a basis of A, and similarly for bases of B and C. Often we assume that all tensors involved in the discussion belong to the same space $A \otimes B \otimes C$; this is not restrictive, since we may re-embed the spaces A, B, C into larger spaces whenever it is needed. We say that two tensors are *isomorphic* if they are the same up to a change of bases in A,B and C.

One may define border rank in terms of degeneration: $\underline{\mathbf{R}}(T) \leq r$ if and only if $M_{\langle 1\rangle}^{\oplus r}$ degenerates to T. The *border subrank* of T, denoted $\underline{\mathbf{Q}}(T)$, is the largest q such that T degenerates to $M_{\langle 1\rangle}^{\oplus q}$.

The *asymptotic rank* of T is $\underset{\sim}{\mathbf{R}}(T) := \lim_{N\to\infty} \underline{\mathbf{R}}(T^{\boxtimes N})^{1/N}$. Thus $\omega = \log_{\mathbf{m}} \underset{\sim}{\mathbf{R}}(M_{\langle \mathbf{m}\rangle})$ for any $\mathbf{m} > 2$. The *asymptotic subrank* of T is $\underset{\sim}{\mathbf{Q}}(T) = \lim_{N\to\infty} \underline{\mathbf{Q}}(T^{\boxtimes N})^{1/N}$. These limits exist and are finite, see [41]. Moreover $\underset{\sim}{\mathbf{R}}(T) \leq \underline{\mathbf{R}}(T)$ and $\underset{\sim}{\mathbf{Q}}(T) \geq \underline{\mathbf{Q}}(T)$.

A tensor $T \in A \otimes B \otimes C$ is *concise* if the induced linear maps $T_A : A^* \to B \otimes C$, $T_B : B^* \to A \otimes C$, $T_C : C^* \to A \otimes B$ are injective. We say that a concise tensor $T \in \mathbb{C}^m \otimes \mathbb{C}^m \otimes \mathbb{C}^m$ has *minimal rank* (resp. *minimal border rank*) if $\mathbf{R}(T) = m$ (resp. $\underline{\mathbf{R}}(T) = m$).

The laser method and the Coppersmith-Winograd tensors

So far, the best upper bounds for ω have been obtained using the laser method applied to the big Coppersmith-Winograd tensor, which is

$$T_{CW,q} := \sum_{j=1}^{q} a_0 \otimes b_j \otimes c_j + a_j \otimes b_0 \otimes c_j + a_j \otimes b_j \otimes c_0 + \\ + a_0 \otimes b_0 \otimes c_{q+1} + a_0 \otimes b_{q+1} \otimes c_0 + a_{q+1} \otimes b_0 \otimes c_0 \in (\mathbb{C}^{q+2})^{\otimes 3}.$$

It was used to obtain the current world record $\omega < 2.373$ and all bounds below $\omega < 2.41$. The barrier identified in [4] said that $T_{CW,q}$ cannot be used to prove $\omega < 2.3$ using the standard laser method, and a geometric identification of this barrier in terms of asymptotic subrank was given in [13]: $\underset{\sim}{\mathbf{Q}}(M_{\langle \mathbf{n} \rangle}) = \mathbf{n}^2$ which is maximal, which is used to show any tensor with non-maximal asymptotic subrank cannot be used to prove $\omega = 2$ by the laser method, and Strassen [43] had shown $\underset{\sim}{\mathbf{Q}}(T_{CW,q})$ is non-maximal.

The second best tensor for the laser method so far has been the little Coppersmith-Winograd tensor, which is

$$T_{cw,q} := \sum_{j=1}^{q} a_0 \otimes b_j \otimes c_j + a_j \otimes b_0 \otimes c_j + a_j \otimes b_j \otimes c_0 \in (\mathbb{C}^{q+1})^{\otimes 3}. \tag{1}$$

The laser method was used to prove the following inequality:

▶ **Theorem 2.** *[18] For all k and q,*

$$\omega \leq \log_q(\frac{4}{27}(\underline{\mathbf{R}}(T_{cw,q}^{\boxtimes k}))^{\frac{3}{k}}). \tag{2}$$

More precisely, the ingredients needed for the proof but not the statement appears in [18]. It was pointed out in [11, Ex. 15.24] that the statement holds with $\underline{\mathbf{R}}(T_{cw,q}^{\boxtimes k})^{\frac{3}{k}}$ replaced by $\underset{\sim}{\mathbf{R}}(T_{cw,q})^3$ and the proof implicitly uses (2). The equation does appear in [29, Thm. 5.1.5.1].

An easy calculation shows $\underline{\mathbf{R}}(T_{cw,q}) = q+2$ (one more than minimal). Applying Theorem 2 to $T_{cw,8}$ with $k = 1$ gives $\omega \leq 2.41$ [18]. Theorem 2 shows that, unlike $T_{CW,q}$, $T_{cw,2}$ is not subject to the barriers of [4, 3, 2, 13] for proving $\omega = 2$, and $T_{cw,q}$, for $2 \leq q \leq 10$ are not subject to the barriers for proving $\omega < 2.3$. Thus, if any Kronecker power of $T_{cw,q}$ for $2 \leq q \leq 10$ is strictly sub-multiplicative, one can get new upper bounds on ω, and if it were the case that $\underset{\sim}{\mathbf{R}}(T_{cw,2}) = 3$, one would obtain that ω is two. Hence the questions:

▶ **Question 3.** *For given q, k, what is $\underline{\mathbf{R}}(T_{cw,q}^{\boxtimes k})$? Does there exist $q \in \{2, \ldots, 10\}$ and $k \in \mathbb{N}$ such that $\underline{\mathbf{R}}(T_{cw,q}^{\boxtimes k}) < [\underline{\mathbf{R}}(T_{cw,q})]^k$?*

▶ Remark 4. Although we know little about asymptotic rank of explicit tensors beyond matrix multiplication, most tensors have asymptotic rank less than their border rank: For all tensors $T \in \mathbb{C}^m \otimes \mathbb{C}^m \otimes \mathbb{C}^m$, with $m > 3$, outside a set of measure zero (more precisely, for all tensors outside a proper subvariety), Lickteig showed that $\underline{\mathbf{R}}(T) = \lceil \frac{m^3}{3m-2} \rceil$ [35]. Strassen [42, Lemma 3.5] implicitly showed that for any $T \in \mathbb{C}^m \otimes \mathbb{C}^m \otimes \mathbb{C}^m$, if $\underline{\mathbf{R}}(T) > m^{\frac{2\omega}{3}} > m^{1.6}$, then $\underset{\sim}{\mathbf{R}}(T) < \underline{\mathbf{R}}(T)$. It is worth recalling Strassen's proof: any $T \in \mathbb{C}^m \otimes \mathbb{C}^m \otimes \mathbb{C}^m$ is a degeneration of $M_{\langle 1,m,m \rangle} \in \mathbb{C}^{m^2} \otimes \mathbb{C}^m \otimes \mathbb{C}^m$, so $T^{\boxtimes 3}$ is a degeneration of $M_{\langle m^2,m^2,m^2 \rangle} = M_{\langle 1,m,m \rangle} \boxtimes M_{\langle m,1,m \rangle} \boxtimes M_{\langle m,m,1 \rangle}$. In particular $\underline{\mathbf{R}}(T^{\boxtimes 3}) \leq \underline{\mathbf{R}}(M_{\langle m^2,m^2,m^2 \rangle})$ and $\underset{\sim}{\mathbf{R}}(T)^3 = \underset{\sim}{\mathbf{R}}(T^{\boxtimes 3}) \leq \underset{\sim}{\mathbf{R}}(M_{\langle m^2,m^2,m^2 \rangle}) = m^{2\omega}$, so $\underset{\sim}{\mathbf{R}}(T) \leq m^{\frac{2\omega}{3}}$. Since $\omega < 2.4$ we conclude. In particular, note that $\underset{\sim}{\mathbf{R}}(T) \leq m^{1.6}$ for all $T \in \mathbb{C}^m \otimes \mathbb{C}^m \otimes \mathbb{C}^m$.

2 Results

2.1 Lower bounds for Kronecker powers of $T_{cw,q}$

We address Problem 9.8 in [9], which was motivated by Theorem 2: Is $\underline{\mathbf{R}}(T_{cw,q}^{\boxtimes 2}) < (q+2)^2$? We give an almost complete answer:

▶ **Theorem 5.** *For all $q > 2$, $\underline{\mathbf{R}}(T_{cw,q}^{\boxtimes 2}) = (q+2)^2$, and $15 \leq \underline{\mathbf{R}}(T_{cw,2}^{\boxtimes 2}) \leq 16$.*

We also examine the Kronecker cube:

▶ **Theorem 6.** *For all $q > 4$, $\underline{\mathbf{R}}(T_{cw,q}^{\boxtimes 3}) = (q+2)^3$.*

Proofs are given in §4.

Proposition 25 below, combined with the proofs of Theorems 6 and 5, implies

▶ **Corollary 7.** *For all $q > 4$ and all N,*

$$\underline{\mathbf{R}}(T_{cw,q}^{\boxtimes N}) \geq (q+1)^{N-3}(q+2)^3,$$

and $\underline{\mathbf{R}}(T_{cw,4}^{\boxtimes N}) \geq 36 \times 5^{N-2}$.

Previously, in [10] it had been shown that $\underline{\mathbf{R}}(T_{cw,q}^{\boxtimes N}) \geq (q+1)^N + 2^N - 1$ for all q, N, whereas the bound in Corollary 7 is $(q+1)^N + 3(q+1)^{N-1} + 3(q+1)^{N-2} + (q+1)^{N-3}$.

Previous to this work one might have hoped to prove $\omega < 2.3$ simply by using the Kronecker square of, e.g., $T_{cw,7}$. Now, the smallest possible calculation to give a new upper bound on ω from a tensor that has been used in the laser method would be e.g., to prove the fourth Kronecker power of a small Coppersmith-Winograd tensor achieves the lower bound of Corollary 7 (which we do not expect to happen). Of course, one could work directly with the matrix multiplication tensor, in which case the cheapest possible upper bound would come from proving the border rank of the 6×6 matrix multiplication tensor equaled its known lower bound of 69 from [30].

The following corollary of Theorems 5 and 6 is immediate by the semi-continuity property of border rank, as most tensors of border rank $m+1$ in $\mathbb{C}^m \otimes \mathbb{C}^m \otimes \mathbb{C}^m$ may be degnerated to $T_{cw,m-1}$, in fact the set of tensors of border rank $m+1$ is an orbit closure and $T_{cw,m-1}$ lives on the boundary of the orbit.

▶ **Corollary 8.** *Most tensors $T \in \mathbb{C}^m \otimes \mathbb{C}^m \otimes \mathbb{C}^m$ of border rank $m+1$, satisfy $\underline{\mathbf{R}}(T^{\boxtimes 2}) = \underline{\mathbf{R}}(T)^2 = (m+1)^2$ for $m \geq 4$ and $\underline{\mathbf{R}}(T^{\boxtimes 3}) = \underline{\mathbf{R}}(T)^3 = (m+1)^3$ for $m \geq 6$. More precisely all tensors outside of a Zariski closed subset of the set of tensors of border rank $m+1$. In particular the set of such is of full measure.*

2.2 A skew cousin of $T_{cw,q}$

In light of the negative results for complexity theory above, one might try to find a better tensor than $T_{cw,q}$ that is also not subject to the barriers. In [16], when q is even, we introduced a skew cousin of the big Coppersmith-Winograd tensor, which has the largest symmetry group of any tensor in its space satisfying a natural genericity condition. However this tensor turns out not to be useful for the laser method. Inspired by it, we introduce a skew cousin of the small Coppersmith-Winograd tensor when q is even:

$$T_{skewcw,q} := \sum_{j=1}^{q} a_0 \otimes b_j \otimes c_j + a_j \otimes b_0 \otimes c_j + \sum_{\xi=1}^{\frac{q}{2}} (a_\xi \otimes b_{\xi+\frac{q}{2}} - a_{\xi+\frac{q}{2}} \otimes b_\xi) \otimes c_0 \in (\mathbb{C}^{q+1})^{\otimes 3}. \quad (3)$$

In the language of [11], $T_{skewcw,q}$ has the same "block structure" as $T_{cw,q}$, which immediately implies Theorem 2 also holds for $T_{skewcw,q}$:

▶ **Theorem 9.** *For all k,*

$$\omega \leq \log_q\left(\frac{4}{27}(\underline{\mathbf{R}}(T_{skewcw,q}^{\boxtimes k}))^{\frac{3}{k}}\right). \quad (4)$$

In particular, the known barriers do not apply to $T_{skewcw,2}$ for proving $\omega = 2$ and to any $T_{skewcw,q}$ for $q \leq 10$ for proving $\omega < 2.3$. Unfortunately, we have

▶ **Proposition 10.** $\underline{\mathbf{R}}(T_{skewcw,q}) \geq q + 3$.

Proposition 10 is proved in §4.

Thus $\underline{\mathbf{R}}(T_{skewcw,q}) > \underline{\mathbf{R}}(T_{cw,q})$ for all q, in particular $\underline{\mathbf{R}}(T_{skewcw,2}) = 5$, as for all $T \in \mathbb{C}^3 \otimes \mathbb{C}^3 \otimes \mathbb{C}^3$, $\underline{\mathbf{R}}(T) \leq 5$.

However, unlike $T_{cw,2}$, substantial strict sub-multiplicativity holds for the Kronecker square of $T_{skewcw,2}$:

▶ **Theorem 11.** $\underline{\mathbf{R}}(T^{\boxtimes 2}_{skewcw,2}) \leq 17$.

▶ Remark 12. Regarding border rank strict submultiplicativity of Kronecker powers for other explicit tensors, little is known. For matrix multiplication, the only explicit drop under a Kronecker power that is known to our knowledge is [37]: $\underline{\mathbf{R}}(M^{\boxtimes 2}_{\langle 2\rangle}) \leq 46 < 49$.

Previous to this work, we are only aware of one class of tensors other than $M_{\langle 2\rangle}$ for which any bound on the Kronecker squares other than the trivial $\underline{\mathbf{R}}(T^{\boxtimes 2}) \leq \underline{\mathbf{R}}(T)^2$ is known. In [12], they show that

$$\begin{aligned} T_{CGJ,m} :=& a_1 \otimes b_1 \otimes c_1 + a_2 \otimes b_2 \otimes c_2 + a_3 \otimes b_3 \otimes c_3 + (\textstyle\sum_{i=1}^{3} a_i) \otimes (\sum_{j=1}^{3} b_j) \otimes (\sum_{j=1}^{3} c_k) \\ &+ 2(a_1 + a_2) \otimes (b_1 + b_3) \otimes (c_2 + c_3) + a_3 \otimes (\textstyle\sum_{s=4}^{m} b_s \otimes c_s) \in \mathbb{C}^3 \otimes \mathbb{C}^m \otimes \mathbb{C}^m \end{aligned}$$

satisfies $\underline{\mathbf{R}}(T_{CGJ,m}) = m + 2$ and $\underline{\mathbf{R}}(T^{\otimes 2}_{CGJ,m}) \leq (m+2)^2 - 1$. Of course, for any tensor T, $\underline{\mathbf{R}}(T^{\boxtimes 2}) \leq \underline{\mathbf{R}}(T^{\otimes 2})$, and strict inequality, e.g., with $M_{\langle 2\rangle}$ is possible. This is part of a general theory in [12] for constructing examples with a drop of one when the last non-trivial secant variety is a hypersurface.

We also show

▶ **Theorem 13.** $\mathbf{R}(T^{\boxtimes 2}_{skewcw,2}) \leq 18$.

Theorems 11 and 13 are proved in §5.

2.3 Two familiar tensors with no known laser method barriers

Recall from above that either $\underset{\sim}{\mathbf{R}}(T_{cw,2}) = 3$ or $\underset{\sim}{\mathbf{R}}(T_{skewcw,2}) = 3$ would imply $\omega = 2$.

Let $\det_3 \in (\mathbb{C}^9)^{\otimes 3}$ and $\mathrm{perm}_3 \in (\mathbb{C}^9)^{\otimes 3}$ be the 3×3 determinant and permanent polynomials considered as tensors. We observe that if either of these has minimal asymptotic rank, then $\omega = 2$: either $\underset{\sim}{\mathbf{R}}(\det_3) = 9$ or $\underset{\sim}{\mathbf{R}}(\mathrm{perm}_3) = 9$ would imply $\omega = 2$. This observation is an immediate consequence of the following lemma:

▶ **Lemma 14.** *We have the following isomorphisms of tensors:*

$T^{\boxtimes 2}_{cw,2} \cong \mathrm{perm}_3$

$T^{\boxtimes 2}_{skewcw,2} \cong \det_3$.

Lemma 14 is proved in §3.

Lemma 14 thus implies Theorems 11 and 13 may be restated as saying $\underline{\mathbf{R}}(\det_3) \leq 17$ and $\mathbf{R}(\det_3) \leq 18$. Although it is not necessarily relevant for complexity theory, we actually prove stronger statements, which are important for geometry:

A symmetric tensor $T \in S^3\mathbb{C}^m \subseteq \mathbb{C}^m \otimes \mathbb{C}^m \otimes \mathbb{C}^m$ has *Waring rank one* if $T = a \otimes a \otimes a$ for some $a \in \mathbb{C}^3$. The *Waring rank* of T, denoted $\mathbf{R}_S(T)$, is the smallest r such that T is sum of r tensors of Waring rank one. The *Waring border rank* of T, denoted $\underline{\mathbf{R}}_S(T)$, is the smallest r such that T is limit of a sequence of tensors of Waring rank r.

We actually show:

▶ **Theorem 15.** $\mathbf{R}_S(\det_3) \leq 18$.

and

▶ **Theorem 16.** $\underline{\mathbf{R}}_S(\det_3) \leq 17$.

Proofs are respectively given in §5.1 and §5.2.

2.4 Generic tensors in $\mathbb{C}^3 \otimes \mathbb{C}^3 \otimes \mathbb{C}^3$

▶ Remark 17. A generic tensor in $\mathbb{C}^3 \otimes \mathbb{C}^3 \otimes \mathbb{C}^3$ has border rank five. Our numerical experiments suggest that for all $T \in \mathbb{C}^3 \otimes \mathbb{C}^3 \otimes \mathbb{C}^3$:

$$\underline{\mathbf{R}}(T^{\boxtimes 2}) \leq 22 < 25. \tag{5}$$

This is obtained by starting with a tensor whose entries are obtained from making draws according to a uniform distribution on $[-1, 1]$, and proving the result for that tensor. The data to perform an example of this computation is available in Appendix A at `http://www.math.tamu.edu/~jml/CGLVkronsupp.html`.

▶ **Problem 18.** *Write a proof of* (5). *Even better, give a geometric proof.*

The inequality (5) is not too surprising because $\mathbb{C}^3 \otimes \mathbb{C}^3 \otimes \mathbb{C}^3$ is *secant defective*, in the sense that by a dimension count, one would expect the maximum border rank of a tensor to be 4, but the actual maximum is 5. This means that for a generic tensor, there is a 8 parameter family of rank 5 decompositions, and it is not surprising that the naïve 64-parameter family of decompositions of the square might have decompositions of lower border rank on the boundary.

3 Symmetries of tensors and the proof of Lemma 14

3.1 Symmetry groups of tensors and polynomials

The group $GL(A) \times GL(B) \times GL(C)$ acts naturally on $A \otimes B \otimes C$. The map $\Phi : GL(A) \times GL(B) \times GL(C) \to GL(A \otimes B \otimes C)$ has a two dimensional kernel $\ker \Phi = \{(\lambda \mathrm{Id}_A, \mu \mathrm{Id}_B, \nu \mathrm{Id}_C) : \lambda\mu\nu = 1\} \simeq (\mathbb{C}^*)^2$.

In particular, the group $(GL(A) \times GL(B) \times GL(C))/(\mathbb{C}^*)^{\times 2}$ is naturally identified with a subgroup of $GL(A \otimes B \otimes C)$. Given $T \in A \otimes B \otimes C$, the *symmetry group* of a tensor T is the stabilizer of T in $(GL(A) \times GL(B) \times GL(C))/(\mathbb{C}^*)^{\times 2}$, that is

$$G_T := \{g \in (GL(A) \times GL(B) \times GL(C))/(\mathbb{C}^*)^{\times 2} \mid \; g \cdot T = T\}. \tag{6}$$

Let $\mathfrak{S}_k$ be the permutation group on k elements. We record the following observation:

▶ **Proposition 19.** *For any tensor $T \in A \otimes B \otimes C$, $G_{T^{\boxtimes N}} \supset \mathfrak{S}_N$.*

Proof. Write $T^{\boxtimes N} = \sum_{I,J,K} T^{I,J,K} a_I \otimes b_J \otimes c_K$ where $I = (i_1, \ldots, i_N)$, $a_I = a_{i_1} \otimes \cdots \otimes a_{i_N}$, etc.. For $\sigma \in \mathfrak{S}_N$, define $\sigma \cdot T = \sum_{I,J,K} T^{IJK} a_{\sigma(I)} \otimes b_{\sigma(J)} \otimes c_{\sigma(K)}$. Since $T^{IJK} = T^{i_1 j_1 k_1} \cdots T^{i_N j_N k_N}$ we have $T^{IJK} = T^{\sigma(I),\sigma(J),\sigma(K)}$ and we conclude. ◀

▶ Remark 20. For a symmetric tensor (equivalently, a homogeneous polynomial), $T \in S^d A$, one may also consider the symmetry group $G^s_T := \{g \in GL(A) \mid g \cdot T = T\}$ where the action is the induced action on polynomials.

3.2 Proof of Lemma 14

Write $(-1)^\sigma$ for the sign of a permutation σ. Let

$$\det_3 = \sum_{\sigma,\tau\in\mathfrak{S}_3} (-1)^\tau a_{\sigma(1)\tau(1)} \otimes b_{\sigma(2)\tau(2)} \otimes c_{\sigma(3)\tau(3)},$$
$$\mathrm{perm}_3 = \sum_{\sigma,\tau\in\mathfrak{S}_3} a_{\sigma(1)\tau(1)} \otimes b_{\sigma(2)\tau(2)} \otimes c_{\sigma(3)\tau(3)}$$

be the 3×3 determinant and permanent polynomials regarded as tensors in $\mathbb{C}^9\otimes\mathbb{C}^9\otimes\mathbb{C}^9$.

Proof of Lemma 14. After the change of basis $\tilde{b}_0 := -b_0$ and $\tilde{c}_1 := c_2$, $\tilde{c}_2 := -c_1$, we obtain

$$\begin{aligned} T_{skewcw,2} = {} & a_0\otimes b_1\otimes \tilde{c}_2 - a_0\otimes b_2\otimes \tilde{c}_1 + a_2\otimes \tilde{b}_0\otimes c_1 \\ & - a_1\otimes \tilde{b}_0\otimes \tilde{c}_2 + a_1\otimes b_2\otimes c_0 - a_2\otimes b_1\otimes c_0. \end{aligned}$$

This shows that, after identifying the three spaces, $T_{skewcw,2} = a_0\wedge a_1\wedge a_2$ is the unique (up to scale) skew-symmetric tensor in $\mathbb{C}^3\otimes\mathbb{C}^3\otimes\mathbb{C}^3$. In particular, $T_{skewcw,2}$ is invariant under the action of SL_3 on $\mathbb{C}^3\otimes\mathbb{C}^3\otimes\mathbb{C}^3$.

Consequently, the stabilizer of $T_{skewcw,2}^{\boxtimes 2}$ in $GL(\mathbb{C}^9)$ contains (and in fact equals) $SL_3^{\times 2}\rtimes\mathbb{Z}_2$. This is the stabilizer of the determinant polynomial $\det_3$. Since the determinant is characterized by its stabilizer, we conclude.

The tensor $T_{cw,2}$ is symmetric and, after identifying the three spaces, it coincides with $a_0(a_1^2+a_2^2)\in S^3\mathbb{C}^3$. After the change of basis $\tilde{a}_1 := a_1+a_2$, $\tilde{a}_2 := a_1-a_2$, we obtain $T_{cw,2} = a_0\tilde{a}_1\tilde{a}_2\in S^3\mathbb{C}^3$ is the square-free monomial of degree 3. The stabilizer of $T_{cw,2}$ under the action of GL_3 on $S^3\mathbb{C}^3$ is $\mathbb{T}_3^{SL}\rtimes\mathfrak{S}_3$, where $\mathbb{T}_3^{SL}$ denotes the torus of diagonal matrices with determinant one, and $\mathfrak{S}_3$ acts permuting the three basis elements.

Consequently, the stabilizer of $T_{cw,2}^{\boxtimes 2}$ in $GL(\mathbb{C}^9)$ contains (and in fact equals) $(\mathbb{T}_3^{SL}\rtimes\mathfrak{S}_3)^{\times 2}\rtimes\mathbb{Z}_2$. This is the stabilizer of the permanent polynomial perm_3. Since the permanent is characterized by its stabilizer, we conclude. ◀

▶ Remark 21. For the reader's convenience, here are short proofs that $\det_m, \mathrm{perm}_m$ are characterized by their stabilizers: To see $\det_m$ is characterized by its stabilizer, note that $SL_m\times SL_m = SL(E)\times SL(F)$ acting on $S^m(E\otimes F)$ decomposes it to

$$\bigoplus_{|\pi|=m} S_\pi E\otimes S_\pi F$$

which is multiplicity free, with the only trivial module $S_{1^m}E\otimes S_{1^m}F = \Lambda^m E\otimes\Lambda^m F$. To see that perm_m is characterized by its stabilizer, take the above decomposition and consider the $\mathbb{T}^{SL(E)}\times\mathbb{T}^{SL(F)}$-invariants, these are the weight zero spaces $(S_\pi E)_0\otimes(S_\pi F)_0$. By [24], one has the decomposition of the weight zero spaces as $\mathfrak{S}_m^E\times\mathfrak{S}_m^F$-modules to $(S_\pi E)_0\otimes(S_\pi F)_0 = [\pi]_E\otimes[\pi]_F$. The only such that is trivial is the case $\pi=(d)$.

▶ Remark 22. Even Kronecker powers of $T_{skewcw,2}$ are invariant under $SL_3^{\times 2k}$, and coincide, up to a change of basis, with the *Pascal determinants* (see, e.g., [27, §8.3]), $T_{skewcw,2}^{\boxtimes 2k} = PasDet_{k,3}$, the unique, up to scale, tensor spanning $(\Lambda^3\mathbb{C}^3)^{\otimes 2k}\subset S^3((\mathbb{C}^3)^{\otimes 2k})$.

▶ Remark 23. One can regard the 3×3 determinant and permanent as trilinear maps $\mathbb{C}^3\times\mathbb{C}^3\times\mathbb{C}^3\to\mathbb{C}$, where the three copies of $\mathbb{C}^3$ are the first, second and third column of a 3×3 matrix. From this point of view, the trilinear map given by the determinant is $T_{skewcw,2}$ as a tensor and the one given by the permanent is $T_{cw,2}$ as a tensor. This perspective, combined with the notion of product rank, immediately provides the upper bounds $\underline{\mathbf{R}}(\mathrm{perm}_3)\leq 16$ (which is also a consequence of Lemma 14) and $\underline{\mathbf{R}}(\det_3)\leq 20$, see [19, 26].

▶ Remark 24. A similar change of basis as the one performed in the second part of proof of Lemma 14 shows that, up to a change of basis, $T_{skewcw,q} \in \Lambda^3\mathbb{C}^{q+1}$. In particular, its even Kronecker powers are symmetric tensors.

4 Koszul flattenings and lower bounds for Kronecker powers

In this section we review Koszul flattenings, prove a result on propagation of Koszul flattening lower bounds under Kronecker products, and prove Theorems 5 and 6. We give two proofs of Theorem 5 because the first is elementary and method of the second generalizes to give the proof of Theorem 6.

4.1 Definition

Respectively fix bases $\{a_i\}$, $\{b_j\}$, $\{c_k\}$ of the vector spaces A, B, C. Given $T = \sum_{ijk} T^{ijk} a_i \otimes b_j \otimes c_k \in A \otimes B \otimes C$, define the linear map

$$\begin{aligned} T_A^{\wedge p} : \Lambda^p A \otimes B^* &\to \Lambda^{p+1} A \otimes C \\ X \otimes \beta &\mapsto \textstyle\sum_{ijk} T^{ijk} \beta(b_j)(a_i \wedge X) \otimes c_k. \end{aligned}$$

Then [31, Proposition 4.1.1] states

$$\underline{\mathbf{R}}(T) \geq \frac{\operatorname{rank}(T_A^{\wedge p})}{\binom{\dim(A)-1}{p}}. \tag{7}$$

This type of lower bound has a long history: in general, one takes the space $A \otimes B \otimes C$ and linearly embeds it into a large space of matrices. Then if a rank one tensor maps to a rank q matrix, a rank r tensor maps to a rank at most rq matrix, so the size $rq+1$ minors give equations testing for border rank r. In this case the size of the matrices is $\binom{\mathbf{a}}{p}\mathbf{b} \times \binom{\mathbf{a}}{p+1}\mathbf{c}$ and a rank one tensor maps to a matrix of rank $\binom{\mathbf{a}-1}{p}$. Here $\mathbf{a} = \dim A$, $\mathbf{b} = \dim B$ and $\mathbf{c} = \dim C$.

In practice, one takes a subspace $A'^* \subseteq A^*$ of dimension $2p+1$ and restricts T (considered as a trilinear form) to $A'^* \times B^* \times C^*$ to get an optimal bound, so the denominator $\binom{\dim(A)-1}{p}$ is replaced by $\binom{2p}{p}$ in (7). Write $\phi : A \to A/(A'^*)^\perp =: A'$ for the projection onto the quotient: the corresponding Koszul flattening map gives a lower bound for $\underline{\mathbf{R}}(\phi(T))$, which, by linearity, is a lower bound for $\underline{\mathbf{R}}(T)$. The case $p = 1$ is equivalent to Strassen's equations [40]. There are numerous expositions of Koszul flattenings and their generalizations, see, e.g., [27, §7.3], [5, §7.2], [20], [28, §2.4], or [21].

Proof of Proposition 10. Write $q = 2u$. Fix a space $A' = \langle e_0, e_1, e_2 \rangle$. Define $\phi : A \to A'$ by

$$\begin{aligned} &\phi(a_0) = e_0, \\ &\phi(a_i) = e_1 \quad \text{for } i = 1, \dots, u, \\ &\phi(a_s) = e_2 \quad \text{for } s = u+1, \dots, q. \end{aligned}$$

As an element of $\Lambda^3 A$, we have $T_{skewcw,q} = a_0 \wedge \sum_{i=1}^{u} a_i \wedge a_{u+i}$.

We prove that if $T = T_{skewcw,q}$ then $\operatorname{rank}(T_{A'}^{\wedge 1}) = 2(q+2)+1$. This provides the lower bound $\underline{\mathbf{R}}(T) \geq \left\lceil \frac{2(q+2)+1}{2} \right\rceil = q+3$.

We record the images via $T_{A'}^{\wedge 1}$ of a basis of $A' \otimes B^*$. Fix the range of $i = 1, \ldots, u$:

$$\begin{aligned}
T_{A'}^{\wedge 1}(e_0 \otimes \beta_0) &= (e_0 \wedge e_1) \otimes \textstyle\sum_{i=1}^{u} c_{u+i} - (e_0 \wedge e_2) \otimes \sum_{i=1}^{u} c_i,\\
T_{A'}^{\wedge 1}(e_0 \otimes \beta_i) &= (e_0 \wedge e_2) \otimes c_0,\\
T_{A'}^{\wedge 1}(e_0 \otimes \beta_{u+i}) &= (e_0 \wedge e_1) \otimes c_0,\\
T_{A'}^{\wedge 1}(e_1 \otimes \beta_0) &= (e_1 \wedge e_2) \otimes \textstyle\sum_{i=1}^{u} c_{u+i},\\
T_{A'}^{\wedge 1}(e_1 \otimes \beta_i) &= (e_0 \wedge e_1) \otimes c_{u+i} + e_1 \wedge e_2 \otimes c_0,\\
T_{A'}^{\wedge 1}(e_1 \otimes \beta_{u+i}) &= e_0 \wedge e_1 \otimes c_i,\\
T_{A'}^{\wedge 1}(e_2 \otimes \beta_0) &= (e_1 \wedge e_2) \otimes \textstyle\sum_{i=1}^{u} c_i,\\
T_{A'}^{\wedge 1}(e_2 \otimes \beta_i) &= e_0 \wedge e_2 \otimes c_{u+i},\\
T_{A'}^{\wedge 1}(e_2 \otimes \beta_{u+i}) &= (e_0 \wedge e_2) \otimes c_i - e_1 \wedge e_2 \otimes c_0.
\end{aligned}$$

Notice that the image of $\sum_{i=1}^{u}(e_1 \otimes \beta_i) - \sum_{i=1}^{u}(e_2 \otimes \beta_{u+i}) - e_0 \otimes \beta_0$ is (up to scale) $e_1 \wedge e_2 \otimes c_0$.

This shows that the image of $T_{A'}^{\wedge 1}$ contains

$$\Lambda^2 A' \otimes c_0 + e_1 \wedge e_2 \otimes \langle \textstyle\sum_{i=1}^{u} c_i, \sum_{i=1}^{u} c_{u+i} \rangle + \langle e_0 \wedge e_1, e_0 \wedge e_2 \rangle \otimes \langle c_1, \ldots, c_q \rangle.$$

These summands are in disjoint subspaces, so we conclude

$$\operatorname{rank}(T_{A'}^{\wedge 1}) \geq 3 + 2 + 2q = 2q + 5. \qquad \blacktriangleleft$$

4.2 Propagation of lower bounds under Kronecker products

A tensor $T \in A \otimes B \otimes C$, with $\dim B = \dim C$ is 1_A-*generic* if $T(A^*) \subseteq B \otimes C$ contains a full rank element. Here is a partial multiplicativity result for Koszul flattening lower bounds under Kronecker products:

▶ **Proposition 25.** *Let $T_1 \in A_1 \otimes B_1 \otimes C_1$ with $\dim B_1 = \dim C_1$ be a tensor with a Koszul flattening lower bound for border rank $\underline{\mathbf{R}}(T) \geq r$ given by ${T_1}_{A_1}^{\wedge p}$ (possibly after a restriction ϕ). Let $T_2 \in A_2 \otimes B_2 \otimes C_2$, with $\dim B_2 = \dim C_2 = \mathbf{b}_2$ be 1_{A_2}-generic. Then*

$$\underline{\mathbf{R}}(T_1 \boxtimes T_2) \geq \left\lceil \frac{\operatorname{rank}({T_1}_{A_1}^{\wedge p}) \cdot \mathbf{b}_2}{\binom{2p}{p}} \right\rceil. \tag{8}$$

In particular, if $\frac{\operatorname{rank}({T_1}_{A_1}^{\wedge p})}{\binom{2p}{p}} \in \mathbb{Z}$, then $\underline{\mathbf{R}}(T_1 \boxtimes T_2) \geq r\mathbf{b}_2$.

Proof. After applying a restriction ϕ as described above, we may assume $\dim A_1 = 2p + 1$ so that the lower bound for T_1 is

$$\underline{\mathbf{R}}(T_1) \geq \left\lceil \frac{\operatorname{rank}({T_1}_{A_1}^{\wedge p})}{\binom{2p}{p}} \right\rceil.$$

Let $\alpha \in A_2^*$ be such that $T(\alpha) \in B_2 \otimes C_2$ has full rank $\mathbf{b}_2$, which exists by 1_{A_2}-genericity. Define $\psi : A_1 \otimes A_2 \to A_1$ by $\psi = \operatorname{Id}_{A_1} \otimes \alpha$ and set $\Psi := \psi \otimes \operatorname{Id}_{B_1 \otimes C_1 \otimes B_2 \otimes C_2}$. Then $(\Psi(T_1 \boxtimes T_2)_{A_1}^{\wedge p})$ provides the desired lower bound.

Indeed, the linear map $(\Psi(T_1 \boxtimes T_2)_{A_1}^{\wedge p})$ coincides with ${T_1}_{A_1}^{\wedge p} \boxtimes T_1(\alpha)$. Since matrix rank is multiplicative under Kronecker product, we conclude. $\blacktriangleleft$

4.3 First proof of Theorem 5

When $q = 3$, the result is true by a direct calculation using the $p = 2$ Koszul flattening with a sufficiently generic $\mathbb{C}^5 \subset A^*$, which is left to the reader. In what follows we treat the case $q > 3$.

Write $a_{ij} = a_i \otimes a_j \in A^{\otimes 2}$ and similarly for $B^{\otimes 2}$ and $C^{\otimes 2}$. Let $A' = \langle e_0, e_1, e_2\rangle$ and define the linear map $\phi_2 : A^{\otimes 2} \to A'$ via

$$\begin{aligned}
&\phi_2(a_{00}) = \phi_2(a_{01}) = \phi_2(a_{10}) = e_0 + e_1,\\
&\phi_2(a_{11}) = e_0,\\
&\phi_2(a_{02}) = \phi_2(a_{20}) = e_1 + e_2\\
&\phi_2(a_{33}) = \phi_2(a_{21}) = e_2\\
&\phi_2(a_{0i}) = \phi_2(a_{i0}) = e_1 \quad \text{for } i = 3, \dots, q\\
&\phi_2(a_{ij}) = 0 \quad \text{for all other pairs } (i, j).
\end{aligned}$$

Write $T_q := T^{\boxtimes 2}_{cw,q}|_{A^{*\prime}\otimes B^{*\otimes 2}\otimes C^{*\otimes 2}}$. Consider the $p = 1$ Koszul flattening $(T_q)^{\wedge 1}_{A'} : A' \otimes B^{\otimes 2^*} \to \Lambda^2 A' \otimes C^{\otimes 2}$.

We are going to prove that $\operatorname{rank}((T_q)^{\wedge 1}_{A'}) = 2(q+2)^2$. This provides the lower bound $\underline{\mathbf{R}}(T^{\boxtimes 2}_{cw,q}) \geq (q+2)^2$ and equality follows because of the submultiplicativity properties of border rank under Kronecker product.

We proceed by induction on q. When $q = 4$ one does a direct computation with the $p = 1$ Koszul flattening, which is left to the reader, and which provides the base of the induction.

Write $W_j = a_0 \otimes b_j \otimes c_j + a_i \otimes b_0 \otimes c_j + a_i \otimes b_i \otimes c_0$. Then $T_{cw,q} = \sum_{j=1}^{q} W_j$, so that $T^{\boxtimes 2}_{cw,q} = \sum_{ij} W_i \boxtimes W_j$.

If $q \geq 4$, write $T_{cw,q} = T_{cw,q-1} + W_q$, so $T^{\boxtimes 2}_{cw,q} = T^{\boxtimes 2}_{cw,q-1} + T_{cw,q-1} \boxtimes W_q + W_q \boxtimes T_{cw,q-1} + W_q \boxtimes W_q$. Let $S_q = (T_{cw,q-1} \boxtimes W_q + W_q \boxtimes T_{cw,q-1} + W_q \boxtimes W_q)|_{A'\otimes B^{*\otimes 2}\otimes C^{*\otimes 2}}$.

Write $U_1 = A' \otimes \langle \beta_{ij} : i, j = 0, \dots, q-1\rangle$ and $U_2 = A' \otimes \langle \beta_{qi}, \beta_{iq} : i = 0, \dots, q\rangle$ so that $U_1 \oplus U_2 = A' \otimes B^{\otimes 2*}$. Similarly, define $V_1 = \Lambda^2 A' \otimes \langle c_{ij} : i, j = 0, \dots, q-1\rangle$ and $V_2 = \Lambda^2 A' \otimes \langle c_{qi}, c_{iq} : i = 0, \dots, q\rangle$, so that $V_1 \oplus V_2 = \Lambda^2 A' \otimes C^{\otimes 2}$. Observe that $(T_{q-1})^{\wedge 1}_{A'}$ is identically 0 on U_2 and its image is contained in V_1. Moreover, the image of U_1 under $(S_q)^{\wedge 1}_{A'}$ is contained in V_1. Representing the Koszul flattening in blocks, we have

$$(T_{q-1})^{\wedge 1}_{A'} = \begin{bmatrix} M_{11} & 0 \\ 0 & 0 \end{bmatrix} \qquad (S_q)^{\wedge 1}_{A'} = \begin{bmatrix} N_{11} & N_{12} \\ 0 & N_{22} \end{bmatrix}$$

therefore $\operatorname{rank}((T_q)^{\wedge 1}_{A'}) \geq \operatorname{rank}(M_{11} + N_{11}) + \operatorname{rank}(N_{22})$.

First, we prove that $\operatorname{rank}(M_{11} + N_{11}) \geq \operatorname{rank}(M_{11}) = 2(q+1)^2$. This follows by a degeneration argument. Consider the linear map given by pre-composing the Koszul flattening with the projection onto U_1. Its rank is semicontinuous under degeneration. Since $T^{\boxtimes 2}_{cw,q}$ degenerates to $T^{\boxtimes 2}_{cw,q-1}$, we deduce $\operatorname{rank}(M_{11} + N_{11}) \geq \operatorname{rank}(M_{11})$. The equality $\operatorname{rank}(M_{11}) = 2(q+1)^2$ follows by the induction hypothesis.

We show that $\operatorname{rank}(N_{22}) = 2(2q+3)$. The following equalities are modulo V_1. Moreover, each equality is modulo the tensors resulting from the previous ones. They are all straightforward applications of the Koszul flattening map, which in these cases, can always

be performed on some copy of $W_i \boxtimes W_j$.

$$
\begin{aligned}
&(S_q)^{\wedge 1}_{A'}(e_1 \otimes \beta_{qj}) \equiv e_1 \wedge e_0 \otimes c_{qj} \quad \text{for } j = 3, \ldots, q\\
&(S_q)^{\wedge 1}_{A'}(e_1 \otimes \beta_{jq}) \equiv e_1 \wedge e_0 \otimes c_{jq} \quad \text{for } j = 3, \ldots, q\\
&(S_q)^{\wedge 1}_{A'}(e_0 \otimes \beta_{3q}) \equiv e_0 \wedge e_1 \otimes c_{0q}\\
&(S_q)^{\wedge 1}_{A'}(e_0 \otimes \beta_{q3}) \equiv e_0 \wedge e_1 \otimes c_{q0}\\
&(S_q)^{\wedge 1}_{A'}(e_0 \otimes \beta_{q1}) \equiv e_0 \wedge e_1 \otimes c_{q1}\\
&(S_q)^{\wedge 1}_{A'}(e_0 \otimes \beta_{1q}) \equiv e_0 \wedge e_1. \otimes c_{1q}
\end{aligned}
$$

Further passing modulo $\langle e_0 \wedge e_1 \rangle \otimes C$, we obtain

$$
\begin{aligned}
&(S_q)^{\wedge 1}_{A'}(e_0 \otimes \beta_{0q}) \equiv e_0 \wedge e_2 \otimes c_{2q}\\
&(S_q)^{\wedge 1}_{A'}(e_0 \otimes \beta_{q0}) \equiv e_0 \wedge e_2 \otimes c_{q2}\\
&(S_q)^{\wedge 1}_{A'}(e_0 \otimes \beta_{q2}) \equiv e_0 \wedge e_2 \otimes c_{0q}\\
&(S_q)^{\wedge 1}_{A'}(e_0 \otimes \beta_{2q}) \equiv e_0 \wedge e_2 \otimes c_{q0}\\
&(S_q)^{\wedge 1}_{A'}(e_1 \otimes \beta_{20}) \equiv e_1 \wedge e_2 \otimes c_{0q}\\
&(S_q)^{\wedge 1}_{A'}(e_1 \otimes \beta_{02}) \equiv e_1 \wedge e_2 \otimes c_{q0}\\
&(S_q)^{\wedge 1}_{A'}(e_1 \otimes \beta_{q0}) \equiv e_1 \wedge e_2 \otimes c_{2q}\\
&(S_q)^{\wedge 1}_{A'}(e_1 \otimes \beta_{0q}) \equiv e_1 \wedge e_2 \otimes c_{q2},
\end{aligned}
$$

and modulo the above,

$$
\begin{aligned}
&(S_q)^{\wedge 1}_{A'}(e_2 \otimes \beta_{qj}) \equiv e_2 \wedge (e_0 + e_1) \otimes c_{qj} \quad \text{for } j = 3, \ldots, q\\
&(S_q)^{\wedge 1}_{A'}(e_2 \otimes \beta_{jq}) \equiv e_2 \wedge (e_0 + e_1) \otimes c_{jq} \quad \text{for } j = 3, \ldots, q\\
&(S_q)^{\wedge 1}_{A'}(e_2 \otimes \beta_{q1}) \equiv e_2 \wedge (e_0 + e_1) \otimes c_{q1}\\
&(S_q)^{\wedge 1}_{A'}(e_2 \otimes \beta_{1q}) \equiv e_2 \wedge (e_0 + e_1) \otimes c_{1q}.
\end{aligned}
$$

Finally passing modulo $\langle e_1 \wedge e_2 \rangle$, we have

$$
\begin{aligned}
&(S_q)^{\wedge 1}_{A'}(e_2 \otimes \beta_{q0}) \equiv e_2 \wedge e_0 \otimes c_{q1}\\
&(S_q)^{\wedge 1}_{A'}(e_2 \otimes \beta_{0q}) \equiv e_2 \wedge e_0 \otimes c_{1q}.
\end{aligned}
$$

All the tensors listed above are linearly independent. Adding all the contributions together, we obtain

$$\operatorname{rank}((S_q)^{\wedge 1}_{A'}) = [2(q-3)+1] + 4 + 8 + 2 + [2(q-3)+1] + 4 = 2(2q+3)$$

as desired, and since $2(q+3)^2 = 2(q+1)^2 + 2(2q+3)$, this concludes the proof. ◀

4.4 A short detour on computing ranks of equivariant maps

We briefly explain how to exploit Schur's Lemma (see, e.g., [23, §1.2]) to compute the rank of an equivariant linear map. This is a standard technique, used extensively e.g., in [32, 25] and will reduce the proof of Theorems 5 and 6 to the computation of the ranks of specific linear maps in small dimension.

Let G be a reductive group. In the proof of Theorems 5 and 6, G will be the product of symmetric groups. Let Λ_G be the set of irreducible representations of G. For $\lambda \in \Lambda_G$, let W_λ denote the corresponding irreducible module.

Suppose U, V are two representations of G. Write $U = \bigoplus_{\lambda\in\Lambda_G} W_\lambda^{\oplus m_\lambda}$, $V = \bigoplus_{\lambda\in\Lambda_G} W_\lambda^{\oplus \ell_\lambda}$, where m_λ is the multiplicity of W_λ in U and ℓ_λ is the multiplicity of W_λ in V. The direct summand corresponding to λ is called the *isotypic component* of type λ.

Let $f : U \to V$ be a G-equivariant map. By Schur's Lemma [23, §1.2], f decomposes as $f = \oplus f_\lambda$, where $f_\lambda : W_\lambda^{\oplus m_\lambda} \to W_\lambda^{\oplus \ell_\lambda}$. Consider multiplicity spaces M_λ, L_λ with $\dim M_\lambda = m_\lambda$ and $\dim L_\lambda = \ell_\lambda$ so that $W_\lambda^{\oplus m_\lambda} \simeq M_\lambda \otimes W_\lambda$ as a G-module, where G acts trivially on M_λ and similarly $W_\lambda^{\oplus \ell_\lambda} \simeq L_\lambda \otimes W_\lambda$.

By Schur's Lemma, the map $f_\lambda : M_\lambda \otimes W_\lambda \to L_\lambda \otimes W_\lambda$ decomposes as $f_\lambda = \phi_\lambda \otimes \mathrm{Id}_{[\lambda]}$, where $\phi_\lambda : M_\lambda \to L_\lambda$. Thus $\mathrm{rank}(f)$ is uniquely determined by $\mathrm{rank}(\phi_\lambda)$ for $\lambda \in \Lambda_G$.

The ranks $\mathrm{rank}(\phi_\lambda)$ can be computed via restrictions of f. For every λ, fix a vector $w_\lambda \in W_\lambda$, so that $M_\lambda \otimes \langle w_\lambda\rangle$ is a subspace of U. Here and in what follows, for a subset $X \subset V$, $\langle X\rangle$ denotes the span of X. Then the rank of the restriction of f to $M_\lambda \otimes \langle w_\lambda\rangle$ coincides with the rank of ϕ_λ.

We conclude

$$\mathrm{rank}(f) = \textstyle\sum_\lambda \mathrm{rank}(\phi_\lambda)\cdot \dim W_\lambda.$$

The second proof of Theorem 5 and proof of Theorem 6 will follow the algorithm described above, exploiting the symmetries of $T_{cw,q}$. Consider the action of the symmetry group $\mathfrak{S}_q$ on $A \otimes B \otimes C$ defined by permuting the basis elements with indices $\{1, \dots, q\}$. More precisely, a permutation $\sigma \in \mathfrak{S}_q$ induces the linear map defined by $\sigma(a_i) = a_{\sigma(i)}$ for $i = 1, \dots, q$ and $\sigma(a_0) = a_0$. The group $\mathfrak{S}_q$ acts on B, C similarly, and the simultaneous action on the three factors defines an $\mathfrak{S}_q$-action on $A \otimes B \otimes C$. The tensor $T_{cw,q}$ is invariant under this action.

4.5 Second Proof of Theorem 5

When $q = 3$, as before, one uses the $p = 2$ Koszul flattening with a sufficiently generic $\mathbb{C}^5 \subset A^*$.

For $q \geq 4$, we apply the $p = 1$ Koszul flattening map to the same restriction of $T_{cw,q}^{\boxtimes 2}$ as the first proof, although to be consistent with the code at the website, we use the less appealing swap of the roles of a_2 and a_3 in the projection ϕ.

Since $T_{cw,q}$ is invariant under the action of $\mathfrak{S}_q$, $T_{cw,q}^{\boxtimes 2}$ is invariant under the action of $\mathfrak{S}_q \times \mathfrak{S}_q$, acting on $A^{\otimes 2} \otimes B^{\otimes 2} \otimes C^{\otimes 2}$. Let $\Gamma := \mathfrak{S}_{q-3} \times \mathfrak{S}_{q-3}$ where $\mathfrak{S}_{q-3}$ is the permutation group on $\{4, \dots, q\}$, so $T_{cw,q}^{\boxtimes 2}$ is invariant under the action of Γ. Note that Γ acts trivially on A', so $(T_q)^{\wedge 1}_{A'}$ is Γ-equivariant, because in general, Koszul flattenings are equivariant under the product of the three general linear groups, which is $GL(A') \times GL(B^{\otimes 2}) \times GL(C^{\otimes 2})$ in our case. (We remind the reader that $T_q := T_{cw,q}^{\boxtimes 2}|_{A^{*\prime}\otimes B^{*\otimes 2}\otimes C^{*\otimes 2}}$.) We now apply the method described in §4.4 to compute $\mathrm{rank}((T_q)^{\wedge 1}_{A'})$.

Let [triv] denote the trivial $\mathfrak{S}_{q-3}$-representation and let V denote the standard representation, that is the Specht module associated to the partition $(q-4, 1)$ of $q-3$. We have $\dim[\mathrm{triv}] = 1$ and $\dim V = q-4$. (When $q = 4$ only the trivial representation appears.)

The spaces B, C are isomorphic as $\mathfrak{S}_{q-3}$-modules and they decompose as $B = C = [\mathrm{triv}]^{\oplus 5} \oplus V$. After fixing a 5-dimensional multiplicity space $\mathbb{C}^5$ for the trivial isotypic component, we write $B^* = C = \mathbb{C}^5 \otimes [\mathrm{triv}] \oplus V$. To distinguish the two $\mathfrak{S}_{q-3}$-actions, we write $B \otimes B = ([\mathrm{triv}]_L^{\oplus 5} \oplus V_L) \otimes ([\mathrm{triv}]_R^{\oplus 5} \oplus V_R)$.

Thus,

$$\begin{aligned} B^{*\otimes 2} = C^{\otimes 2} =& (\mathbb{C}^5 \otimes [\mathrm{triv}]_L \oplus V_L) \otimes (\mathbb{C}^5 \otimes [\mathrm{triv}]_R \oplus V_R) \\ =& (\mathbb{C}^5 \otimes \mathbb{C}^5) \otimes ([\mathrm{triv}]_L \otimes [\mathrm{triv}]_R) \quad \oplus \\ & \mathbb{C}^5 \otimes ([\mathrm{triv}]_L \otimes V_R) \quad \oplus \\ & \mathbb{C}^5 \otimes (V_L \otimes [\mathrm{triv}]_R) \quad \oplus \\ & (V_L \otimes V_R). \end{aligned}$$

Write $W_1, \ldots, W_4$ for the four irreducible representations in the decomposition above and let $M_1, \ldots, M_4$ be the four corresponding multiplicity spaces.

Recall from [22] that a basis of V is given by standard Young tableaux of shape $(q-4, 1)$ (with entries in $4, \ldots, q$ for consistency with the action of $\mathfrak{S}_{q-3}$); let w_{std} be the vector corresponding to the standard tableau having $4, 6, \ldots, q$ in the first row and 5 in the second row. We refer to [22, §7] for the straightening laws of the tableaux. Let w_{triv} be a generator of the trivial representation [triv].

For each of the four isotypic components in the decomposition above, we fix a vector $w_i \in W_i$ and explicitly realize the subspaces $M_i \otimes \langle w_i \rangle$ of $B^{*\otimes 2}$ as follows:

W_i	w_i	$\dim M_i$	$M_i \otimes \langle w_i \rangle$
$[\mathrm{triv}]_L \otimes [\mathrm{triv}]_R$	$w_{\mathrm{triv}} \otimes w_{\mathrm{triv}}$	25	$\begin{smallmatrix}\langle \beta_{ij} : i,j=0,\ldots,3\rangle \oplus \\ \langle \sum_{j=4}^{q} \beta_{ij} : i=0,\ldots,3\rangle \oplus \\ \langle \sum_{i=4}^{q} \beta_{ij} : j=0,\ldots,3\rangle \oplus \\ \langle \sum_{i,j=4}^{q} \beta_{ij} \rangle\end{smallmatrix}$
$[\mathrm{triv}]_L \otimes V_R$	$w_{\mathrm{triv}} \otimes w_{std}$	5	$\begin{smallmatrix}\langle \beta_{i5}-\beta_{i4} : i=0,\ldots,3\rangle \oplus \\ \langle \sum_{i=4}^{q} (\beta_{i5}-\beta_{i4}) \rangle\end{smallmatrix}$
$V_L \otimes [\mathrm{triv}]_R$	$w_{std} \otimes w_{\mathrm{triv}}$	5	$\begin{smallmatrix}\langle \beta_{5j}-\beta_{4j} : j=0,\ldots,3\rangle \oplus \\ \langle \sum_{j=4}^{q} (\beta_{5j}-\beta_{4j}) \rangle\end{smallmatrix}$
$V_L \otimes V_R$	$w_{std} \otimes w_{std}$	1	$\langle \beta_{55} - \beta_{45} - \beta_{54} + \beta_{44} \rangle$.

The subspaces in $C^{\otimes 2}$ are realized similarly.

Since $(T^{\boxtimes 2}_{cw,q})^{\wedge 1}_{A'}$ is Γ-equivariant, by Schur's Lemma, it has the isotypic decomposition $(T^{\boxtimes 2}_{cw,q})^{\wedge 1}_{A'} = f_1 \oplus f_2 \oplus f_3 \oplus f_4$, where

$$f_i : A' \otimes (M_i \otimes W_i) \to \Lambda^2 A' \otimes W_i.$$

As explained in §4.4, it suffices to compute the ranks of the four restrictions $\Phi_i : A' \otimes M_i \otimes \langle w_i \rangle \to \Lambda^2 A' \otimes M_i \otimes \langle w_i \rangle$.

Using the bases presented in the fourth column of the table above, we write down the four matrices representing the maps $\Phi_1, \ldots, \Phi_4$.

The map Φ_4 is represented by the 3×3 matrix

$$\begin{pmatrix} -1 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix},$$

so $\mathrm{rank}(\Phi_4) = 2$.

The map Φ_2 is represented by the 15×15 matrix (here $q' = q - 3$)

$$\begin{pmatrix}
0 & -1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
-1 & -1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & q' & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\
-1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & q' & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix}$$

We prove the matrix above and those that follow are as asserted for all q in §7. The proof goes by showing each entry must be a low degree polynomial in q, and then one simply tests enough small cases to fix the polynomials. Thus $\operatorname{rank}(\Phi_2) = 12$, and similarly for Φ_3.

The map Φ_1 is represented by a 75×75 matrix that can be presented in block form

$$\begin{pmatrix}
-X & Y & 0 \\
-Z & 0 & Y \\
0 & -Z & X
\end{pmatrix}$$

with X the matrix

$$\begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix},$$

Y the matrix

$$\begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & q' & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & q' & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & q' & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
0 & q' & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & q' & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & q' & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & q' & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & q' & 0 & 0 & 0 & 1
\end{pmatrix},$$

and Z the matrix

$$\left(\begin{array}{cccccccccccccccccccccccccc}
0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&1&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&1&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0\\
0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&1&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0\\
1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0
\end{array}\right)$$

We compute $\operatorname{rank}(\Phi_1) = 72$.

Although these matrices are of fixed size, they are obtained via intermediate tensors whose dimensions depend on q, which created a computational challenge. Two ways of addressing the challenge (including the one utilized in the code) are explained in §7.

The relevant matrices and the implementation of the method of §7 to justify them for all q, together with the code for the computation of their ranks are available at the website `http://www.math.tamu.edu/~jml/CGLVkronsupp.html`, Appendix D. The ranks are bounded below by taking a matrix M (which has some entries depending linearly on q), multiplying it on the left by a rectangular matrix P whose entries are rational functions of q, and on the right by a rectangular matrix Q whose entries are constant, to obtain a square matrix PMQ that is upper triangular with ± 1 on the diagonal, and thus its rank is its dimension. Finally one checks that the least common multiple of the denominators of the entries of P has no integral solution when $q > 4$.

Adding all the contributions gives

$$\begin{aligned}\operatorname{rank}(T^{\wedge 1}_{A'}) &= 2\cdot\dim(V\otimes V) + 12\cdot\dim([\mathrm{triv}]\otimes V) +\\ &\quad + 12\cdot\dim(V\otimes[\mathrm{triv}]) + 72\cdot\dim([\mathrm{triv}]\otimes[\mathrm{triv}]) =\\ &= 2\cdot(q-4)^2 + 12\cdot(q-4) + 12\cdot(q-4) + 72\cdot 1 = 2(q+2)^2.\end{aligned}$$

This concludes the proof of Theorem 5.

▶ Remark 26. One might have hoped to exploit the full symmetry group $\mathfrak{S}_q \times \mathfrak{S}_q$ to simplify the argument further. However there is no choice of a restriction map ψ which is $\mathfrak{S}_{q-s}\times\mathfrak{S}_{q-s}$-invariant for $s < 3$ that gives rise to a Koszul flattening map of sufficiently high rank to prove the result.

4.6 Proof of Theorem 6

We will use a Koszul flattening with $p = 2$, so we need a 5 dimensional subspace of $(A^*)^{\otimes 3}$. Let

$$A'^* := \left\langle \begin{array}{c} \alpha_{000},\\ \sum_{i=1}^{q}(\alpha_{i00}+\alpha_{0i0}+\alpha_{00i}),\\ \alpha_{001}+\alpha_{010}+\alpha_{012}+\alpha_{102}+\alpha_{110}+\alpha_{121}+\alpha_{200}+\alpha_{211},\\ \alpha_{022}+\alpha_{030}+\alpha_{031}+\alpha_{100}+\alpha_{103}-\alpha_{120}+\alpha_{210}+\alpha_{212}+\alpha_{300},\\ \alpha_{002}+\alpha_{004}+\alpha_{011}+\alpha_{014}+\alpha_{020}+\alpha_{023}+\alpha_{032}+\alpha_{040}+\alpha_{100}+\alpha_{122}+\alpha_{220}+\alpha_{303} \end{array} \right\rangle.$$

Write $\phi_3 : A^{\otimes 3} \to A'$ for the resulting projection map and, abusing notation, for the induced map $A^{\otimes 3} \otimes B^{\otimes 3} \otimes C^{\otimes 3} \to A' \otimes B^{\otimes 3} \otimes C^{\otimes 3}$. Write $T = \phi_3(T^{\boxtimes 3}_{cw,q})$, suppressing the q from the notation. Consider the Koszul flattening:

$$(T)^{\wedge 2}_{A'} : \Lambda^2 A' \otimes B^{*\otimes 2} \to \Lambda^3 A' \otimes C^{\otimes 2}.$$

We will show $\operatorname{rank}((T)^{\wedge 2}_{A'}) = 6(q+2)^3$, which implies $\underline{\mathbf{R}}(T^{\boxtimes 3}_{cw,q}) \geq (q+2)^3$.

In order to compute $\operatorname{rank}((T)^{\wedge 2}_{A'})$, we follow the same strategy as before. The code to generate these matrices is available at `www.math.tamu.edu/~jml/CGLVkronsupp.html`, Appendix D. The explanation of how we proved they are as asserted is outlined in §7.

The map $(T)^{\wedge 2}_{A'}$ is invariant under the action of $\Gamma = \mathfrak{S}_{q-4} \times \mathfrak{S}_{q-4} \times \mathfrak{S}_{q-4}$ where the first copy of $\mathfrak{S}_{q-4}$ permutes the basis elements with indices $5, \ldots, q$ of the first factors, and similarly for the other copies of $\mathfrak{S}_{q-4}$. Let [triv] be the trivial $\mathfrak{S}_{q-4}$-representation and let V be the standard representation, namely the Specht module associated to the partition $(q-5, 1)$. Here $\dim V = q-5$, so if $q = 5$, only the trivial representation appears.

The $\mathfrak{S}_{q-4}$-isotypic decomposition of B (and C) is $\mathbb{C}^6 \otimes [\text{triv}] \oplus V$ and this induces the decomposition of $B^{*\otimes 3} \simeq C^{\otimes 3}$ given by

$$\begin{aligned} B^{*\otimes 3} \simeq C^{\otimes 3} = &(\mathbb{C}^6)^{\otimes 3} \otimes ([\text{triv}]_1 \otimes [\text{triv}]_2 \otimes [\text{triv}]_3) \oplus \\ &(\mathbb{C}^6)^{\otimes 2} \otimes [([\text{triv}]_1 \otimes [\text{triv}]_2 \otimes V_3) \oplus \\ &([\text{triv}]_1 \otimes V_2 \otimes [\text{triv}]_3) \oplus \\ &(V_1 \otimes [\text{triv}]_2 \otimes [\text{triv}]_3)] \oplus \\ &(\mathbb{C}^6) \otimes [([\text{triv}]_1 \otimes V_2 \otimes V_3) \oplus \\ &(V_1 \otimes V_2 \otimes [\text{triv}]_3) \oplus \\ &(V_1 \otimes [\text{triv}]_2 \otimes V_3)] \oplus \\ &V_1 \otimes V_2 \otimes V_3 \end{aligned}$$

consisting of eight isotypic components. As in the previous proof, for each of the eight irreducible components W_i, we consider $w_i \in W_i$ and we compute the rank of the restriction to $\Lambda^2 A' \otimes M_i \otimes \langle w_i \rangle$ of the Koszul flattening; call this restriction Φ_i.

The ranks of the restrictions are recorded in the following table:

W_i	$\dim(\Lambda^2 A' \otimes M_i)$	$\operatorname{rank}(\Phi_i)$
$[\text{triv}]_1 \otimes [\text{triv}]_2 \otimes [\text{triv}]_3$	$6^3 \cdot \binom{5}{2} = 2160$	2058
$[\text{triv}]_1 \otimes [\text{triv}]_2 \otimes V_3$ (and permutations)	$6^2 \cdot \binom{5}{2} = 360$	294
$[\text{triv}]_1 \otimes V_2 \otimes V_3$ (and permutations)	$6 \cdot \binom{5}{2} = 60$	42
$V_1 \otimes V_2 \otimes V_3$	$\binom{5}{2} = 10$	6

The relevant matrices and the implementation of §7 to justify them for all q, with the code computing their ranks are available at `http://www.math.tamu.edu/~jml/CGLVkronsupp.html`, Appendix D. As before, the ranks are bounded below by taking a matrix M (which has some entries depending linearly on q), multiplying it on the left by a rectangular matrix P whose entries are rational functions of q, and on the right by a rectangular matrix Q whose

entries are constant, to obtain a square matrix PMQ that is upper triangular with ± 1 on the diagonal, and thus its rank is its dimension. Finally one checks that the least common multiple of the denominators of the entries of P has no integral solution when $q > 4$.

Adding all the contributions together, we obtain

$$\begin{aligned}\operatorname{rank}(T^{\wedge 2}_{A'}) =& 6 \cdot \dim(V \otimes V \otimes V) + \\ & 42 \cdot 3 \cdot \dim([\mathrm{triv}] \otimes V \otimes V) + \\ & 294 \cdot 3 \cdot \dim([\mathrm{triv}] \otimes [\mathrm{triv}] \otimes V) + \\ & 2058 \cdot \dim([\mathrm{triv}] \otimes [\mathrm{triv}] \otimes [\mathrm{triv}]) = 6 \cdot (q+2)^3.\end{aligned}$$

This concludes the proof.

5 Upper bounds for Waring rank and border rank of $\det_3$

5.1 Proof of Theorem 15

We give the rank 18 decomposition for $\det_3$ explicitly, as a collection of 18 linear forms on $\mathbb{C}^9 = \mathbb{C}^3 \otimes \mathbb{C}^3$ whose cubes add up to $\det_3$. The linear forms are given in coordinates recorded in the matrices below: the 3×3 matrix (ζ_{ij}) represents the linear forms $\sum_{ij} \zeta_{ij} x_{ij}$. This presentation highlights some of the symmetries of the decomposition.

Let $\vartheta = \exp(2\pi i/6)$ and let $\overline{\vartheta}$ be its inverse. The tensor $\det_3 = T^{\boxtimes 2}_{skewcw,2} = \det(x_{ij}) \in S^3(\mathbb{C}^3 \otimes \mathbb{C}^3)$ satisfies

$$\det_3 = \sum_1^{18} L_i^3$$

where $L_1, \ldots, L_{18}$ are the 18 linear forms given by the following coordinates:

$$L_1 = \begin{pmatrix} -\vartheta & 0 & 0 \\ 0 & -\frac{1}{3} & 0 \\ 0 & 0 & \overline{\vartheta} \end{pmatrix} \quad L_2 = \begin{pmatrix} -\overline{\vartheta} & 0 & 0 \\ 0 & -\frac{1}{3} & 0 \\ 0 & 0 & \vartheta \end{pmatrix} \quad L_3 = \begin{pmatrix} -\overline{\vartheta} & 0 & 0 \\ 0 & \frac{1}{3}\overline{\vartheta} & 0 \\ 0 & 0 & \overline{\vartheta} \end{pmatrix}$$

$$L_4 = \begin{pmatrix} -1 & 0 & 0 \\ 0 & 0 & -\overline{\vartheta} \\ 0 & -\frac{1}{3}\vartheta & 0 \end{pmatrix} \quad L_5 = \begin{pmatrix} \overline{\vartheta} & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -\frac{1}{3}\vartheta & 0 \end{pmatrix} \quad L_6 = \begin{pmatrix} \vartheta & 0 & 0 \\ 0 & 0 & -\vartheta \\ 0 & -\frac{1}{3}\vartheta & 0 \end{pmatrix}$$

$$L_7 = \begin{pmatrix} 0 & \frac{1}{3}\overline{\vartheta} & 0 \\ -\vartheta & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad L_8 = \begin{pmatrix} 0 & \frac{1}{3}\overline{\vartheta} & 0 \\ -\overline{\vartheta} & 0 & 0 \\ 0 & 0 & -\overline{\vartheta} \end{pmatrix} \quad L_9 = \begin{pmatrix} 0 & \frac{1}{3}\vartheta & 0 \\ -\overline{\vartheta} & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$L_{10} = \begin{pmatrix} 0 & -\frac{1}{3}\vartheta & 0 \\ 0 & 0 & \overline{\vartheta} \\ -1 & 0 & 0 \end{pmatrix} \quad L_{11} = \begin{pmatrix} 0 & -\frac{1}{3}\overline{\vartheta} & 0 \\ 0 & 0 & \vartheta \\ -1 & 0 & 0 \end{pmatrix} \quad L_{12} = \begin{pmatrix} 0 & \frac{1}{3} & 0 \\ 0 & 0 & -1 \\ -1 & 0 & 0 \end{pmatrix}$$

$$L_{13} = \begin{pmatrix} 0 & 0 & 1 \\ -1 & 0 & 0 \\ 0 & -\frac{1}{3} & 0 \end{pmatrix} \quad L_{14} = \begin{pmatrix} 0 & 0 & 1 \\ \overline{\vartheta} & 0 & 0 \\ 0 & \frac{1}{3}\vartheta & 0 \end{pmatrix} \quad L_{15} = \begin{pmatrix} 0 & 0 & 1 \\ \vartheta & 0 & 0 \\ 0 & \frac{1}{3}\overline{\vartheta} & 0 \end{pmatrix}$$

$$L_{16} = \begin{pmatrix} 0 & 0 & \overline{\vartheta} \\ 0 & -\frac{1}{3}\vartheta & 0 \\ 1 & 0 & 0 \end{pmatrix} \quad L_{17} = \begin{pmatrix} 0 & 0 & \overline{\vartheta} \\ 0 & -\frac{1}{3}\overline{\vartheta} & 0 \\ -\overline{\vartheta} & 0 & 0 \end{pmatrix} \quad L_{18} = \begin{pmatrix} 0 & 0 & \vartheta \\ 0 & -\frac{1}{3}\overline{\vartheta} & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

The equality can be verified by hand. A Macaulay2 file performing the calculation is available at `http://www.math.tamu.edu/~jml/CGLVkronsupp.html`, Appendix B. ◀

5.2 Proof of Theorem 16

As in the case of Theorem 15, we prove Theorem 16 by explicitly giving 17 linear forms, depending on a parameter t, whose cubes provide a border rank 17 expression for $\det_3$. The algebraic numbers involved are more complicated than in the previous case.

The result was achieved by numerical methods, which allowed us to sparsify the decomposition and ultimately determine the value of the coefficients. The linear forms in the decomposition are described below.

Consider

$$L_1(t) = \begin{pmatrix} z_1 & 0 & 0 \\ 0 & z_2 t & 0 \\ -1 & 0 & 0 \end{pmatrix} \quad L_2(t) = \begin{pmatrix} z_3 & 0 & 0 \\ z_4 & 0 & z_5 t \\ z_6 & 0 & 0 \end{pmatrix} \quad L_3(t) = \begin{pmatrix} -z_{36} & z_7 t & 0 \\ -z_{38} & 0 & -z_{39} t \\ 0 & 0 & t \end{pmatrix}$$

$$L_4(t) = \begin{pmatrix} 0 & 0 & t \\ -z_{34} & 0 & 0 \\ 0 & z_8 t & -z_{35} t \end{pmatrix} \quad L_5(t) = \begin{pmatrix} 0 & -z_{19} t & -z_{20} t \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{pmatrix} \quad L_6(t) = \begin{pmatrix} -z_{22} & z_9 t & 0 \\ -z_{23} & 0 & -z_{24} t \\ -z_{25} & 0 & 0 \end{pmatrix}$$

$$L_7(t) = \begin{pmatrix} z_{10} & z_{11} t & 0 \\ z_{12} & 0 & z_{13} t \\ z_{14} & 0 & 0 \end{pmatrix} \quad L_8(t) = \begin{pmatrix} z_{15} & -t & 0 \\ z_{16} & 0 & z_{17} t \\ z_{18} & 0 & 0 \end{pmatrix} \quad L_9(t) = \begin{pmatrix} 0 & z_{19} t & z_{20} t \\ 0 & z_{21} t & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

$$L_{10}(t) = \begin{pmatrix} -z_{41} & 0 & 0 \\ 0 & 0 & 0 \\ -z_{44} & 0 & 0 \end{pmatrix} \quad L_{11}(t) = \begin{pmatrix} z_{22} & 0 & 0 \\ z_{23} & 0 & z_{24} t \\ z_{25} & 0 & 0 \end{pmatrix} \quad L_{12}(t) = \begin{pmatrix} -z_{31} & z_{26} t & 0 \\ 0 & z_{27} t & 0 \\ 0 & 0 & t \end{pmatrix}$$

$$L_{13}(t) = \begin{pmatrix} z_{28} & z_{29} t & 0 \\ z_{30} & 0 & -t \\ 0 & t & 0 \end{pmatrix} \quad L_{14}(t) = \begin{pmatrix} z_{31} & z_{32} t & 0 \\ 0 & 0 & 0 \\ 0 & z_{33} t & -t \end{pmatrix} \quad L_{15}(t) = \begin{pmatrix} 0 & 0 & -t \\ z_{34} & 0 & 0 \\ 0 & 0 & z_{35} t \end{pmatrix}$$

$$L_{16}(t) = \begin{pmatrix} z_{36} & z_{37} t & 0 \\ z_{38} & 0 & z_{39} t \\ 0 & z_{40} t & -t \end{pmatrix} \quad L_{17}(t) = \begin{pmatrix} z_{41} & z_{42} t & 0 \\ 0 & z_{43} t & 0 \\ z_{44} & 0 & 0 \end{pmatrix}$$

The coefficients $z_1, \ldots, z_{44}$ are algebraic numbers described as follows. Let y_* be a real root of the polynomial

$$x^{27} - 2x^{26} + 17x^{25} - 29x^{24} + 81x^{23} + 52x^{22} - 726x^{21} + 3451x^{20} - 10901x^{19} + 25738x^{18} - 50663x^{17} + 72133x^{16} - 72973x^{15} + 10444x^{14} + 138860x^{13} - 308611x^{12} + 427344x^{11} - 267416x^{10} - 196096x^9 + 762736x^8 - 1236736x^7 + 1092352x^6 - 537600x^5 - 42240x^4 + 684032x^3 - 1136640x^2 + 1146880x - 520192.$$

For $i = 1, \ldots, 44$, we consider algebraic numbers y_j in the field extension $\mathbb{Q}[y_*]$, described as a polynomial of degree (at most) 26 in y_* with rational coefficients. Notice that all the y_j's are real. The expressions of the $y_1, \ldots, y_{44}$ in terms of y_* are provided in the file `yy_exps` at `http://www.math.tamu.edu/~jml/CGLVkronsupp.html`, Appendix C. Let z_j be the unique real cubic root of y_j.

We are going to prove that, with this choice of coefficients z_j,

$$t^2 \det_3 + O(t^3) = \sum_{i=1}^{17} L_i(t)^3. \tag{9}$$

The condition $t^2 \det_3 + O(t^3) = \sum_{i=1}^{17} L_i(t)^3$ is equivalent to the fact that the degree 0 and the degree 1 components of $\sum_{i=1}^{17} L_i(t)^3$ vanish and that the degree 2 component equals $\det_3$. Given the sparse structure of the $L_i(t)$, this reduces to a system of 54 cubic equations in the 44 unknowns $z_1, \ldots, z_{44}$. Our goal is to show that the algebraic numbers described above are a solution of this system.

We show that the z_i's satisfy each equation as follows. After evaluating the equations at the z_i's, there are two possible cases

1. all monomials appearing in the equation are elements of $\mathbb{Q}[y_*]$; we say that this is an equation of type 1; there are 14 such equations;
2. at least one monomial appearing in the equation is not an element of $\mathbb{Q}[y_*]$; we say that this is an equation of type 2; there are 40 such equations.

For equations of type 1, we provide expressions of each monomial in terms of y_*. To verify that each expression is indeed equal to the corresponding monomial, it suffices to compare the cube of the given expression and the expression obtained by evaluating the monomial at the y_j's. Finally, the equation can be verified in $\mathbb{Q}[y_*]$. This is performed by the file `checkingType1eqns.m2`.

For equations of type 2, let u be one of the monomials which do not belong to $\mathbb{Q}[y_*]$. We claim that it is possible to choose the monomial in such a way that $\mathbb{Q}[u^3] = \mathbb{Q}[y_*]$. For each equation, we choose one of the monomials and we verify the claim as follows. The element u^3 has an expression in terms of y_* which equals the chosen monomial evaluated at the y_i's. Let M_u be the 27×27 matrix with rational entries such that

$$(1, u^3, \cdots, u^{3\cdot 26}) = \left(1, y_*, \ldots, y_*^{26}\right) \cdot M_u;$$

M_u can be computed directly by considering the expressions of the powers of u^3 in terms of y_*. Then $\mathbb{Q}[u^3] = \mathbb{Q}[y_*]$ if and only if M_u is full rank.

In particular y_* has an expression in terms of u^3, which can be computed inverting the matrix M_u. A consequence of this is that $\mathbb{Q}[u] = \mathbb{Q}[y_*, u]$.

At this point, we observe that $\mathbb{Q}[u]$ contains the other monomials occurring in the equation as well. To see this, we proceed as in the case of equations of type 1. For each monomial occurring in the equation, we provide an expression in terms of u (in fact, to speed up the calculation, we provide an expression in terms of u and y_*, which is equivalent to an expression in u because $\mathbb{Q}[u^3] = \mathbb{Q}[y_*]$ and y_* has a unique expression in terms of u^3); we compare the cube of this expression (appropriately reduced modulo the minimal polynomial of y_* and the relation between u^3 and y_*) with the expression obtained by evaluating the monomial at the y_i's (expressed in terms of y_*). This shows that all monomials occurring in the expression belong to $\mathbb{Q}[u]$, and verifies that the given expressions are indeed equal to the corresponding monomials. Finally, the equation is verified in $\mathbb{Q}[u]$ as in the case of type 1. This is performed by the file `checkingType2eqns.m2`. ◀

5.2.1 Discussion of how the decomposition was obtained

Many steps were accomplished by finding solutions of polynomial equations by nonlinear optimization. In each case, this was accomplished using a variant of Newton's method applied to the mapping of variable values to corresponding polynomial values. The result of this procedure in each case is limited precision machine floating point numbers.

First, we attempted to solve the equations describing a Waring rank 17 decomposition of $\det_3$ with nonlinear optimization, namely, $\det_3 = \sum_{i=1}^{17} (w_i')^{\otimes 3}$, where $w_i' \in \mathbb{C}^{3\times 3}$. Instead of finding a solution to working precision, we obtained a sequence of local refinements to an approximate solution where the norm of the defect is slowly converging to to zero, and some of the parameter values are exploding to infinity. Numerically, these are Waring decompositions of polynomials very close to $\det_3$.

Next, this approximate solution needed to be upgraded to a solution to equation (9).

We found a choice of parameters in the neighborhood of a solution, and then applied local optimization to solve to working precision. We used the following method: Consider the linear mapping $M : \mathbb{C}^{17} \to S^3(\mathbb{C}^{3\times 3})$, $M(e_i) = (w_i')^{\otimes 3}$, and let $M = U\Sigma V^*$ be its

singular value decomposition (with respect to the standard inner products for the natural coordinate systems). We observed that the singular values seemed to be naturally partitioned by order of magnitude. We estimated this magnitude factor as $t_0 \approx 10^{-3}$, and wrote Σ' as Σ where we multiplied each singular value by $(t/t_0)^k$, with k chosen to agree with this observed partitioning, so that the constants remaining were reasonably sized. Finally, we let $M' = U\Sigma' V^*$, which has entries in $\mathbb{C}[[t]]$. M' is thus a representation of the map M with a parameter t.

Next, for each i, we optimized to find a best fit to the equation $(a_i + tb_i + t^2c_i)^{\otimes 3} = M'(e_i)$, which is defined by polynomial equations in the entries of a_i, b_i and c_i. The a_i, b_i and c_i we constructed in this way proved to be a good initial guess to optimize equation (9), and we immediately saw quadratic convergence to a solution to machine precision. At this point, we greedily sparsified the solution by speculatively zero-ing values and re-optimizing, rolling back one step in case of failure. After sparsification, it turned out the c_i were not needed. The resulting matrices are those given in the proof.

To compute the minimal polynomials and other integer relationships between quantities, we used Lenstra-Lenstra-Lovász integer lattice basis reduction [34]. As an example, let $\zeta \in \mathbb{R}$ be approximately an algebraic number of degree k. Let N be a large number inversely proportional to the error of ζ. Consider the integer lattice with basis $\{e_i + \lfloor N\zeta^i \rfloor e_{k+1}\} \subset \mathbb{Z}^{k+2}$, for $0 \leq i \leq k$. Then elements of this lattice are of the form $v_0e_0 + \cdots + v_ke_k + Ee_{k+1}$, where $E \approx Np(\zeta)$, $p = v_0 + v_1x + \cdots x_kx^k$. Polynomials p for which ζ is an approximate root are distinguished by the property of having relatively small Euclidean norm in this lattice. Computing a small norm vector in an integer lattice is accomplished by LLL reduction of a known basis.

For example, the fact that the number field of degree 27 obtained by adjoining any z_α^3 to $\mathbb{Q}$ contains all the rest was determined via LLL reduction, looking for expressions of z_α^3 as a polynomial in z_β^3 for some fixed β. These expressions of z_α^3 in a common number field can be checked to have the correct minimal polynomial, and thus agree with our initial description of the z_α. LLL reduction was also used to find the expressions of values as polynomials in the primitive root of the various number fields.

After refining the known value of the parameters to $10,000$ bits of precision using Newton's method, LLL reduction was successful in identifying the minimal polynomials. The degrees were simply guessed, and the results checked by evaluating the computed polynomials in the parameters to higher precision.

▶ Remark 27. With the minimal polynomial information, it is possible to check that equation (9) is satisfied to any desired precision by the parameters.

6 Tight Tensors in $\mathbb{C}^3 \otimes \mathbb{C}^3 \otimes \mathbb{C}^3$

Following an analysis started in [17], we consider Kronecker squares of *tight* tensors in $\mathbb{C}^3 \otimes \mathbb{C}^3 \otimes \mathbb{C}^3$. We compute their symmetry groups and numerically provide bounds to their tensor rank and border rank, highlighting the submultiplicativity properties.

We refer to [44, 11, 17] for an exposition of the role of tightness in Strassen's work and in the laser method. We point out that $T_{CW,q}$ and $T_{cw,q}$ are tight. (If one uses the combinatorial definition of tightness, which depends on a choice of basis, they are not tight in their standard presentations.)

6.1 Tight tensors

Recall the map $\Phi : GL(A) \times GL(B) \times GL(C) \to GL(A \otimes B \otimes C)$ from Section 3.1 defining the action of $GL(A) \times GL(B) \times GL(C)$ on $A \otimes B \otimes C$. Its differential $d\Phi$ defines a map at the level of Lie algebras, mapping $\mathfrak{gl}(A) \oplus \mathfrak{gl}(B) \oplus \mathfrak{gl}(C)$ to a subalgebra of $\mathfrak{gl}(A \otimes B \otimes C)$, isomorphic to $(\mathfrak{gl}(A) \oplus \mathfrak{gl}(B) \oplus \mathfrak{gl}(C))/\mathbb{C}^2$. Write $\mathfrak{g}_T \subseteq \mathfrak{gl}(A) \oplus \mathfrak{gl}(B) \oplus \mathfrak{gl}(C)$ for the annihilator of T under this action.

A tensor $T \in A \otimes B \otimes C$ is *tight* if $\mathfrak{g}_T/\mathbb{C}^2$ contains a regular semisimple element. Tightness can be defined combinatorially with respect to a basis, see e.g. [17, Def. 1.3]. In particular, the combinatorial definition makes it clear that tightness depends on the support of a tensor in a given basis; we say that a support $\mathcal{S}$ is tight if every tensor having support $\mathcal{S}$ is tight.

Given concise tensors $T_1 \in A_1 \otimes B_1 \otimes C_1$ and $T_2 \in A_2 \otimes B_2 \otimes C_2$, [17, Theorem 4.1] shows that

$$\mathfrak{g}_{T_1 \boxtimes T_2} \supseteq \mathfrak{g}_{T_1} \otimes \mathrm{Id}_{A_2 \otimes B_2 \otimes C_2} + \mathrm{Id}_{A_1 \otimes B_1 \otimes C_1} \otimes \mathfrak{g}_{T_2}; \tag{10}$$

moreover if $\mathfrak{g}_{T_1} = 0$ and $\mathfrak{g}_{T_2} = 0$ then equality holds $\mathfrak{g}_{T_1 \boxtimes T_2} = 0$.

The strict containment in (10) occurs, for instance, in the case of the matrix multiplication tensor. In [17], we posed the problem of characterizing tensors $T \in A \otimes B \otimes C$ such that $\mathfrak{g}_T \otimes \mathrm{Id}_{A \otimes B \otimes C} + \mathrm{Id}_{A \otimes B \otimes C} \otimes \mathfrak{g}_T$ is strictly contained in $\mathfrak{g}_{T^{\boxtimes 2}} \subset \mathfrak{gl}(A^{\otimes 2}) + \mathfrak{gl}(B^{\otimes 2}) + \mathfrak{gl}(C^{\otimes 2})$.

Proposition 29 provides several additional examples of tensors in $\mathbb{C}^3 \otimes \mathbb{C}^3 \otimes \mathbb{C}^3$ for which this containment is strict.

6.2 Tight supports in $\mathbb{C}^3 \otimes \mathbb{C}^3 \otimes \mathbb{C}^3$

From [17, Proposition 2.14], one obtains an exhaustive list of unextendable tight supports for tensors in $\mathbb{C}^3 \otimes \mathbb{C}^3 \otimes \mathbb{C}^3$, up to the action of $\mathbb{Z}_2 \times \mathfrak{S}_3$, where $\mathfrak{S}_3$ acts permuting the factors and $\mathbb{Z}_2$ acts by reversing the order of the basis elements. In fact, tightness is invariant under the action of the full $\mathfrak{S}_3$ acting by permutation on the basis vectors. This additional simplification, pointed out by J. Hauenstein, provides the following list of 9 unextendable tight supports up to the action of $((\mathfrak{S}_3)^{\times 3}) \rtimes \mathfrak{S}_3$.

$$\mathcal{T}_1 = \{(1,1,3),(1,2,2),(2,1,2),(3,3,1)\};$$
$$\mathcal{T}_2 = \{(1,1,3),(1,3,2),(2,3,1),(3,2,2)\};$$
$$\mathcal{T}_3 = \{(1,1,3),(1,2,2),(1,3,1),(2,1,2),(3,2,1)\};$$
$$\mathcal{T}_4 = \{(1,1,3),(1,2,2),(2,1,2),(2,3,1),(3,2,1)\};$$
$$\mathcal{T}_5 = \{(1,1,3),(1,2,2),(2,3,1),(3,1,2),(3,2,1)\};$$
$$\mathcal{T}_6 = \{(1,1,3),(1,3,2),(2,2,2),(3,1,2),(3,3,1)\};$$
$$\mathcal{T}_7 = \{(1,1,3),(1,2,2),(1,3,1),(2,1,2),(2,2,1),(3,1,1)\};$$
$$\mathcal{T}_8 = \{(1,1,3),(1,3,2),(2,2,2),(2,3,1),(3,1,2),(3,2,1)\};$$
$$\mathcal{T}_9 = \{(1,2,3),(1,3,2),(2,1,3),(2,2,2),(2,3,1),(3,1,2),(3,2,1)\};$$

Supports $\mathcal{S}_2$ and $\mathcal{S}_3$ of [17] are equivalent to support $\mathcal{S}_1 = \mathcal{T}_1$; supports $\mathcal{S}_8$ and $\mathcal{S}_{10}$ are equivalent to support $\mathcal{S}_6 = \mathcal{T}_4$.

The following result characterizes tight tensors in $\mathbb{C}^3 \otimes \mathbb{C}^3 \otimes \mathbb{C}^3$ up to isomorphism.

▶ **Proposition 28.** *Let $T \in \mathbb{C}^3 \otimes \mathbb{C}^3 \otimes \mathbb{C}^3$ be a tight tensor with unextendable tight support in some basis. Then, up to permuting the three factors, T is isomorphic to exactly one of the following.*

$$
\begin{aligned}
T_1 :=& a_1 \otimes b_1 \otimes c_3 + a_1 \otimes b_2 \otimes c_2 + a_2 \otimes b_1 \otimes c_2 + a_3 \otimes b_3 \otimes c_1 \\
T_2 :=& a_1 \otimes b_1 \otimes c_3 + a_1 \otimes b_3 \otimes c_2 + a_2 \otimes b_3 \otimes c_1 + a_3 \otimes b_2 \otimes c_2 \\
T_3 :=& a_1 \otimes b_1 \otimes c_3 + a_1 \otimes b_2 \otimes c_2 + a_1 \otimes b_3 \otimes c_1 + a_2 \otimes b_1 \otimes c_2 + a_3 \otimes b_2 \otimes c_1 \\
T_4 :=& a_1 \otimes b_1 \otimes c_3 + a_1 \otimes b_2 \otimes c_2 + a_2 \otimes b_1 \otimes c_2 + a_2 \otimes b_3 \otimes c_1 + a_3 \otimes b_2 \otimes c_1 \\
T_5 :=& a_1 \otimes b_1 \otimes c_3 + a_1 \otimes b_2 \otimes c_2 + a_2 \otimes b_3 \otimes c_1 + a_3 \otimes b_1 \otimes c_2 + a_3 \otimes b_2 \otimes c_1 \\
T_6 :=& a_1 \otimes b_1 \otimes c_3 + a_1 \otimes b_3 \otimes c_2 + a_2 \otimes b_2 \otimes c_2 + a_3 \otimes b_1 \otimes c_2 + a_3 \otimes b_3 \otimes c_1 \\
T_7 :=& a_1 \otimes b_1 \otimes c_3 + a_1 \otimes b_2 \otimes c_2 + a_1 \otimes b_3 \otimes c_1 + a_2 \otimes b_1 \otimes c_2 + a_2 \otimes b_2 \otimes c_1 + a_3 \otimes b_1 \otimes c_1 \\
T_8 :=& a_1 \otimes b_1 \otimes c_3 + a_1 \otimes b_3 \otimes c_2 + a_2 \otimes b_2 \otimes c_2 + a_2 \otimes b_3 \otimes c_1 + a_3 \otimes b_1 \otimes c_2 + a_3 \otimes b_2 \otimes c_1 \\
T_{9,\mu} :=& a_1 \otimes b_2 \otimes c_3 + a_1 \otimes b_3 \otimes c_2 + a_2 \otimes b_1 \otimes c_3 + a_2 \otimes b_2 \otimes c_2 + a_2 \otimes b_3 \otimes c_1 + a_3 \otimes b_1 \otimes c_2 \\
& + \mu \cdot a_3 \otimes b_2 \otimes c_1 .
\end{aligned}
$$

Proof. The result of [17, Proposition 2.14] and the discussion above shows that T is, up to permutation of the factors, equivalent to a tensor with support $\mathcal{T}_i$ for some $i = 1, \ldots, 9$.

For $i = 1, \ldots, 8$, it is straightforward to verify that all tensors with support $\mathcal{T}_i$ are isomorphic, via the change of bases given by three diagonal matrices.

The case of $\mathcal{T}_9$ is slightly more involved but essentially the same argument shows that a tensor T with support $\mathcal{T}_9$ is isomorphic to $T_{9,\mu}$, for some μ.

Finally, we have to show that any two of the tensors in the statement are not isomorphic. For tensors having distinct supports, this is a consequence of Proposition 29 below: indeed, if T, T' are two of the tensors above, Proposition 29 shows that either $\dim \mathfrak{g}_T \neq \dim \mathfrak{g}_{T'}$ or $\dim \mathfrak{g}_{T^{\boxtimes 2}} \neq \dim \mathfrak{g}_{T'^{\boxtimes 2}}$.

As for the tensors with support $\mathcal{T}_9$, we proceed as follows. Let $T = T_{9,\mu}$ and $T' = T_{9,\mu'}$ with $\mu \neq \mu'$. We show that T is not isomorphic to T'. Suppose by contradiction that there is a triple of 3×3 matrices $g = (g_A, g_B, g_C) \in GL_3 \times GL_3 \times GL_3$ with $g(T) = T'$. One sees that in each case, g_A, g_B, g_C have to be diagonal matrices, and an explicit calculation shows that there is no triple of diagonal matrices such that $g(T) = T'$. ◀

We point out that T_7 is isomorphic to the Coppersmith-Winograd tensor $T_{CW,1}$, as well as to the structure tensor of the algebra $\mathbb{C}[x]/(x^3)$.

The tensors $T_{cw,2}$ and $T_{skewcw,2}$ are degenerations of $T_{9,\mu}$, respectively for $\mu = 1$ and $\mu = -1$. In particular, they do not have an unextendable tight support in some basis.

▶ **Proposition 29.** *For $i = 1, \ldots, 9$, the following table records $\dim \mathfrak{g}_{T_i}$ and $\dim \mathfrak{g}_{T_i^{\boxtimes 2}}$.*

T	$\dim \mathfrak{g}_T$	$\dim \mathfrak{g}_{T^{\boxtimes 2}}$
T_1	5	22
T_2	3	9
T_3	5	13
T_4	4	9
T_5	3	7
T_6	2	5
T_7	6	28
T_8	1	2
$T_{9,-1}$	5	10
$T_{9,\mu}$ *(for $\mu \neq 0, -1$)*	1	2

In summary

$$\dim \mathfrak{g}_{T^{\boxtimes 2}} > 2 \dim \mathfrak{g}_T$$

for tight tensors in $\mathbb{C}^3 \otimes \mathbb{C}^3 \otimes \mathbb{C}^3$ with unextendable tight supports $\mathcal{T}_1, \ldots, \mathcal{T}_7$.

Proof. For $T_1, \ldots, T_8$ and for the $T_{9,-1}$, the proof follows by a direct calculation. The first part of the file `symmetryTightSupports.m2` at `www.math.tamu.edu/~jml/CGLVkronsupp.html`, Appendix E computes the dimension of the symmetry algebras of interest in these cases.

The second part of the file deals with the case $T_{9,\mu}$ when $\mu \neq -1$. By tightness, $\dim \mathfrak{g}_{T_{9,\mu}} \geq 1$.

Consider the linear map $\omega_{T_{9,\mu}} : \mathfrak{gl}(A)+\mathfrak{gl}(B)+\mathfrak{gl}(C) \to A\otimes B\otimes C$ defined by $(X,Y,Z) \mapsto (X,Y,Z).T_{9,\mu}$. Then $\mathfrak{g}_{T_{9,\mu}} = [\ker(\omega_{T_{9,\mu}})]/\mathbb{C}^2$, where $\mathbb{C}^2$ corresponds to $\ker d\Phi$.

The second part of the file `symmetryTightSupports.m2` computes a matrix representation of $\omega_{T_{9,\mu}}$, depending on a parameter μ (`t` in the file). Let F_μ be this 27×27 matrix representation. Then, it suffices to select a 24×24 submatrix whose determinant is a nonzero univariate polynomial in μ. If μ is a value for which $\dim \mathfrak{g}_{T_{9,\mu}} > 1$, then μ has to be a root of this univariate polynomial.

In the example computed in the file, we select a 24×24 submatrix whose determinant is $(\mu+1)^6\mu$, showing that the only possible values of μ for which $\dim \mathfrak{g}_{T_{9,\mu}} > 1$ are $\mu = 0$ or $\mu = -1$. The case $\mu = -1$ was considered separately. The case $\mu = 0$ does not correspond to a unextendable support, so it is not of interest. We point out that however, $\omega_{T_{9,0}} = 24$, namely $\dim \mathfrak{g}_{T_{9,0}} = 1$.

For $T_{9,\mu}^{\boxtimes 2}$, we follow essentially the same argument. By tightness, and (10), we obtain $\dim \mathfrak{g}_{T_{9,\mu}^{\boxtimes 2}} \geq 2$. The third part of `symmetryTightSupports.m2` computes a matrix representation of the map $\omega_{T_{9,\mu}^{\boxtimes 2}}$, depending on a parameter μ: this is a 729×243 matrix of rank at most 239.

In the example computed in the file, we select a 239×239 submatrix whose determinant is the univariate polynomial $\mu^8(\mu+1)^{12}$. As before, we conclude. ◀

We also provide the values of the border rank of the tensors in $\mathbb{C}^3 \otimes \mathbb{C}^3 \otimes \mathbb{C}^3$ having unextendable tight support and numerical evidence for the values of border rank of their Kronecker square. They are recorded in the following table. The values of the border rank for the T_i's are straightforward to verify. The lower bounds for the Kronecker squares are obtained via Koszul flattenings. In the cases labeled by N/A the upper bounds coincide with the multiplicative upper bound; in the other cases, the upper bound is obtained via numerical method, and the last column of the table records the ℓ_2 distance (in the given basis) between the tensor obtained via the numerical approximation and the Kronecker square.

T	$\underline{\mathbf{R}}(T)$	$\underline{\mathbf{R}}(T^{\boxtimes 2})$	ℓ_2 error for upper bound in $T^{\boxtimes 2}$ decomposition
T_1	3	9	N/A
T_2	4	$[11, 14]$	0.000155951
T_3	4	$[11, 14]$	0.00517612
T_4	4	14	0.0144842
T_5	4	$[11, 15]$	0.0237172
T_6	4	$[11, 15]$	0.00951205
T_7	3	9	N/A
T_8	4	$[14, 16]$	N/A
$T_{9,-1}$	5	$[16, 19]$	0.0231353
$T_{9,\mu}$ (for $\mu \neq 0, -1$)	4	$[15, 16]$	N/A

7 Justification of the matrices

In this section, describe two ways of proving that the matrices appearing in the second proof of Theorem 5 and the proof of Theorem 6 are as asserted, one of which is carried out explicitly in the code at `http://www.math.tamu.edu/~jml/CGLVkronsupp.html`.

The computational issue is that, although the sizes of the matrices are fixed, they are obtained via intermediate matrices whose dimensions depend on q so one needs a way of encoding such matrices and tensors efficiently. The first method of proof critically relies on the definition of a class of tensors, which we call *box parameterized*, whose entries and dimensions depend on a parameter q in a very structured way. In this proof one shows the entries of the output matrices are low degree, say δ, polynomials in q, and then by computing the first $\delta+1$ cases directly, one has proven they are as asserted for all q. The second method, which is implemented in the code, does not rely on the structure to prove anything, but the structure allows an efficient coding of the tensors that significantly facilitates the computation.

A k-way sequence of tensors $T_q \in A_1^q \otimes \cdots \otimes A_k^q$ parametrized by $q \in \mathbb{N}$ is *basic box parameterized* if it is of the form

$$T_q = p(q)\sum_{(i_1,\ldots,i_k)\in\Phi} t_{i_1,\ldots,i_k},$$

where $\{a_{\alpha,s}\}$ is a basis of A_α^q, $t_{i_1,\ldots,i_k} = a_{1,i_1}\otimes\cdots\otimes a_{k,i_k}$, p is a polynomial, and the index set Φ is defined by conditions $f_jq+h_j \le i_j \le g_jq+d_j$, $f_j,g_j\in\{0,1\}$, $h_j,d_j\in\mathbb{Z}_{\ge 0}$, for each j, and any number of equalities $i_j = i_k$ between indices.

We sometimes abuse notation and consider Φ to be its set of indices or the set of equations and inequalities defining the set of indices; no confusion should arise.

Tensor products of basic box parameterized tensors are basic box parameterized:

$$\begin{aligned}&(p_1(q)\textstyle\sum_{(i_1,\ldots,i_k)\in\Phi_1} t_{i_1,\ldots,i_k}) \otimes (p_2(q)\sum_{(j_1,\ldots,j_l)\in\Phi_2} j_{i_1,\ldots,j_l})\\ &= p_1(q)p_2(q)\textstyle\sum_{(i_1,\ldots,i_k,j_1,\ldots,j_l)\in\Phi_1\times\Phi_2} t_{i_1,\ldots,i_k,j_1,\ldots,j_l}.\end{aligned}$$

We next show that contraction of a basic box parameterized tensor is basic box parameterized when $q \ge \max_{i,j}\{|h_i-h_j|,|d_i-d_j|\}$, where i and j range over those indices related by equality to the ones being contracted. To do this, we first show they are closed under summing along a coordinate (with the same restriction on q), which we may take to be i_1 without loss of generality. (This corresponds to contracting with the vector $\sum_{i_1} a^*_{1,i_i} \in (A_1^q)^*$.) That is, we wish to show

$$p(q)\textstyle\sum_{(i_1,\ldots,i_k)\in\Phi} t_{i_2,\ldots,i_k}$$

is basic box parameterized with the above restriction on q. For this consider two cases. First, suppose there is a coordinate $j\neq 1$ so that $i_1 = i_j \in \Phi$. To construct the summed tensor, adjoin to Φ equalities $i_j = i_k$ for all k for which $i_1 = i_k \in \Phi$. Then, deleting i_1 from the indices and replacing the bounds on i_j with

$$\max(f_jq+h_j, f_1q+h_1) \le i_j \le \min(g_jq+d_j, g_1q+d_1)$$

yields the summed tensor. The max and the min can be replaced with one of their arguments provided $q \ge \max(|h_1-h_j|,|d_1-d_j|)$, so the sum is basic box parameterized with our restriction on q. Otherwise, suppose there is no coordinate so that $i_1 = i_j \in \Phi$. Then the summed tensor is $(g_1q+d_1-f_1q-h_1+1)p(q)\sum_{(i_2,\ldots,i_k)\in\Phi} t_{i_2,\ldots,i_k}$, which is basic box parameterized.

Finally, to compute the contraction, say between indices i_j and i_k, adjoin $i_j = i_k$ as a condition to Φ and then sum over i_j and then over i_k using the previous technique.

Call a tensor *box parameterized* if it is a finite sum of basic box parameterized tensors. Clearly box parameterized tensors are closed under tensor products and contraction, possibly with an easily computed restriction on q.

Now, $T_{cw,q} \in (\mathbb{C}^{q+1})^{\otimes 3} = A \otimes B \otimes C$ is clearly box parameterized as a 3-way tensor. The tensors $\phi_2 \in A' \otimes (A^{\otimes 2})^*$ (where $\dim A' = 3$) and $\phi_3 \in A' \otimes (A^{\otimes 3})^*$ (where $\dim A' = 5$) defining the projection maps are box parameterized as 3-way and 4-way tensors, respectively. The tensors $KF_1 \in (A' \otimes B^{\otimes 2} \otimes C^{\otimes 2})^* \otimes ((A')^* \otimes B^{\otimes 2}) \otimes (\Lambda^2 A' \otimes C^{\otimes 2}))$ and $KF_2 \in (A' \otimes B^{\otimes 3} \otimes C^{\otimes 3})^* \otimes ((\Lambda^2 A')^* \otimes B^{\otimes 3}) \otimes (\Lambda^3 A' \otimes C^{\otimes 3}))$ defining the Koszul flattenings are also box parameterized, as they are the tensor product of tensors of fixed size with identity tensors, which are basic box parameterized. From this, we see that the corresponding Koszul flattenings are box parameterized, viewed in $A'^* \otimes B^{\otimes 2} \otimes \Lambda^2 A' \otimes C^{\otimes 2}$ as a 6-way tensor for the square and $\Lambda^2 A'^* \otimes B^{\otimes 3} \otimes \Lambda^3 A' \otimes C^{\otimes 3}$ as an 8-way tensor for the cube.

Finally, consider the change of basis map which block diagonalizes the flattening according to Schur's lemma. We explain the square case, the cube case is available in the Appendix. This change of basis is the Kronecker product of the 3×3 identity with the Kronecker square of the map represented by the following $q+1 \times q+1$ matrix

$$\begin{pmatrix}
1 & & & & & & & & \\
& 1 & & & & & & & \\
& & 1 & & & & & & \\
& & & 1 & & & & & \\
& & & & 1 & 1 & 1 & \cdots & 1 \\
& & & & -1 & 1 & & & \\
& & & & & -1 & 1 & & \\
& & & & & & & \ddots & 1 \\
& & & & & & & & -1 \quad 1
\end{pmatrix}$$

Let E_0 denote the projection operator to the isotypic component of the trivial representation. In bases, this corresponds to the first five rows of the matrix above. Let E_1 denote the projection onto the standard representation, which corresponds to the sixth row. It is easy to see that the first 6 columns of the inverse is the matrix

$$\frac{1}{q-3}\begin{pmatrix}
q-3 & & & & & & \\
& q-3 & & & & & \\
& & q-3 & & & & \\
& & & q-3 & & & \\
& & & & 1 & -(q-4) \\
& & & & 1 & 1 \\
& & & & \vdots & \\
& & & & 1 & 1
\end{pmatrix}$$

Write F_0 for the inclusion of the trivial representation into the space in the original basis, which is represented by the first five columns of this matrix, and F_1 for the inclusion of the standard representation which is represented by the sixth column. Write V_0 for the trivial representation of $\mathfrak{S}_{q-3}$ and V_1 for the standard representation. Then,

$$f_{V_i \boxtimes V_j} = (\mathrm{Id}_{A'} \boxtimes E_i \boxtimes E_j) \circ (T_{cw,q}^{\otimes 2})_{A'}^{\wedge 2} \circ (\mathrm{Id}_{\Lambda^2 A'} \boxtimes F_i \boxtimes F_j).$$

(These four maps were labeled $f_1, \ldots, f_4$ in §4.5.) Since E_i and $(q-3)F_i$ are clearly box parametrized, it follows that $(q-3)^2 f_{V_i \boxtimes V_j}$ is box parametrized. A similar argument shows that the cube $(q-3)^3 f_{V_i \boxtimes V_j \boxtimes V_k}$ is box parameterized.

At this point the first method shows the entries of the matrices are low degree polynomials in q so one can conclude by checking the first few cases.

The fact that all tensors involved are basic box parameterized guided us how to encode these maps efficiently so that they could be computed by direct calculation, which provides the second method and is described in Appendix D.

References

1 J. Alman. Limits on the Universal Method for Matrix Multiplication. *CoRR*, abs/1812.08731, 2018. arXiv:1812.08731.

2 J. Alman and V. Vassilevska Williams. Limits on All Known (and Some Unknown) Approaches to Matrix Multiplication. *ArXiv e-prints*, October 2018. arXiv:1810.08671.

3 J. Alman and V. V. Williams. Further Limitations of the Known Approaches for Matrix Multiplication. In *9th Innov. Th. Comp. Science Conf., ITCS 2018, January 11-14, 2018, Cambridge, MA, USA*, pages 25:1–25:15, 2018.

4 A. Ambainis, Y. Filmus, and F. Le Gall. Fast matrix multiplication: limitations of the Coppersmith-Winograd method. In *Proc. of the 47th ACM Symp. Th. Comp.*, pages 585–593. ACM, 2015.

5 E. Ballico, A. Bernardi, M. Christandl, and F. Gesmundo. On the partially symmetric rank of tensor products of W-states and other symmetric tensors. *Atti Accad. Naz. Lincei Rend. Lincei Mat. Appl.*, 30:93–124, 2019.

6 A. R. Benson and G. Ballard. A Framework for Practical Parallel Fast Matrix Multiplication. In *Proceedings of the 20th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, PPoPP 2015, pages 42–53, New York, NY, USA, February 2015. ACM. doi:10.1145/2688500.2688513.

7 D. Bini. Relations between exact and approximate bilinear algorithms. Applications. *Calcolo*, 17(1):87–97, 1980.

8 D. Bini, G. Lotti, and F. Romani. Approximate solutions for the bilinear form computational problem. *SIAM J. Comput.*, 9(4):692–697, 1980.

9 M. Bläser. Fast Matrix Multiplication. *Theory of Computing, Graduate Surveys*, 5:1–60, 2013.

10 M. Bläser and V. Lysikov. On degeneration of tensors and algebras. In *41st International Symposium on Mathematical Foundations of Computer Science*, volume 58 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages Art. No. 19, 11. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2016.

11 P. Bürgisser, M. Clausen, and M. A. Shokrollahi. *Algebraic complexity theory*, volume 315 of *Grundlehren der Mathematischen Wissenschaften*. Springer-Verlag, Berlin, 1997.

12 M. Christandl, F. Gesmundo, and A. K. Jensen. Border rank is not multiplicative under the tensor product. *SIAM J. Appl. Alg. Geom.*, 3:231–255, 2019.

13 M. Christandl, P. Vrana, and J. Zuiddam. Barriers for fast matrix multiplication from irreversibility. *CoRR*, abs/1812.06952, 2018. arXiv:1812.06952.

14 H. Cohn, R. Kleinberg, B. Szegedy, and C. Umans. Group-theoretic Algorithms for Matrix Multiplication. In *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '05, pages 379–388, Washington, DC, USA, 2005. IEEE Computer Society. doi:10.1109/SFCS.2005.39.

15 H. Cohn and C. Umans. A group theoretic approach to fast matrix multiplication. *Proceedings of the 44th annual Symposium on Foundations of Computer Science*, 2:438–449, 2003.

16 A. Conner, F. Gesmundo, J. M. Landsberg, , and E. Ventura. Tensors with maximal symmetries. *arXiv*, 2019. arXiv:1909.09518.

17 A. Conner, F. Gesmundo, J. M. Landsberg, E. Ventura, and Y. Wang. A geometric study of Strassen's asymptotic rank conjecture and its variants. *arXiv*, 2018. arXiv:1811.05511.

18 D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *J. Symb. Comput.*, 9(3):251–280, 1990.

19 H. Derksen. On the nuclear norm and the singular value decomposition of tensors. *Found. Comp. Math.*, 16(3):779–811, 2016.

20 H. Derksen and V. Makam. On non-commutative rank and tensor rank. *Linear Multilinear Algebra*, 66(6):1069–1084, 2018. doi:10.1080/03081087.2017.1337058.

21 K. Efremenko, A. Garg, R. Oliveira, and A. Wigderson. Barriers for rank methods in arithmetic complexity. In *9th Innovations in Theoretical Computer Science*, volume 94 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages Art. No. 1, 19. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2018.

22 W. Fulton. *Young tableaux. With applications to representation theory and geometry*, volume 35 of *London Mathematical Society Student Texts*. Cambridge University Press, Cambridge, 1997.

23 W. Fulton and J. Harris. *Representation theory: a first course*, volume 129 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 1991.

24 D. A. Gay. Characters of the Weyl group of $SU(n)$ on zero weight spaces and centralizers of permutation representations. *Rocky Mountain J. Math.*, 6(3):449–455, 1976.

25 F. Gesmundo, C. Ikenmeyer, and G. Panova. Geometric complexity theory and matrix powering. *Diff. Geom. Appl.*, 55:106–127, 2017.

26 N. Ilten and Z. Teitler. Product ranks of the 3×3 determinant and permanent. *Canad. Math, Bull.*, 59(2):311–319, 2016.

27 J. M. Landsberg. *Tensors: Geometry and Applications*, volume 128 of *Graduate Studies in Mathematics*. American Mathematical Society, Providence, RI, 2012.

28 J. M. Landsberg. *Geometry and complexity theory*, volume 169 of *Cambridge Studies in Advanced Mathematics*. Cambridge University Press, Cambridge, 2017.

29 J. M. Landsberg. *Tensors: Asymptotic Geometry and Developments 2016–2018*, volume 132 of *CBMS Regional Conference Series in Mathematics*. AMS, 2019.

30 J. M. Landsberg and M. Michałek. A $2n^2 - \log(n) - 1$ lower bound for the border rank of matrix multiplication. *Int. Math. Res. Not.*, 15:4722–4733, 2018.

31 J. M. Landsberg and G. Ottaviani. Equations for secant varieties of Veronese and other varieties. *Ann. Mat. Pura Appl. (4)*, 192(4):569–606, 2013.

32 J. M. Landsberg and G. Ottaviani. New lower bounds for the border rank of matrix multiplication. *Th. of Comp.*, 11(11):285–298, 2015.

33 F. Le Gall. Powers of tensors and Fast Matrix Multiplication. In *Proc. 39th Int. Symp. Symb. Alg. Comp.*, pages 296–303. ACM, 2014.

34 A. K. Lenstra, Jr. H. W. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. *Math. Ann.*, 261(4):515–534, 1982. doi:10.1007/BF01457454.

35 T. Lickteig. Typical tensorial rank. *Lin. Alg. Appl.*, 69:95–120, 1985.

36 A. Schönhage. Partial and total matrix multiplication. *SIAM J. Comp.*, 10(3):434–455, 1981.

37 A. V. Smirnov. The Approximate Bilinear Algorithm of Length 46 for Multiplication of 4 x 4 Matrices. *arXiv*, 2014. arXiv:1412.1687.

38 A. Stothers. *On the Complexity of Matrix Multiplication.* PhD thesis, U. Edinburgh, 2010.

39 V. Strassen. Gaussian elimination is not optimal. *Numerische mathematik*, 13(4):354–356, 1969.

40 V. Strassen. Rank and optimal computation of generic tensors. *Lin. Alg. Appl.*, 52/53:645–685, 1983.

41 V. Strassen. Relative bilinear complexity and matrix multiplication. *J. Reine Angew. Math.*, 375/376:406–443, 1987.

42 V. Strassen. The asymptotic spectrum of tensors. *J. Reine Angew. Math.*, 384:102–152, 1988.

43 V. Strassen. Degeneration and complexity of bilinear maps: some asymptotic spectra. *J. Reine Angew. Math.*, 413:127–180, 1991. doi:10.1515/crll.1991.413.127.

44 V. Strassen. Algebra and complexity. In *First European Congress of Mathematics Paris, July 6–10, 1992*, pages 429–446. Springer, 1994.

45 V. V. Williams. Multiplying matrices faster than Coppersmith-Winograd. In *Proc. 44th ACM Symp. Th. Comp. – STOC'12*, pages 887–898. ACM, 2012.

Algorithms and Lower Bounds for Cycles and Walks: Small Space and Sparse Graphs

Andrea Lincoln
MIT, Cambridge, MA, USA
andreali@mit.edu

Nikhil Vyas
MIT, Cambridge, MA, USA
nikhilv@mit.edu

Abstract

We consider space-efficient algorithms and conditional time lower bounds for finding cycles and walks in graphs. We give a reduction that connects the running time of undirected $2k$-cycle to finding directed odd cycles, s-t connectivity in directed graphs, and Max-3-SAT. For example, we show that if $2k$-cycle on $O(n)$-edge graphs can be solved in $O(n^{1.5-\epsilon})$ time for some $\epsilon > 0$ then, a $2^{n(1-\epsilon')}$ time algorithm exists for Max-3-SAT for some $\epsilon' > 0$. Additionally, we give a tight combinatorial lower bound for $2k$-cycle detection, specifically when k is odd, of $m^{2k/(k+1)+o(1)}$ given the Combinatorial k-Clique Hypothesis.

On the algorithms side, we present a randomized algorithm for directed s-t connectivity using $O(\lg(n)^2)$ space and $O(n^{\lg(n)/2+o(\lg(n))})$ expected time, giving a time improvement over Savitch's famous algorithm, which takes at least $n^{\lg(n)-o(\lg(n))}$ time. Under the conjecture that every $O(\lg(n)^2)$-space algorithm for directed s-t connectivity requires $n^{\Omega(\lg(n))}$ time, we show that undirected $2k$-cycle in $O(\lg(n))$ space requires $n^{\Omega(\lg(k))}$ time.

2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness; Theory of computation → Graph algorithms analysis

Keywords and phrases k-cycle, Space, Savitch, Sparse Graphs, Max-3-SAT

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.11

Funding *Andrea Lincoln*: Supported by NSF Grants CCF-1417238, CCF-1528078 and CCF-1514339, and BSF Grant BSF:2012338.
Nikhil Vyas: Supported by an Akamai Presidential Fellowship, NSF Grant CCF-1552651 and CCF-1909429.

1 Introduction

The k-cycle problem, determining whether a graph on n nodes and m edges contains a cycle of length k, is a fundamental subgraph detection problem. Past work has explored efficient algorithms for detecting/counting, odd/even, directed/undirected cycles in both sparse and dense graphs (e.g. [17, 5, 2, 8, 18]). In this paper, we explore k-cycle algorithms using small space, e.g., $\text{poly}(\log n)$.

When $k \geq \Omega(n)$, for example, $k = n/2$, detecting a k-cycle is well-known to be NP-hard [9]. When k is constant and algorithms are given $O(n^2)$ space, odd cycles can be detected in matrix multiplication time [3, 5]. When k is constant and algorithms are given $O(n)$ space, even cycles can be detected in $O(n^2)$ time [17]. Given an undirected sparse graph, detecting a $2k$-cycle can be done faster, specifically $O(m^{2k/(k+1)})$ time [5]. However, this work leaves open the question of what happens in the very small space regime.

When algorithms are allowed $O(\lg(n) \cdot \lg(k))$ space, the Savitch algorithm can be used to solve k-cycle detection in time $n^{\lg(k)+o(\lg(k))}$ time. In particular, the Savitch algorithm can be used to detect if two nodes s and t are connected by a path of length at most k in a

directed graph, using $n^{O(\lg(k))}$ time and $O(\lg(n) \cdot \lg(k))$ space. We explore the relationship between small space k-cycle detection and algorithms for s-t connectivity, and show that the complexities of the two problems are tightly connected. To the best of our knowledge, no s-t connectivity algorithms have been reported running in $O(\lg^2 n)$ space and $n^{0.999 \cdot \lg(n)}$ time.

Our Results

The primary contribution of this paper is a reduction from directed k'-cycle to undirected $2k$-cycle for constant k' and k such that $k = k' \operatorname{poly}\lg(k)$. We use this reduction to show hardness for even cycle detection in both small space and in sparse graphs.

We prove that, for small-space algorithms, the time complexities of k-cycle detection and k-path detection are equivalent up to constant factors in the exponent. We show that an $n^{o(\lg(k))}$ time and $O(\lg(n))$ space algorithm for the $2k$-cycle detection problem would imply an algorithm for s-t running in $n^{o(\lg(n))}$ time and $O(\lg^2(n))$ space, a significant improvement over Savitch's algorithm.

We propose two natural lower bound hypotheses, and derive interesting consequences from them. The first hypothesis is that s-t connectivity in $O(\lg^2(n))$ space requires $n^{\Omega(\lg(n))}$ time. The second (stronger) hypothesis is that s-t connectivity in $\operatorname{poly}\lg(n)$ space requires $n^{\Omega(\lg(n))}$ time. We call these hypotheses the Weak Savitch Hypothesis (WSH) and the Strong Savitch Hypothesis (SSH) respectively.

▶ **Definition 1.** *The Weak Savitch Hypothesis (WSH) states that given a directed graph G with n nodes and two specified nodes s and t determining if a path exists from s to t requires $n^{\Omega(\lg(n))}$ time if the algorithm can only use $O(\lg^2(n))$ space.*

▶ **Definition 2.** *The Strong Savitch Hypothesis (SSH) states that given a directed graph G with n nodes and two specified nodes s and t determining if a path exists from s to t requires $n^{\Omega(\lg(n))}$ time if the algorithm can only use $O(\operatorname{poly}\lg(n))$ space.*

Both of our Hypotheses are about the s-t connectivity problem, two well known algorithms for s-t connectivity are:

- Savitch's algorithm [12] running in time $\tilde{O}(n^{\lg n})$ and space $O(\lg^2 n)$.
- Depth-First search running in polynomial time and space $\tilde{O}(n)$.

The only improvement over these for any space greater than $\lg^2 n$ is the algorithm of Barnes, Buss, Ruzzo and Schieber [4] who gave a space $O(n/2^{\sqrt{\lg(n)}})$, polynomial time algorithm. They are also able to give a time space tradeoff of space s where $s > \lg^2 n$ and time $2^{O(\lg^2(n/s))}\operatorname{poly}(n)$. This time space tradeoff would require $n^{1-o(1)}$ space to achieve $n^{o(\lg n)}$ time. Our hypotheses are much weaker and only claim the non-existence of $n^{o(\lg n)}$ time algorithms for $O(\lg^2(n))$ space and $O(\operatorname{poly}\lg(n))$ space respectively.

We prove the following hardness for $2k$-cycle under Weak Savitch Hypothesis.

▶ **Theorem 3.** *Assuming Weak Savitch Hypothesis (WSH) solving k-cycle in undirected graphs in $O(\lg n)$ space requires $n^{\Omega(\lg k)}$ time.*

Could we strengthen the hypotheses (Weak Savitch Hypothesis (WSH) and the Strong Savitch Hypothesis (SSH)) to state that there are no $n^{(1-\epsilon)\lg(n)}$ time algorithms for any constant $\epsilon > 0$, instead of merely saying there is no $n^{o(\lg(n))}$ time algorithms? We prove that such a hypothesis would actually be false for randomized algorithms.

▶ **Theorem 4.** *There is a randomized algorithm for directed st-connectivity which runs in expected time $n^{\lg n/2+o(\lg n)}$ and $O(\lg^2(n))$ space.*

Our randomized algorithm works as follows. First, for sufficiently short path lengths (up to $\lg(n)$) we run Savitch's algorithm as usual. For longer path lengths, instead of trying to find the middle vertex of the path we look for any vertex which is approximately in the middle. We show that this minor modification can actually speed up Savitch's algorithm by a quadratic factor. Similar ideas were used by Zwick [19] to give algorithms for All pairs shortest paths problem in arbitrary space regime.

Other Implications

Our k'-cycle to $2k$-cycle reduction ($k = k' \text{poly} \lg(k)$) has many more implications. We give lower bounds for sparse even-cycle detection by reducing Max-3-SAT to sparse even-cycle detection and give a tight combinatorial lower bound for sparse even-cycle. We achieve this by combining our reduction and previous hardness results for directed k-cycle [10].

▶ **Theorem 5.** *If undirected $2k$-cycle, for any large enough constant k can be solved in a graph G with $m = O(n)$ in time $O(m^{1.5-\epsilon})$ then max-3-SAT can be solved in time $O^*(2^{(1-\epsilon')n})$.*

We additionally present a very simple reduction from detecting odd k-cycles to detecting $2k$-cycles. This reduction is inspired by a reduction from Dahlgaard et al. [5]. Our results here are tight, but only in the "combinatorial" regime. The notion of *combinatorial algorithms* is not formally defined, however, it is a commonly used notion (e.g. [10, 1]) which informally just excludes fast matrix multiplication as a subroutine (e.g. [6, 15]). The category of combinatorial algorithms considered generally assumes boolean matrix multiplication requires $n^{3-o(1)}$ time, and attempts to capture algorithms that are fast in practice [16]. We can obtain a combinatorial lower bound for the undirected $2k$-cycle detection of $\Theta(m^{2-o(1)})$ time, for all large enough constants k, if the Combinatorial k-Clique Hypothesis is true.

Additionally for *odd* k we show a tight bound for combinatorial $2k$-cycle detection. If you can detect $2k$-cycles combinatorially in time $m^{(1-\epsilon)2k/(k+1)}$ for some $\epsilon > 0$ then you violate the Combinatorial k-Clique Hypothesis. Dahlgaard et al. present an algorithm for all $2k$-cycles that runs in time $m^{2k/(k+1)}$. So for odd k, we have a tight combinatorial lower bound for $2k$-cycles.

Previous Work

Over the years, there have been many algorithmic results for the cycle detection problem. In arbitrary space, $2k$-cycle can be solved in $O(n^2)$ space as long as k is a constant [17]. This beats the matrix multiplication (n^ω) time algorithms for odd k-cycle detection assuming the matrix multiplication constant is larger than two ($\omega > 2$) [2]. Dahlgaard et al. give an algorithm for sparse even $2k$-cycle detection which runs in $O(m^{2k/(k+1)})$ time, where m is the number of edges in the input graph [5]. These algorithms require $\Omega(m)$ space to run. In very small space finding a k-cycle takes much longer, and as we show in this paper has a strong relationship to s-t connectivity in small space. Savitch's algorithm solves s-t connectivity in space $\lg^2(n)$ in time $n^{\lg(n)-o(\lg(n))}$ [12]. In this paper we present a randomized algorithm which runs in space $\lg^2(n)$ in time $n^{\lg(n)/2+o(\lg(n))}$, improving over the Savitch algorithm. The problem of solving s-t connectivity in less than $\lg^2(n)$ space has also been explored. Gopalan et al. present a randomized time space tradeoff for the s-t connectivity problem where given $\lg^2(n)/\lg(d)$ space they acheive a running time of $O(n^{d \lg(n)/\lg(d)})$ for $d \in [2, n]$ [7].

While we provide a reduction from directed k-cycle for arbitrarily large k to even cycle, a previous paper reduced 3-cycle specifically to even cycle [5]. Their reduction technique inspired ours. They give a lower bound of $m^{3/2-o(1)}$ for even cycle relying on a combinatorial

conjecture for the running time of 3-cycle detection. Notably, they assume no combinatorial $n^{3-o(1)}$ algorithm exists for 3-cycle detection. We give a matching lower bound of $m^{3/2-o(1)}$ using a *non-combinatorial* hypothesis.

Furthermore, our reduction gives an improved lower bound of $m^{2-o(1)}$ using a combinatorial assumption. Giving a tight combinatorial lower bound for the even cycle problem. We give a combinatorial lower bound of $m^{\frac{2k}{k+1}-o(1)}$ for the undirected $2k$-cycle detection problem for odd k. This is closely related to the running time of $O(m^{2k/(k+1)})$ for the algorithm detecting *even* k cycles combinatorially by Dahlgaard et al. [5]. We can additionally use a simple reduction inspired by Dahlgaard et al. [5] and a reduction from k-clique to sparse k-cycle from previous work [10] to give a tight combinatorial lower bound for the $4k+2$-cycle problem. to give a lower bound of $m^{\frac{2k}{k+1}-o(1)}$ for detecting $2k = 4k'+2$ cycles. This shows optimality of the Dahlgaard et al algorithm for half of all even k-cycle detection problems.

By giving a reduction from directed k'-cycle to $2k$-cycle we can show $n^{\Omega(\lg(k))}$ hardness for $2k$-cycle in $O(\lg(n))$ space. However, one can not show this hardness using the 3-cycle reduction of Dahlgaard et al., because 3-cycle has an $O(n^3)$-time logspace algorithm.

In the weaker model of AC_0 formulas a $n^{\Omega(\lg(k))}$ lower bound is known unconditionally for the $\leq k$ connectivity problem [11].

Improving low-space algorithms for directed st-connectivity is a longstanding open problem [14]. Barnes, Buss, Ruzzo and Schieber [4] gave a sublinear space $O(n/2^{\sqrt{\lg(n)}})$, polynomial time algorithm. Even their results only give $(\lg n)^{O(\lg n)}$ factor improvements over Savitch if we restrict to polylog space. Hence for polylog space they require $n^{\lg n(1-o(1))}$ time. Melkebeek and Prakriya [13] gave a unambiguous algorithm running in polynomial time and $O(\lg^{3/2} n)$ space. The time and space here are both better than the ones in our hypotheses but unambiguous Turing machines are possibly much more powerful then deterministic ones.

Organization

In Section 2 we cover definitions and preliminaries. In Section 3 we present a faster s-t connectivity algorithm. In Section 4 we give our reduction between directed k-cycle and even k-cycle. In Section 5 we cover the implications of our reduction.

2 Preliminaries

We will always assume k in the k-cycle/walk problems to be a constant unless stated otherwise.

Notation

We will use the notation $\tilde{O}(f(n))$ to suppress sub-polynomial factors $\left(f(n)^{o(1)}\right)$.

Definitions

▶ **Definition 6.** *A k-cycle in a graph G is a set of k distinct vertices, $x_1, \ldots, x_k$ in G such that the edges $(x_1, x_2), \ldots, (x_{k-1}, x_k), (x_k, x_1)$ all exist in G.*

A k-walk in a graph G is a set of k not necessarily distinct vertices, $x_1, \ldots, x_k$ in G such that the edges $(x_1, x_2), \ldots, (x_{k-1}, x_k)$ all exist in G.

Throughout the paper we will take k to be a large enough constant. We will have runtimes of the form $n^{\Theta(\lg(k))}$ where Θ hides a constant independent of k; such notation only makes sense for k that can grow unboundedly.

We will assume throughout that all graphs are presented in adjacency list format. This is done for simplicity and not assuming this has no effect on our results.

▶ **Reminder of Definition 1.** *The Weak Savitch Hypothesis (WSH) states that given a directed graph G with n nodes and two specified nodes s and t determining if a path exists from s to t requires $n^{\Omega(\lg(n))}$ time if the algorithm can only use $O(\lg^2(n))$ space.*

▶ **Reminder of Definition 2.** *The Strong Savitch Hypothesis (SSH) states that given a directed graph G with n nodes and two specified nodes s and t determining if a path exists from s to t requires $n^{\Omega(\lg(n))}$ time if the algorithm can only use $O(\text{poly}\lg(n))$ space.*

Currently the best known algorithms for directed s t connectivity which run in time $n^{o(\lg(n))}$ require $n^{1-o(1)}$ space[4].

3 Making Savitch run faster

3.1 Savitch's Algorithm

We begin by recalling Savitch's classical algorithm.

Algorithm 1 Savitch's Algorithm.

Input: Graph $G = (V, E)$, vertices s, t, $k \geq 0$
Output: If there exists a path of length at most k from s to t.

```
function SA(G, s, t, k)
    if k = 1 then
        Output TRUE if (s, t) ∈ E, otherwise FALSE
    Let |V| = n.
    for v ∈ [n] do
        if SA(G, s, v, ⌊k/2⌋) ∧ SA(G, v, t, ⌈k/2⌉) = TRUE then
            Output TRUE
    Output ret
```

▶ **Lemma 7** ([12]). *Savitch's algorithm solves directed $\leq k$-reachability in time $T_S(n,k) = \tilde{O}(n^{\lg(k)})$ using space $O(\lg(k)\lg(n))$.*

Savitch's Algorithm [12] solves the directed $s - t$ connectivity problem as follows. It considers the more general problem of $\leq k$ connectivity. $k = n$ is equivalent to $s - t$ connectivity. For determining if there is a path of length at most k between s and t, we guess the middle vertex v in such a path, then recursively try to solve $\leq k/2$ connectivity for (s, v) and (v, t). This gives us the recurrence:

$$T(n,k) \leq n(2 \cdot T(n,k/2)) + \text{poly}(n)$$

with the base case $T(n,1) = O(\text{poly}(n))$. This base case is unaffected by being in adjacency list format or adjacency matrix format. This solves to $T(n,n) = \tilde{O}(n^{\lg n})$.

We now turn improving the running time of s-t connectivity in $O(\lg^2(n))$ space.

Similarly to the Savich's algorithm we will be guessing a node in the middle of the hypothetical path. However, Savich's algorithm requires the node that is exactly in the middle of the path. In our algorithm we will accept a node that is approximately in the middle. There is a trade-off with approximating. By approximating we improve the probability of

an accept. Consider if we accepted any node that was with $\epsilon_k \cdot k$ nodes of the true middle of a k length path, then we have a $2\epsilon_k/n$ probability of correctly guessing such a node. However, this approximation factor trades off how much progress we make with a correct guess. In Savich's Algorithm each correct guess halves the path length. In contrast, with an approximation factor of ϵ_k we divide the path into the less favorable $1/2 - \epsilon_k$ and $1/2 + \epsilon_k$ split. This trade-off is optimal when ϵ_k is approximately $1/\lg(n)$, so we accept any node that is in the middle $1/\lg(n)$ fraction of a k length path.

Our square-root speedup intuitively comes from improving the probability of correctly guessing from $1/n$ to $k/(\lg(n) \cdot n)$. This approaches n as k approaches $\lg(n)$, but achieves a huge savings when k is much larger than $\lg(n)$.

▶ **Reminder of Theorem 4.** *There is a randomized algorithm for directed st-connectivity which runs in expected time* $n^{\lg n/2+o(\lg n)}$ *and* $O(\lg^2(n))$ *space.*

Proof. For sufficiently short path lengths (up to $\lg(n)$) we run Savitch's algorithm as usual. For longer path lengths, instead of trying to find the middle vertex of the path we look for any vertex which is approximately in the middle. We describe the algorithm in full detail below.

Algorithm 2 Improved Savitch's Algorithm.

Input: Graph $G = (V, E)$, vertices s, t
Output: If there exists a path from s to t in G.

function ISA(G, s, t)
    Output $ISA_2(G, s, t, n)$
function $ISA_2(G, s, t, k)$
    if $k \leq \lg(n)$ **then**
        Output SA(G, s, t, k)
    let V_r be a list of randomly sampled nodes from V.
    for $v \in V_r$ **do**
        if $ISA_2(G, s, v, k/2 + k/\lg(n))) \wedge ISA_2(G, v, t, k/2 + k/\lg(n)) =$ TRUE **then**
            Output TRUE
    Output FALSE

In Savitch's algorithm we try to guess the middle vertex in a path. Our improvement derives from the fact that we only look for a vertex which is approximately in the middle.

Savitch's algorithm considers the subproblem of $\leq k$ connectivity. We consider a more relaxed notion described below.

$ISA_2(G, s, t, k)$ returns YES if there exists a path of length at most k between s and t and returns NO if there is no path between s and t. ISA_2 as implemented may return either YES or NO if there is a path from s to t but no path of length at most k. Returning YES on larger length paths does not affect correctness as in st-connectivity we are only concerned about existence of a path, not a bounded length path.

Suppose there exists an at most k length path between s and t. We guess a vertex v hoping that v is one of the middle $2\delta_k = 2k/\lg(n)$ vertices in this path. We will guess v uniformly at random. The probability of a correct guess is $\epsilon_k = 2\delta_k/n$.

If our guess is correct distance between (s, v) and (v, t) are at most $k/2 + \delta_k$ hence we can recurse on $ISA_2(G, s, v, k/2 + \delta_k)$ and $ISA_2(G, v, t, k/2 + \delta_k)$.

Let $T(n, k, s, t)$ be the random variable which denotes the time for solving $ISA_2(G, s, t, k)$ using Algorithm 2.

Let $T_E(n,k) = \max_{s,t} \mathbb{E}[T(n,k,s,t)]$.

Setting $\delta_k = \max(k/(\lg n), 1)$ gives us the following recurrence for $k \geq \lg(n)$:

$$T_E(n,k) \leq \sum_{i=1}^{\infty} \Pr[i^{th} \text{ guess was the first correct guess}] \cdot i \cdot (2T_E(n, k/2 + \delta_k)) + \text{poly}(n))$$

$$T_E(n,k) \leq \sum_{i=1}^{\infty} \left((1-\epsilon_k)^{i-1} \epsilon_k \cdot i \cdot (2T_E(n, k/2 + \delta_k)) + \text{poly}(n)\right).$$

We can rewrite this equation as:

$$\begin{aligned} T_E(n,k) &\leq \left(\sum_{i=1}^{\infty} i\left((1-\epsilon_k)^{i-1}\epsilon_k\right)\right) \cdot (2T_E(n, k/2 + \delta_k) + \text{poly}(n)) \\ &\leq (1/\epsilon_k) \cdot (2T_E\left(n, k/2 + \delta_k\right) + \text{poly}(n)) \\ &\leq 2\,(1/\epsilon_k) \cdot T_E\left(n, k/2 + \delta_k\right) + \text{poly}(n). \end{aligned}$$

Now we can plug in $\epsilon_k = 2\delta_k/n$ and $\delta_k = \max(k/(\lg n), 1)$ to get the following:

$$\begin{aligned} T_E(n,k) &\leq (n/\delta_k) \cdot T_E\left(n, k/2 + \delta_k\right) + \text{poly}(n) \\ &\leq \frac{n \lg(n)}{k} \cdot T_E\left(n, k/2 + k/\lg(n)\right) + \text{poly}(n). \end{aligned}$$

For $k \geq \lg(n)$, at each recursion step k gets multiplied by $(1/2 + 1/\lg(n))$. Let k_j denote its value after j steps and k_0 denotes its initial value. For directed st connectivity $k_0 = n$.

For $k_0 = n$ and for $j \leq b = \lceil -\frac{\lg(n/\lg(n))}{\lg(1/2+1/\lg(n))} \rceil$ we have $k_j = k_0(1/2+1/\lg(n))^j = n(1/2+1/\lg(n))^j$ as for $j < b$, $n(1/2+1/\lg(n))^j > \lg(n)$. Note that b is positive ($\lg(1/2+1/\lg(n))$ is negative). Note that $k_b = n(1/2+1/\lg(n))^j \leq \lg(n)$.

$$T_E(n,n) \leq \left(\prod_{j=0}^{b-1} \frac{n\lg(n)}{k_j}\right) \cdot (T_E(n,k_b) + \text{poly}(n)).$$

As $k_j = n(1/2+1/\lg(n))^j$ for $j \leq b$ and $k_b \leq \lg(n)$

$$T_E(n,n) \leq \left(\prod_{j=0}^{b-1} \frac{n\lg(n)}{n(1/2+1/\lg(n))^j}\right) \cdot (T_E(n,\lg(n)) + \text{poly}(n)).$$

When we are looking for paths of length at most $\lg(n)$ we will call the original Savitch's algorithm. So, $T_E(n,\lg(n)) = T_S(n,\lg(n)) = \tilde{O}(n^{\lg\lg n}) = n^{o(\lg(n))}$ by Lemma 7. Recalling that b is positive and $b = \lceil -\frac{\lg(n/\lg(n))}{\lg(1/2+1/\lg(n))} \rceil$:

$$\begin{aligned} T_E(n,n) &\leq \left(\prod_{j=0}^{b-1} \frac{\lg(n)}{(1/2+1/\lg(n))^j}\right) \cdot (n^{o(\lg(n))} + \text{poly}(n)) \\ &\leq \left(\frac{\lg^b(n)}{(1/2+1/\lg(n))^{b^2/2}}\right) \cdot (n^{o(\lg(n))}) \end{aligned}$$

As $b = \lceil -\frac{\lg(n/\lg(n))}{\lg(1/2+1/\lg(n))} \rceil$, $\lg^b(n) = n^{o(\lg(n))}$ we have

$$\begin{aligned}
T_E(n,n) &\leq (1/2+1/\lg(n))^{-b^2/2} \cdot n^{o(\lg(n))} \\
&\leq 2^{-\frac{\lg(1/2+1/\lg(n))\lg^2(n)}{2\lg(1/2+1/\lg(n))^2}} \cdot n^{o(\lg(n))} \\
&\leq 2^{-\frac{\lg^2(n)}{2\lg(1/2+1/\lg(n))}} \cdot n^{o(\lg(n))} \\
&\leq 2^{\frac{\lg^2(n)}{2(1-\lg(1+2/\lg(n)))}} \cdot n^{o(\lg(n))} \\
&\leq 2^{\frac{\lg^2(n)}{2(1-2/\lg(n)))}} \cdot n^{o(\lg(n))} \\
&\leq 2^{\frac{\lg^2(n)}{2}} \cdot n^{o(\lg(n))} \\
&\leq n^{(\lg(n)/2+o(\lg(n)))}.
\end{aligned}$$

So $T_E(n,n) \leq n^{(\lg(n)/2+o(\lg(n)))}$, giving the desired result. ◀

4 Reductions Between k-cycle Problems

In this section we will give a reduction from directed k' cycle to $2k$ cycle, where k' is odd and $k' \leq k/\lg^{O(1)}(k)$. This will allow us to give lower bounds for even cycle as well as odd cycle in the sparse setting. First, we will need a helper lemma about the existence of primes with appropriate properties.

▶ **Lemma 8.** *There exists a constant $k^\star$ such that $\forall k \geq k^\star$ we can write $2k = \sum_{i=0}^{4} z_i r_i$ where $z_i \in \mathbb{N}, r_i \leq \lg^{O(1)}(k), \sum |z_i r_i - 2k/5| \leq \lg^{O(1)}(k)$ and any four of r_i's together have a common prime factor which does not divide k.*

Proof. Let p_i for $0 \leq i \leq 4$ be distinct prime numbers which do not divide k. By the prime number theorem we can take $p_i \leq \lg^2(k)$ for large enough constant k. Define $r_i = \prod_{j \neq i} p_j$. This allows us to satisfy the requirements that $r_i \leq \lg^{O(1)}(k)$ and any four of r_i's together have a common prime factor which does not divide k.

As the $gcd(r_i) = 1$, by Eulid's gcd algorithm there exists integers y_i such that $1 = \sum y_i r_i$ hence there exists integers z_i for which $2k = \sum z_i r_i$. Choose z_i's such that $\sum |z_i r_i - 2k/5|$ is minimized. Assume $\sum |z_i r_i - 2k/5| > 10 \lg^{11}(k)$. This implies that there exists a j, j' such that $z_j r_j - 2k/5 > \lg^{11}(k)$ and $z_{j'} r_{j'} - 2k/5 < -\lg^{11}(k)$. We can substitute z_j with $z_j - p_j$ and $z_{j'}$ with $z_{j'} + p_{j'}$. This maintains the sum $\sum z_i r_i = 2k$ as $r_j p_j = r_{j'} p_{j'}$. The substitution decreases $z_j r_j - 2k/5$ by $\prod p_i < \lg^{10}(k) < \lg^{11}(k)$ and increases $z_{j'} r_{j'} - 2k/5$ by $\prod p_i < \lg^{10}(k) < \lg^{11}(k)$. Hence both $|z_j r_j - 2k/5|$ and $|z_{j'} r_{j'} - 2k/5|$ decrease while $|z_i r_i - 2k/5|$ for $i \neq j, j'$ remain the same. This gives a contradiction as we chose z_i's to minimize $\sum |z_i r_i - 2k/5|$. So we can assume that $\sum |z_i r_i - 2k/5| \leq 10 \lg^{11}(k)$. This also implies that for all i, $z_i r_i > 2k/5 - \lg^{O(1)}(k)$ hence $z_i > 0$ for large enough k. ◀

Given a sparse graph G there are many possible ways for the input to be formatted. Given an adjacency list format there exist two different efficient algorithms for returning if (u, v) is an edge in the graph. If one is worried about space and not time (as we are in our $O(\lg(n))$ space lower bounds) there is a $O(\lg(n))$ space $O(\lg(n))$ time algorithm. Simply binary search through the adjacency list input to find (u, v). If one is worried about time and not space then one can hash the adjacency list giving a $O(n)$ space data structure that has lookup time $O(1)$. We don't care about $\lg(n)$ factors in our time and thus default to the $O(\lg(n))$ space and time algorithm.

▶ **Theorem 9.** *There exists a constant $k^\star$ such that $\forall k \geq k^\star$, there exists $k' \geq k/(\lg^{O(1)} k)$ such that a n-node, m edge instance of directed k' cycle can be reduced to $\tilde{O}(m)$ node, $\tilde{O}(m)$ edge instance of $2k$ cycle on graph G. Further any bit in the description of G can be produced by a $O(\lg(n))$ space and $O(\lg(n))$ time.*

Proof. Take z_i, r_i's satisfying Lemma 8 for k. Set $k' = \sum z_i$. Let G be a instance of directed k' cycle with n nodes and m edges. We will first apply color coding using Lemma 20, to produce a k' partite graph. This graph will only have edges between "adjacent" partitions. As explained in Lemma 20.

Let $s_i = \sum_{j<i} z_j$. We create a new graph G' by replacing each directed edge in between the s_i to the $s_{(i+1 \mod 5)}$(cyclically) parts by a r_i length path with undirected edges. As we are replacing z_i partitions of edges by an r_i length path this gives a natural partition of G' with $\sum z_i r_i = 2k$ parts.

$$\begin{aligned}\text{Number of nodes and edges in } G' &\leq \max(r_i) \cdot m \\ &\leq \lg^{O(1)}(k) \cdot m \quad \text{[By Lemma 8]} \\ &\leq O(m) \qquad \text{[As } k \text{ is a constant]}\end{aligned}$$

We will now show that the new graph G' has a $2k$-cycle iff G had a directed k'-cycle.

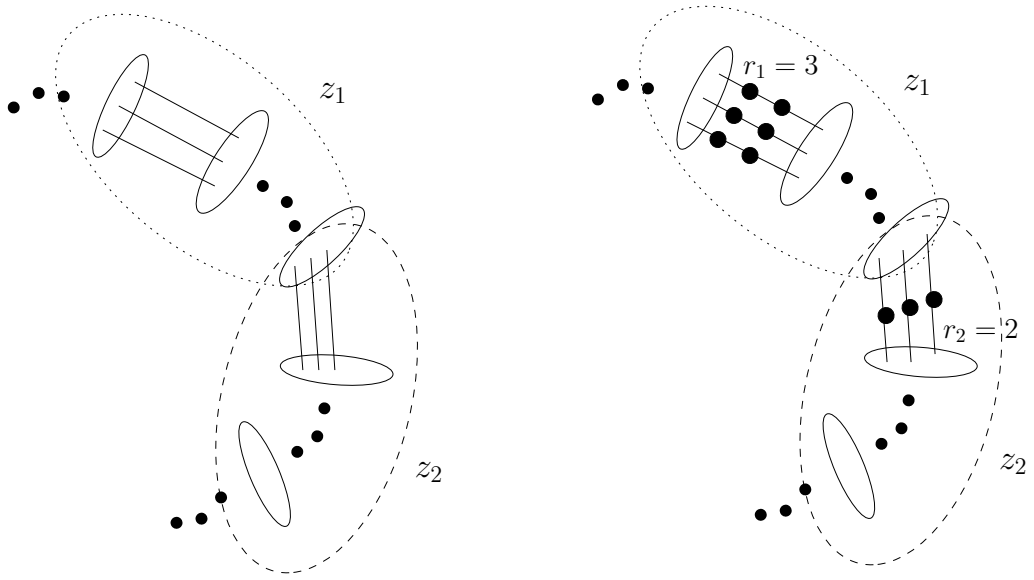


Figure 1 Depicting the edge transformation from G to G'. The dotted oval contains z_1 edge sets. The dashed oval contains z_2 edge sets. Every edge in the dotted oval is turned into r_1 edges by replacing the edge with a path. Every edge in the dashed oval is turned into r_2 edges by replacing the edge with a path.

Each edge in the graph G' is part of one of the introduced r_i length paths. If one of the edges from an r_i length path is used for a $2k$ cycle then all the edges in the path must be used as intermediate vertices in the path have no other edges that the edges in the path itself. Hence we can think of our $2k$ cycle as composed of meta-edges of lengths r_i. Suppose for constructing the $2k$ cycle we only use at most 4 out of the 5 possible r_i lengths. Then we cannot construct a $2k$ cycle as these four r_i's share a common prime factor which does not occur in $2k$. Hence we need to use all 5 possible r_i lengths. Wlog we can assume that we start from 0^{th} partition. There are 2 approaches to hitting all 5 possible r_i length paths:

1. Go around all the $2k$ partitions.
2. Do not go around all partitions.

For type 1, note that we can never back-track as otherwise we will have more than $2k$ edges. If we find a $2k$ cycle using type 1 without backtracking this exactly corresponds to a directed k' cycle in G by replacing all meta-edges by the corresponding edge of G.

For type 2 paths note that we must hit all of the 5 possible lengths.

Let group i be the set of edges produced in G' from the edges between the s_i and $s^{th}_{i+1 \mod 5}$ partitions in G. To pass from one side of group i to the next requires passing through at least z_i paths of length r_i. So passing from one side of group i to the other requires a path of length at least $z_i r_i$.

To hit all possible lengths we must have at least one edge from all five groups. To pass through all five groups at minimum requires doubling back through three groups. To double back through a group requires two paths each of length $z_i r_i$. See Figure2 for a depiction.

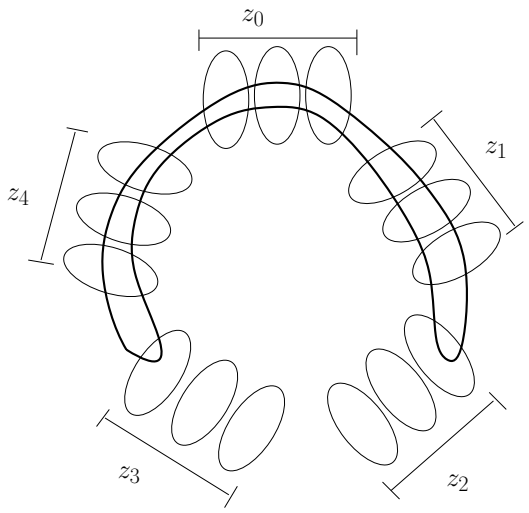


Figure 2 In bold we depict a path passing through partitions, reaching at least one r_i length path of each of the 5 possible lengths. Each oval represents a partition of s_j. Between each partition is a path of one of the lengths r_i. Note that to reach all five the path must "double back" through three partitions.

For simplicity understand r_{i+c} and z_{i+c} in the next equation to be $r_{(i+c \mod 5)}$ and $z_{(i+c \mod 5)}$ respectively. So the minimum length of a type two cycle is

$$\min_i \left(2r_i + 2z_{i+1}r_{i+1} + 2z_{i+2}r_{i+2} + 2z_{i+3}r_{i+3} + 2r_{i+4}\right).$$

By Lemma 8 $\sum|z_j r_j - 2k/5| \leq \lg^{O(1)}(k)$, so $\sum|2z_j r_j - 4k/5| \leq 2\lg^{O(1)}(k)$.
Thus, $\sum_{j=i+1}^{i+3}|2z_j r_j - 4k/5| \leq 2\lg^{O(1)}(k)$. Finally giving

$$\sum_{j=i+1}^{i+3} 2z_j r_j \geq \frac{12}{5}k - 2\lg^{O(1)}(k) = 2.4k - 2\lg^{O(1)}(k) > 2k.$$

A type 2 $2k$-cycle, which doesn't form a cycle through all partitions, must have length greater than $2k$. So, no such cycle exists.

Thus, the only $2k$-cycles in this graph correspond exactly to a directed k'-cycle in the original graph. ◀

5 Implications of k-Cycle Reductions

We now discuss the implications of the reduction from directed k'-cycle to undirected $2k$-cycle. We will start by showing that the algorithms for $2k$-cycle in $\lg(n)$ space are tight up to constant factors in the exponent. Next, we will give lower bounds for $2k$-cycle from Max-3-SAT.

5.1 2k-Cycle $\lg(n)$ Space Algorithms are Tight

Given the Weak Savitch Hypothesis we can show that undirected $2k$-cycle problems have nearly optimal algorithms.

In this section we give a reduction that, for all large enough k, reduces directed k'-cycle to undirected $2k$-cycle. That is, there is some constant $k^\star$ such that for all $k > k^\star$ we can reduce directed k'-cycle to undirected $2k$-cycle. Furthermore, in our reduction $k' = \Theta(k/\lg(k))$. So, for all $k > k^\star$ we have that $2k$-cycle reduces from directed $\Theta(k/\lg(k))$-cycle.

Following Lemmas 10 and 11 are very similar to lemmas in previous work and are roughly folklore. We present their proofs for completeness.

▶ **Lemma 10.** *If the Weak Savitch Hypothesis is true then $\leq k-$connectivity in an directed graph G with n nodes requires $n^{\Omega(\lg k)}$ time if we are restricted to $O(\lg n)$ space.*

Proof. We can consider the Savitch algorithm for determining if there is a path of length $\leq \ell$ between two nodes in a graph with n nodes. The Savitch algorithm [12] as described in Section 3 has the following recurrence :

$$T(n, \ell) \leq 2nT(n, \ell/2) + \text{poly}(n).$$

Where the standard base case is $T(n, 1) = O(\lg(n))$. This gives an algorithm running in time $O(n^{(\lg \ell)}\text{poly}(n))$.

If Weak Savitch Hypothesis is true then there exists some constant $\epsilon \geq 0$ such that no algorithm solving st connectivity (equivalent to $\leq n$ connectivity) in $\lg^2(n)$ space runs in time $n^{\epsilon \lg(ell)}$.

If an algorithm for k-path runs in $n^{o(\lg(k))}$ and $O(\lg n)$ space then we can repeatedly use it to find an $\leq \ell$ length path. Specifically, we will run the k-path algorithm where we treat an edge as existing if there is an ℓ/k walk between two nodes. To determine if such an edge exists we will recurse. This gives us the following recurrence:

$$T(n, \ell) \leq n^{o(\lg(k))}T(n, \ell/k) + \tilde{O}(1).$$

For $\ell = n$, this recurrence gives us $T(n, n) \leq n^{o(\lg(k))\lg_k(n)}$ This running time can be simplified to $O(n^{o(\lg(k))\lg(n)/\lg(k)})$.

For all ϵ there exists some sufficiently large but constant k such that $o(\lg(k))/\lg(k) \leq \epsilon$ which contradicts the Weak Savitch Hypothesis. ◀

▶ **Lemma 11.** *Assuming Weak Savitch Hypothesis (WSH) directed, k-cycle cannot be solved in $O(\lg n)$ space and $n^{o(\lg k)}$ time.*

Proof. Suppose directed k-cycle is in $O(\lg n)$ space and $n^{o(\lg k)}$ time. Suppose we are given an instance of $k-1$-walk $(G = (V, E), a, b)$. Construct a new graph G' with k copies of each vertex v named $v_1, v_2, \ldots, v_k$. Add u_i, u_{i+1} as an edge. Add (u_i, v_{i+1}) as an edge in G' if (u, v) was an edge in G. Also add (a_k, b_1) as an edge. Now there is a k-cycle in G' iff there was a $k-1$-walk from a to b in G. As the number of vertices in G' is kn and k

is a constant hence we have a $O(\lg n)$ space and $n^{o(\lg k)}$ time algorithm for (un)directed k-walk. By Lemma 10 this contradicts Weak Savitch Hypothesis (WSH). Note that we will not actually construct G' instead we use $O(\lg n)$ space and have a $O(\lg n)$ time algorithm that serves as an oracle for G'. ◀

▶ **Reminder of Theorem 3.** *Assuming Weak Savitch Hypothesis (WSH) solving k-cycle in undirected graphs in $O(\lg n)$ space requires $n^{\Omega(\lg k)}$ time.*

Proof. By Lemma 11 we know that k'-cycle in directed graphs cannot be solved in $O(\lg n)$ space and $n^{o(\lg k')}$.

Suppose we have a $O(\lg n)$ space and $n^{o(\lg k)}$ time algorithm for k-cycle in undirected sparse graphs for even k. Then let k' be the largest integer satisfying the definition in Theorem 9. Then by Theorem 9 we can reduce directed k'-cycle to k-cycle and solve it in time $n^{o(\lg k)} = n^{o(\lg k')}$ because $k' \geq k/\lg^{O(1)}(k)$. Note that as in the reduction we can produce each bit in $O(\lg(n))$ space and time $O(\lg(n))$ our space and running time bounds are not affected. The reduction only increases the number of vertices by a constant factor and k is large enough. Using Lemma 11 gives a contradiction. ◀

▶ **Corollary 12.** *Assuming Weak Savitch Hypothesis (WSH) detecting k-cycles in undirected sparse graphs in $O(\lg(n))$ space takes $n^{\Theta(\lg k)}$ time.*

Proof. By Theorem 3 we have that k-cycle in log space requires $n^{\Omega(\lg k)}$ time. By Lemma 21 we have that k-cycle can be solved in log space and $n^{O(\lg k)}$ time. Thus, the k-cycle problem requires $n^{\Theta(\lg k)}$ time, conditioned on WSH.

◀

Note that we can replace the Weak Savitch Hypothesis with the Strong Savitch Hypothesis in these proofs. They will in fact make the statements stronger. If the Strong Savitch Hypothesis is true then k-cycle in undirected graphs cannot be solved in $O(\text{poly}\lg(n))$ space and $n^{o(\lg k)}$ time.

5.2 Even Cycle Lower Bounds From Max-CSP

Previous work [10] has connected efficiently detecting directed k-cycles in sparse graphs with solving the Max-L-SAT problem [10]. In fact, this work connects directed k-cycle detection to Max-L-CSP problems, where every clause is representable by a L degree Boolean polynomial. In this section we will limit our focus to the Max-3-SAT and Max-L-SAT problems.

If, for some particular constant L, one believes that the Max-L-SAT problem requires $2^{n(1-o(1))}$ time, then from this we get a super linear lower bound for constant length even cycle. In fact, we get a super linear lower bound for constant length undirected even cycle in sparse graphs where $m = O(n)$. Such a super linear linear lower bound for undirected even cycle has not previously been obtained.

Our reduction provides a lower bound for sparse undirected even cycle from assumptions made about difficult exponential time Constraint Satisfaction Problems (CSPs). Notably, we are able to provide super linear lower bounds for even cycles from assumptions stated in the paper of Lincoln, Vassilevska Williams and Williams [10].

▶ **Theorem 13** (Sparse Directed k-Cycle is Hard [10])**.** *If directed k-cycle, for any large enough k, can be solved in a graph G in time $O(m^{c_k-\epsilon})$, where $c_k = k/(k - \lceil k/3 \rceil + 1)$ then max-3-SAT can be solved in time $O^*(2^{(1-\epsilon')n})$.*

Using our reduction and the above theorem we can provide a lower bound for undirected sparse cycle.

▶ **Reminder of Theorem 5.** *If undirected $2k$-cycle, for any large enough constant k can be solved in a graph G with $m = O(n)$ in time $O(m^{1.5-\epsilon})$ then max-3-SAT can be solved in time $O^*(2^{(1-\epsilon')n})$.*

Proof. By applying Theorem 9 we transform an input graph into one that has $m = O(n)$. Furthermore, it shows that if undirected $2k$-cycle, for all constant k, can be solved in time $O(m^{1.5-\epsilon})$ then, there exists an infinite sequence of increasing odd cycle lengths solved in time $O(m^{1.5-\epsilon})$.

Given an infinite increasing sequence of odd cycle lengths they solve the same length of directed cycle. Then, there exists some $k \geq \frac{1}{\epsilon}3 \geq \frac{1}{\epsilon}\frac{9}{4}$ which, if solved in $O(m^{1.5-\epsilon})$ time is solved faster than $O(m^{c_k-\epsilon})$, where $c_k = k/(k - \lceil k/3 \rceil + 1)$ time.

Thus, undirected even cycle can not be solved in time $O(m^{1.5-\epsilon})$ ◀

We can give a more precise result due to our reduction's tight relationship between $k' = \lg^{O(1)}(k)k$. Note that $2k$-cycle can be reduced to k'-cycle where $k' = \Theta(2k/\lg^{O(1)}(2k))$ then. Given this $2k$-cycle requires $\Omega(m^{c_{k'}-o(1)})$, where $c_{k'} = k'/(k' - \lceil k'/3 \rceil + 1)$. Thus we have that $2k$- cycle requires at least $\Omega(n^{1.5-O(\lg^{O(1)}(k)/k)})$ time.

We would like to note that no super-linear hardness was known for even cycles prior to this. And, here, we have connected the hardness of even cycles to that of Max-3-SAT.

We would further like to note that we can combine our reduction with the more general result of [10], which gets super linear hardness for directed cycle from Max-L-SAT. Max-L-SAT is more believably $2^{n-o(n)}$ hard the larger L is. From Max-L-SAT we continue to show super-linear hardness for undirected $2k$-cycle problems. However, instead of showing $O(m^{1.5-O(1)})$ hardness the result is instead $O(m^{\frac{L}{L-1}-o(1)})$ hardness.

5.3 Combinatorial Even Cycle Lower Bounds From the Combinatorial K-Clique Conjecture

Abboud, Backurs and, Vassilevska Williams present the Combinatorial k-Clique Hypothesis (CKCH) [1]. First, what is a *combinatorial* algorithm? In her survey Vassilevska Williams defines it as follows: *"The desire for more practical algorithms motivates the notion of "combinatorial" algorithms. This notion is not well-defined, however it roughly means that the runtime should have a small constant in the big-O, and that the algorithm is feasibly implementable.[16]"* Generally, combinatorial assumptions assume that matrix multiplication requires $n^{3-o(1)}$ time, that is there is no fast matrix multiplication allowed. Given this context, we now re-produce the Combinatorial k-Clique Hypothesis from [1].

▶ **Definition 14** (Combinatorial k-Clique Hypothesis [1])**.** *Let C be the smallest constant such that a* combinatorial *algorithm running in $O(n^{Ck/3})$ time exists for detecting a k-clique in a n node $O(n^2)$ edge graph, for all sufficiently large constants k.*

The Combinatorial k-Clique Hypothesis states that $C = 3$[1]*.*

More informally, no $O(n^{k(1-\epsilon)})$ *combinatorial* algorithm exists for the k-clique problem.

We can once again use the k-clique to k-cycle reduction of Lincoln, Vassilevska Williams and Williams [10]. Their reduction is itself combinatorial.

[1] See the discussion on page 2 of Abboud, Backurs and, Vassilevska Williams [1]. For a short formal statement see Hypothesis 1.3 in Lincoln, Vassilevska Williams and Williams [10].

▶ **Theorem 15** (Combinatorial Sparse k-Cycle Lower Bound [10]). *Detecting a directed odd k-cycle in a graph with n nodes and $m = n^{(k+1)/(k-1)}$ in time $O((nm)^{(1-\epsilon)}) = O(m^{(1-\epsilon)2k/(k+1)}) = O(n^{(1-\epsilon)2k/(k-1)})$ for any constant $\epsilon > 0$ violates the Combinatorial k-Clique Hypothesis* [2].

▶ **Corollary 16.** *Detecting a directed k-cycle in a graph with n nodes and $m = n^{k/(k-2)}$ in time $O(nm^{(1-\epsilon)}) = O(m^{(2k-2)/k})$ for any constant $\epsilon > 0$ violates the Combinatorial k-Clique Hypothesis.*

Proof. There is an immediate reduction from directed k-cycle to directed $k+1$-cycle that adds $O(n)$ edges to the graph to get a lower bound for even k. For odd k we have a better lower bound. ◀

Recall that Dahlgaard et al. have an algorithm for even $2k$-cycle detection that runs in $\tilde{O}\left(m^{2k/(k+1)}\right)$ time [5]. We will show that this algorithm is optimal for even $k > 3$ that are twice an odd number. That is, the half of even numbers represented by $4k+2$ for integer $k \geq 1$.

▶ **Theorem 17.** *Let $2k = (4k'+2)$ for some constant integer k'.*
Detecting $2k$-cycles combinatorially requires time $m^{(1-o_k(1))2k/(k+1)}$ if the Combinatorial k-Clique Hypothesis is true.

Proof. We will start with Theorem 15. Note that $2k'+1$ is odd. Take the directed instance and use color coding to produce a $2k'+1$-partite undirected instance. Furthermore, this graph is a $2k'+1$-cycle graph. That is, partition $P_{(i)}$ only has edges to partitions $P_{(i-1 \mod (k'+1))}$ and $P_{(i+1 \mod (2k'+1))}$.

To solve an undirected odd $2k'+1$-cycle problem in a $2k'+1$-cycle graph we will create an instance of $4k'+2$-cycle by replacing every edge (u, v) with a node, x_{uv} and two edges $(u, x_{u,v})$ and $(x_{u,v}, v)$.

Any $4k'+2$-cycle in this new graph with contain $2k'+1$ x_{uv} type nodes. In fact these x_{uv} type nodes will be every other node. If an x_{uv} type node is in a cycle so are u and v. So, a $4k'+2$ length path will look like $v_1, x_{v_1v_2}, v_2, x_{v_2v_3}, v_3 \ldots, v_{2k'}, x_{v_{2k'}v_{2k'+1}}, v_{2k'+1}$. The existence of a $4k'+2$ length path in our new graph corresponds exactly to there being a $2k'+1$ length path in the original graph.

Theorem 15 tells us that there is a $m^{(4k'+2)/(2k'+2)-o(1)}$ combinatorial lower bound if Combinatorial k-Clique Hypothesis is true. In our new graph m has grown by a factor of only two. If we plug in $2k = 4k'+2$ we get the desired lower bound of $m^{2k/(k+1)-o(1)}$ from Combinatorial k-Clique Hypothesis. ◀

The above result works for specific even cycles i.e. when $2k = 4k'+2$, next we prove a lower bound which holds for $2k$ cycles for all large enough k.

▶ **Corollary 18.** *If there exists a constant $\epsilon > 0$ such that for some $k > \frac{2}{\epsilon}$ directed k-cycle can be detected in $O(m^{2-\epsilon})$ time using a combinatorial algorithm then the Combinatorial k-Clique Hypothesis is violated.*

Proof. For $k > \frac{2}{\epsilon}$, $m^{2-\epsilon} = O(m^{(2k-2)/(k)(1-\delta)})$ for some constant δ. So, by Corollary 16 such an algorithm would violate the Combinatorial k-Clique Hypothesis. ◀

[2] See the discussion after Hypothesis 1.3 in the introduction of Lincoln, Vassilevska Williams and Williams [10].

▶ **Theorem 19.** *Assume the Combinatorial k-Clique Hypothesis. Then, for all constant $\epsilon > 0$ there exits a k_0 such that for all $k \geq k_0$ even $2k$-cycles in an undirected graph with n nodes and $m = O(n)$ edges requires $\Omega(m^{2-\epsilon})$ time.*

Proof. By Theorem 9 there exits a $k' \geq k/(\lg^{O(1)} k)$ such that if directed k' cycle requires $\Omega(m^{2-\epsilon})$ then so does undirected $2k$ cycle on a graph with $O(m)$ vertices and $O(m)$ edges.

As long as $k' = k/(\lg^{O(1)} k) > 2/\epsilon$ we know that directed k' cycle is $\Omega(m^{2-\epsilon})$ hard. As for a large enough k, $k/(\lg^{O(1)} k) > 2/\epsilon$ we have that for large enough k undirected $2k$ cycle on graphs with $m = O(n)$ requires $\Omega(m^{2-\epsilon})$ time. ◀

Note that for all constant k there exist $O(n^2)$ combinatorial algorithms for undirected $2k$-cycle [17]. So, this suggests that these algorithms are optimal, for *combinatorial* algorithms for all large enough k.

6 Conclusion

We present a faster algorithm for small space cycle detection. We also present lower bounds for the efficiency of k-cycle in this small space regime. We show the running time of odd as well as even k-cycle must grow as $n^{\Omega(\lg(k))}$ when given only $O(\lg^2(n))$ space.

There are many future directions of research. We leave open several algorithms questions as well as questions about lower bounds.

While we get tight upper and lower bounds in the regime we consider, there are many open questions about k-cycle outside of this regime. What is the space-time trade-off for these k-cycle algorithms when the space available is polynomial in n? For example, how efficient are algorithms when given $n^{1/8}$ space? Additionally, our results apply to large constant k, but do not imply statements for specific fixed k. What are the fastest small space algorithms for $k = 3, 4, 5 \ldots$? Finally, our results are not very *fine-grained*, we are agnostic to the specific constants in the exponents of these algorithms. So, what is the specific constant? Can k-cycle algorithms be solved in small space in $n^{\lg(k)/2}$ time? $n^{\lg(k)/100}$ time?

We also leave open related algorithmic questions about s t connectivity in small space. We present a s t connectivity algorithm in small space which runs in $n^{\lg(n)/2}$ expected time. Can a deterministic algorithm have the same running time? What are the fastest randomized and deterministic algorithms for the s t connectivity problem?

We leave open lower bounds questions as well. Can we find matching, or at least non-trivial, lower bounds for k-cycle algorithms in small polynomial space. Relatedly, we get combinatorial lower bounds for even cycle in arbitrary space. Can one find tight *non-combinatorial* lower bounds for even cycle in arbitrary space?

References

1. Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. If the Current Clique Algorithms are Optimal, So is Valiant's Parser. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 98–117, 2015. doi:10.1109/FOCS.2015.16.
2. Noga Alon, Raphael Yuster, and Uri Zwick. Finding and Counting Given Length Cycles. *Algorithmica*, 17(3):209–223, 1997. doi:10.1007/BF02523189.
3. Noga Alon, Raphael Yuster, and Uri Zwick. Color Coding. In *Encyclopedia of Algorithms*, pages 335–338. Springer, 2016. doi:10.1007/978-1-4939-2864-4_76.
4. Greg Barnes, Jonathan F. Buss, Walter L. Ruzzo, and Baruch Schieber. A Sublinear Space, Polynomial Time Algorithm for Directed s-t Connectivity. *SIAM J. Comput.*, 27(5):1273–1282, 1998. doi:10.1137/S0097539793283151.

5 Søren Dahlgaard, Mathias Bæk Tejs Knudsen, and Morten Stöckel. Finding even cycles faster via capped k-walks. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 112–120, 2017. doi:10.1145/3055399.3055459.

6 François Le Gall. Powers of tensors and fast matrix multiplication. In *International Symposium on Symbolic and Algebraic Computation, ISSAC '14, Kobe, Japan, July 23-25, 2014*, pages 296–303, 2014. doi:10.1145/2608628.2608664.

7 Parikshit Gopalan, Richard J. Lipton, and Aranyak Mehta. Randomized Time-Space Tradeoffs for Directed Graph Connectivity. In Paritosh K. Pandya and Jaikumar Radhakrishnan, editors, *FST TCS 2003: Foundations of Software Technology and Theoretical Computer Science, 23rd Conference, Mumbai, India, December 15-17, 2003, Proceedings*, volume 2914 of *Lecture Notes in Computer Science*, pages 208–216. Springer, 2003. doi:10.1007/978-3-540-24597-1_18.

8 Richard C. Holt and Edward M. Reingold. On the time required to detect cycles and connectivity in graphs. *Mathematical systems theory*, 6(1):103–106, March 1972. doi:10.1007/BF01706081.

9 Richard M. Karp. Reducibility Among Combinatorial Problems. In *Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, USA*, pages 85–103, 1972. doi:10.1007/978-1-4684-2001-2_9.

10 Andrea Lincoln, Virginia Vassilevska Williams, and R. Ryan Williams. Tight Hardness for Shortest Cycles and Paths in Sparse Graphs. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 1236–1252, 2018.

11 Benjamin Rossman. Formulas vs. circuits for small distance connectivity. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 203–212, 2014.

12 Walter J. Savitch. Relationships Between Nondeterministic and Deterministic Tape Complexities. *J. Comput. Syst. Sci.*, 4(2):177–192, 1970. doi:10.1016/S0022-0000(70)80006-X.

13 Dieter van Melkebeek and Gautam Prakriya. Derandomizing Isolation in Space-Bounded Settings. *SIAM J. Comput.*, 48(3):979–1021, 2019. doi:10.1137/17M1130538.

14 Avi Wigderson. The Complexity of Graph Connectivity. In *Mathematical Foundations of Computer Science 1992, 17th International Symposium, MFCS'92, Prague, Czechoslovakia, August 24-28, 1992, Proceedings*, pages 112–132, 1992.

15 Virginia Vassilevska Williams. Multiplying matrices faster than coppersmith-winograd. In *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 - 22, 2012*, pages 887–898, 2012. doi:10.1145/2213977.2214056.

16 Virginia Vassilevska Williams. On some fine-grained questions in algorithms and complexity. In *Proceedings of the International Congress of Mathematicians, page to appear*, 2018.

17 Raphael Yuster and Uri Zwick. Finding Even Cycles Even Faster. *SIAM J. Discrete Math.*, 10(2):209–222, 1997. doi:10.1137/S0895480194274133.

18 Raphael Yuster and Uri Zwick. Detecting Short Directed Cycles Using Rectangular Matrix Multiplication and Dynamic Programming. In *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '04, pages 254–260, Philadelphia, PA, USA, 2004. Society for Industrial and Applied Mathematics. URL: http://dl.acm.org/citation.cfm?id=982792.982828.

19 Uri Zwick. All pairs shortest paths using bridging sets and rectangular matrix multiplication. *J. ACM*, 49(3):289–317, 2002. doi:10.1145/567112.567114.

A Basic Lemmas for the Paper

The ideas in various lemmas below are not novel. These are just important working lemmas for the rest of the paper. We cite lemmas we use directly. For those lemmas that are variations on previous work we attempt to reference where the ideas came from.

We will be using color coding to simplify many proofs. To see an example of the output graphs from color coding look at Figure 3.

▶ **Lemma 20** (Color Coding [3]). *Let E be the edge set of the graph G. Given a graph G one can produce $c = \lg(n)^{k^2}$ graphs $G_1, \ldots, G_c$ in $\tilde{O}(|E|)$ time such that:*

- *each graph has k partitions $P_1, \ldots, P_k$ and edges exist only between P_i and $P_{i+1 \mod k}$,*
- *each graph G_i has at most $|E|$ edges and $|V|$ vertices,*
- *if and only if G contains a k-cycle, at least one of the graphs G_i will contains a k-cycle,*
- *and finally if G contains a k-cycle in at least one of the graphs G_i the k cycle will have one node from every partition.*

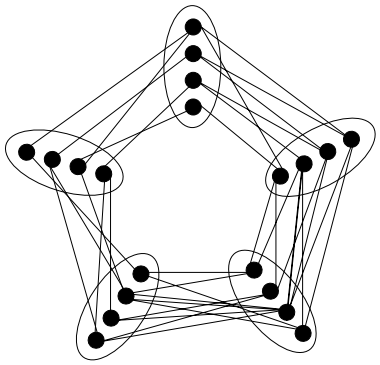


Figure 3 An example of the output of color coding with five colors. There are five partitions and the partions are themselves in a cycle.

▶ **Lemma 21.** *An algorithm exists to detect if a directed k-cycle exists using the edge (s,t) in a graph G with n nodes and $O(n^2)$ edges in time $n^{O(\lg(k))}$ using space $O(\lg(k)\lg(n))$.*

Proof. If a non-colorful cycle we start by using the color coding Lemma 20, to produce graphs with k partitions such that our k-cycle, if it exists, uses one node from each partition in one of these graphs. We consider only the color coded graphs where s and t are in partitions V_1 and V_k respectively. If we start with colorful k-cycle then we simply use the colors themselves for color coding.

Now, if we could detect if a k-path exists between s and t which uses one node from every partition we are done. To do this we will guess, in sequence, each of the n possible choices of nodes that could be in the middle partition. Then, we will recursively solve this same colorful path problem on both shorter paths. The running time of this algorithm is given by the following recurrence with base case $T(n,2) = n^{O(1)}$:

$$T(n,k) = n2T(n,k/2) + O(1).$$

Solving this recurrence gives us $T(n,k) = 2^{\lg(k)} n^{\lg(k)} n^{O(1)}$. When simplified we find $T(n,k) = n^{\lg(k)+o(\lg(k))} = n^{O(\lg(k))}$. This gives us the desired result.

Keeping these guesses in memory requires $\lg(n)$ bits per guess of a node and we must track $O(\lg(k))$ guesses at a time. This gives us $O(\lg(k)\lg(n))$ space. ◀

The above lemma is using the same ideas as Savitch [12].

High-Dimensional Expanders from Expanders

Siqi Liu
EECS Department, UC Berkeley, CA, United States
https://siqi-l.github.io/
sliu18@berkeley.edu

Sidhanth Mohanty
EECS Department, UC Berkeley, CA, United States
http://sidhanthm.com/
sidhanthm@berkeley.edu

Elizabeth Yang
EECS Department, UC Berkeley, CA, United States
https://people.eecs.berkeley.edu/~elizabeth_yang/
elizabeth_yang@berkeley.edu

Abstract

We present an elementary way to transform an expander graph into a simplicial complex where all high order random walks have a constant spectral gap, i.e., they converge rapidly to the stationary distribution. As an upshot, we obtain new constructions, as well as a natural probabilistic model to sample constant degree high-dimensional expanders.

In particular, we show that given an expander graph G, adding self loops to G and taking the tensor product of the modified graph with a high-dimensional expander produces a new high-dimensional expander. Our proof of rapid mixing of high order random walks is based on the decomposable Markov chains framework introduced by [11].

2012 ACM Subject Classification Theory of computation → Expander graphs and randomness extractors

Keywords and phrases High-Dimensional Expanders, Markov Chains

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.12

Related Version A full version of the paper is available at https://arxiv.org/abs/1907.10771.

Funding *Siqi Liu*: This research was supported in part by a Google Faculty Award and donations from the Ethereum Foundation and the Interchain Foundation.
Sidhanth Mohanty: Supported by NSF Grant CCF-1718695.
Elizabeth Yang: Supported by NSF Grant DGE 1752814.

Acknowledgements We thank Tom Gur for introducing us to this intriguing question and for helpful discussions, and we thank Nikhil Srivastava for insightful conversations. We would also like to thank the Simons Institute for the Theory of Computing where a large portion of this work was done. The third author is supported by the National Science Foundation Graduate Research Fellowship under Grant No. DGE 1752814.

1 Introduction

Expander graphs are graphs which are sparse, yet well-connected. They play important roles in applications such as the construction of pseudorandom generators and error-correcting codes [23]. Motivated by both purely theoretical questions, such as the topological overlapping problem, and applications in computer science, such as PCPs, a generalization of expansion to high dimensional complexes has recently emerged. We work with d–dimensional complexes, which not only have vertices and edges, but also hyperedges of k vertices, for any $k \leqslant d+1$. Whereas in the one-dimensional world of graphs, the properties of edge expansion, spectral

expansion and rapid mixing of random walks are equivalent, their generalization to several different characterizations of "expansion" have been developed for these high–dimensional complexes. In particular, the high-dimensional extension of spectral expansion is simple to state, and implies rapid mixing of high order walks [14] and agreement expanders [6].

We construct bounded-degree high–dimensional expanders of all constant–sized dimensions, where the high order random walks have a constant spectral gap, and thus mix rapidly. We base our HDX's from existing T-regular one-dimensional constructions, which can be sampled readily from the space of all T-regular graphs. This endows a natural distribution from which we can sample HDX's of our construction as well. After the first version of this paper was written, it was brought to our notice by a reviewer that the construction in this paper has been previously discussed in the community. Nevertheless, a contribution of our work is a rigorous analysis of the expansion properties of this construction.

One sufficient, but not necessary criterion that implies rapid mixing is *spectral*, which comes from the graph theoretic notion below.

▶ **Definition 1** (Informal). *A d–dimensional λ–spectral expander is a d–dimensional simplicial complex (i.e. a hypergraph whose faces satisfy downward closure) such that*

- *(Global Expansion) The vertices and edges (sets of two vertices) of the complex constitute a λ–spectral expander graph,*
- *(Local Expansion) For every hyperedge E of size $\leqslant d-1$ in the hypergraph, the vertices and edges in the "neighborhood" of E also constitute a λ–spectral expander. (The precise definition of "neighborhood" will be discussed later.)*

Most known constructions of bounded-degree high–dimensional spectral expanders are heavily algebraic, rather than combinatorial or randomized. In contrast, there are a wealth of different constructions for bounded-degree (one-dimensional) expander graphs [10]. Some of these are also algebraic, such as the famous LPS construction of Ramanujan graphs [19], but there are also many simple, probabilistic constructions of expanders. In particular, Friedman's Theorem says that with high probability, random d-regular graphs are excellent expanders [8].

Unfortunately, random d–dimensional hypergraphs with low degrees are not d–dimensional expander graphs. For a hypergraph with n vertices, we need a roughly $n\left(\frac{\log n}{n}\right)^{1/d}$-degree Erdos-Renyi graph to make the neighborhood of every hyperedge of size $\leqslant d-1$ to be connected with high probability. While random low degree hypergraphs are not high–dimensional expanders, our construction provides simple probabilistic high–dimensional expanders of all dimensions.

1.1 Summary of our results

Construction

We construct an H–dimensional simplicial complex $\mathcal{Q}$ on $n \cdot s$ vertices, from a graph G of n vertices and a (small) H–dimensional complete simplicial complex $\mathcal{B}$ on s vertices. To construct $\mathcal{Q}$, we replace each vertex v of G with a copy of $\mathcal{B}$ which we denote $\mathcal{B}_v$. Denote the copy of a vertex $w \in \mathcal{B}$ in $\mathcal{B}_v$ by (v, w). The faces of $\mathcal{Q}$ are chosen in the following way: for every face $\{w_1, w_2, \ldots, w_k\}$ in $\mathcal{B}$, add it $\{(v_1, w_1), (v_2, w_2), \ldots, (v_k, w_k)\}$ to the complex, where for some edge e in G, the vertices $v_1, \ldots v_k$ are each one of the endpoints of e; in particular there are 2 choices for each v_i. The main punchline of our work is that when G is a (triangle-free) expander graph, the high order random walks on $\mathcal{Q}$ mix rapidly. Specifically, we prove:

▶ **Theorem 2** (Main theorem, informal version of Theorem 35). *Suppose G is a triangle-free expander graph with two-sided spectral gap ρ. For every k such that $1 \leqslant k < H$, there is a constant C depending on k, H, s, ρ, but independent of n such that the Markov transition matrix for the up-down walk on the k-faces of $\mathcal{Q}$ has two-sided spectral gap C.*

First attempt at proving rapid mixing of high order random walks

[13], which introduced the notions of up-down and down-up random walks, and subsequent works [6, 14, 15, 1] developed and followed the "local-to-global paradigm" to prove rapid mixing of high order random walks. In particular, each of these works would:

A. Establish that all the links of a relevant simplicial complex have "small" second eigenvalue.

B. Prove or cite a statement about how rapid mixing follows from small second eigenvalues of links (such as Theorem 4).

Then, step A and step B together would imply that the up-down and down-up random walks on the simplicial complexes they cared about mixed rapidly. This immediately motivates first bounding the second eigenvalue of the links of our construction, and applying the quantitatively strongest known version of the type of theorem alluded to in step B. Thus, in Section 4 we analyze the second eigenvalue of all links of $\mathcal{Q}$ and prove:

▶ **Theorem 3** (Informal version of Theorem 37). *The two-sided spectral gap of every link in $\mathcal{Q}$ is bounded by approximately $\frac{1}{2}$.*

And the "quantitatively strongest" known "local-to-global" theorem is

▶ **Theorem 4** (Informal statement of [14, Theorem 5]). *If the second eigenvalue of every link of a simplicial complex $\mathcal{S}$ is bounded by λ, then the up-down walk on k-faces of $\mathcal{S}$, $\mathcal{S}_k^{\uparrow\downarrow}$ satisfies:*

$$\lambda_2(\mathcal{S}_k^{\uparrow\downarrow}) \leqslant \left(1 - \frac{1}{k+1}\right) + k\lambda.$$

Observe that the upper bound on the second eigenvalue of all links must be strictly less than $\frac{1}{k(k+1)}$ to conclude any meaningful bounds on the mixing time of the up-down random walk. Thus, unfortunately, Theorem 3 in conjunction with Theorem 4 fails to establish rapid mixing.

Hence, we depart from the local-to-global paradigm and draw on alternate techniques.

Decomposing Markov chains

Each k-face of $\mathcal{Q}$ is either completely contained in a *cluster* $\{(v, ?)\}$ for a single vertex v in G, or straddles two clusters corresponding to vertices connected by an edge, i.e., is contained in $\{(v, ?)\} \cup \{(u, ?)\}$. Consider performing an up-down random walk on the space of k-faces of $\mathcal{Q}$ (henceforth $\mathcal{Q}_k^{\uparrow\downarrow}$). If we record the single cluster or pair of clusters containing the k-face the random walk visits at each timestep, it would resemble:

$$\{17, 19\} \to \{17, 19\} \to \{17, 19\} \to \{17\} \to \{17\} \to \{17, 155\} \to \{17, 155\} \to \{17, 155\} \to \{155\} \to \{155, 203\} \to \{155, 203\} \to \{155, 203\} \to \{155, 203\} \to \{155, 203\} \to \{203\} \to \{6, 203\} \to \{6, 203\} \to \cdots$$

In the above illustration of a random walk, let us restrict our attention to the segment of the walk where the k-faces are all contained in, say, the pair of clusters $\{155, 203\}$. Intuitively, we expect the random walk restricted to those k-faces to mix rapidly and also exit the set

quickly by virtue of the state space being constant-sized. In particular, if we keep the random walk running for $t \approx C \log n$ steps for some large constant C, it would seem that the number of "exit events"[1] is roughly $\alpha \cdot C \log n$ for some other constant α. The sequence of "exit events" can be viewed as a random walk on the space of edges and vertices of G, and since there are *many* steps in this walk, the expansion properties of G tell us that the location of the random walk after t steps is distributed according to a relevant stationary distribution. In light of these intuitive observations of rapidly mixing in the walks within cluster pairs and also rapidly mixing in a walk on the space of cluster pairs, one would hope that the up-down walk on k-faces mixes rapidly.

This hope is indeed fulfilled and is made concrete in a framework of Jerrum et al. [11]. In their framework, there is a Markov chain M on state space Ω. They show that if Ω can be partitioned into $\Omega_1, \ldots, \Omega_\ell$ such that the chain "restricted" (for some formal notion of restricted) to each Ω_i, and an appropriately defined "macro-chain" (where each partition Ω_i is a state) each have a constant spectral gap, then the original Markov chain M has a constant spectral gap as well. Our proof of the fact that $\mathcal{Q}_k^{\uparrow\downarrow}$ has a constant spectral gap utilizes this result of [11]. This framework of decomposable Markov chains is detailed in Section 2.2.1, and the analysis of the spectral gap of the down-up random walk[2] is in Section 5.

1.2 Related Work

While high–dimensional expanders have been of relatively recent interest, already many different (non-equivalent) notions of high–dimensional expansion have emerged, for a variety of different applications.

The earliest notions of high–dimensional expansion were topological. In this vein of work, [17, 9] introduced coboundary expansion, [7] defined cosystolic expansion, and [7, 12] defined skeleton expansion. To our knowledge, most existing constructions of these types of expanders rely on the Ramanujan complex. We refer the reader to a survey by Lubotzky [18] for more details on these alternate notions of high dimensional expansion and their uses.

To describe notions of high dimensional expansion that are relevant to computer scientists, we need to first highlight a key property of (one-dimensional) expander graphs–that random walks on them mix rapidly to their stationary distribution. The notion of a random walk on graphs was generalized to simplicial complexes in the work of Kaufman and Mass [13] to the "up-down" and "down-up" random walks, whose states are k-faces of a simplicial complex. They were interested in bounded–degree simplicial complexes where the up-down random walk mixed to its stationary distribution rapidly. They then proceed to show that the known construction of Ramanujan complexes from [20] indeed satisfy this property.

A key technical insight in their work that the rapid mixing of up-down random walks follows from certain notions of *local spectral expansion*, i.e., from sufficiently good *two-sided* spectral expansion of the underlying graph of every link. A quantitative improvement between the relationship between the two-sided spectral expansion of links and rapid mixing of random walks was made in [6], and this improvement was used to construct agreement expanders based on the Ramanujan complex construction. Later, [14] showed that one-sided spectral expansion of links actually sufficed to derive rapid mixing of the up-down walk on k-faces.

---

[1] Transitions like $\{17, 19\} \to \{17\}$, $\{155\} \to \{155, 203\}$, and so on.

[2] Which is actually equivalent to proving a spectral gap on the up-down random walk but is more technically convenient. See Fact 33.

1.2.1 HDX Constructions

Although this combinatorial characterization of high–dimensional expansion is slightly weaker than some of the topological characterizations mentioned above, few constructions are known for *bounded degree* HDX's with dimension $\geqslant 2$. Most of these rely on heavy algebra. In contrast, for one-dimensional expander graphs, there are a wealth of different constructions, including ones via graph products and randomized ones. [8] states that even a random d-regular graph is an expander with high probability.

The most well-known construction of bounded-degree high–dimensional expanders are the Ramanujan complexes [20]. These require the *Bruhat-Tits building*, which is a high-dimensional generalization of an infinite regular tree. The underlying graph has degree $q^{O(d^2)}$, where q is a prime power satisfying $q \equiv 1 \pmod 4$. The links can be described by *spherical buildings*, which are complexes derived from subspaces of a vector space, and are excellent expanders.

Dinur and Kaufman showed that given any $\lambda \in (0,1)$, and any dimension d, the d–skeleton of any $d + \lceil 2/\lambda \rceil$–dimensional Ramanujan complex is a d–dimensional λ–spectral expander [6]. Here, the degree of each vertex is $(2/\lambda)^{O((d+2/\lambda)^2)}$. In other words, they "truncate" the Ramanujan complexes, throwing out all faces of size greater than some number k. Their primary motivation was to obtain agreement expanders, which find uses towards PCPs.

Recently, Kaufman and Oppenheim [15] present a construction of one–sided high–dimensional expanders, which are coset complexes of elementary matrix groups. The construction guarantees that for any $\lambda \in (0,1)$ and any dimension d, there exists a infinite family of high–dimensional expanders $\{X_i\}_{i\in\mathbb{N}}$, such that (1) every X_i are d–dimensional λ–one–sided–expander; (2) every X_i's 1-skeleton has degree at most $\Theta\left(\sqrt{\frac{(1/\lambda+d-1)^{(d+2)^2}}{2\log(1/\lambda+d-1)}}\right)$; (3) as i goes to infinity the number of vertices in X_i also goes to infinity.

Even more recently, Chapman, Linial, and Peled [5] also provided a combinatorial construction of two-dimensional expanders. They construct an infinite family of (a,b)-regular graphs, which are a-regular graphs whose links with respect to single vertices are b-regular. The primary motivation for their construction comes from the theory of PCPs. They prove an Alon-Boppana type bound on $\lambda_2(G)$ for any (a,b)-regular graph, and construct a family of graphs where this bound is tight. They also build an (a,b)-regular two-dimensional expander using any non-bipartite graph G of sufficiently high girth; they achieve a local expansion only depending on the girth, and the global expansion depending on the spectral gap of G. Like ours, their construction also resembles existing graph product constructions of one-dimensional expanders.

2 Preliminaries and Notation

2.1 Spectral Graph Theory

While we can describe our constructions combinatorially, our analysis of both the mixing times of certain walks as well as the local expansion will heavily rely on understanding graph spectra.

▶ **Definition 5.** *For an edge-weighted directed graph G on n vertices, we use* $\mathrm{Adj}(G)$ *to denote its (normalized) adjacency matrix, i.e. the matrix given by*

$$\mathrm{Adj}(G)_{(u,v)} = \frac{\mathbf{1}_{(u,v)\in E(G)} \cdot w((u,v))}{\sum_{v:(u,v)\in E(G)} w((u,v))}$$

and write its (right) eigenvalues as

$$1 = \lambda_1(G) \geqslant \lambda_2(G) \geqslant \ldots \geqslant \lambda_n(G) \geqslant -1.$$

Let $\mathsf{Spectrum}(G)$ *to indicate the set* $\{\lambda_i(G)\}$*. We write* $\mathsf{OneSidedGap}(G)$ *for the spectral gap of* G*, which is the quantity* $1 - \lambda_2(G)$*. Graphs with* $\mathsf{OneSidedGap}(G) \geqslant \mu$ *are one-sided* μ*-expanders.*

Most of the graphs we analyze achieve a stronger condition; that the second largest eigenvalue magnitude is not too large. Formally, we write $|\lambda|_i$ *for the* i*-th largest eigenvalue in absolute value. In particular,* $|\lambda|_2 = \max\{|\lambda_2|, |\lambda_n|\}$*. The absolute spectral gap of* G*, denoted* $\mathsf{TwoSidedGap}(G)$*, is the quantity* $1 - |\lambda|_2$*. Graphs with* $\mathsf{TwoSidedGap}(G) \geqslant \mu$ *are two-sided* μ*-expanders.*

▶ Remark 6. For an *undirected* weighted graph, we simply have $w((u, v)) = w((v, u))$, and use this to define the adjacency matrix the same way.

2.1.1 Graph Tensors

Our construction can roughly be described as a tensor product, defined below.

▶ **Definition 7.** *The tensor product* $G \times H$ *of two graphs* G *and* H *is given by*
1. *Vertex set* $V(G \times H) = V(G) \times V(H)$*,*
2. *Edge set* $E(G \times H) = \{((u_1, v_1), (u_2, v_2)) : (u_1, u_2) \in E(G)$ *and* $(v_1, v_2) \in E(H)\}$*.*

The adjacency matrix $\mathrm{Adj}(G \times H)$ is the tensor (Kronecker) product $\mathrm{Adj}(G) \otimes \mathrm{Adj}(H)$. Due to this structure, $\mathsf{Spectrum}(G \times H) = \{\lambda_i \mu_j : \lambda_i \in \mathsf{Spectrum}G,\ \mu_j \in \mathsf{Spectrum}(H)\}$. As 1 is the largest eigenvalue of both $\mathrm{Adj}(G)$ and $\mathrm{Adj}(H)$, it follows that both

$$\mathsf{OneSidedGap}(G \times H) = \min(1 - 1 \cdot \mu_2, 1 - \lambda_2 \cdot 1) = \min(\mathsf{OneSidedGap}(G), \mathsf{OneSidedGap}(H))$$
$$\mathsf{TwoSidedGap}(G \times H) = \min(1 - 1 \cdot |\mu|_2, 1 - |\lambda|_2 \cdot 1) = \min(\mathsf{TwoSidedGap}(G), \mathsf{TwoSidedGap}(H))$$

2.2 Markov Chains

We provide a basic overview of the Markov chain concepts used to analyze our high order walks. We refer to [16] for a detailed and thorough treatment of the fundamentals of Markov chains.

▶ **Definition 8.** *A Markov chain* $M = (\Omega, P)$ *is given by states* Ω *and a transition matrix* P *where* $P[i, j]$ *is the probability of going to state* j *from state* i*. We may also write this quantity as* $M[j \to i]$*.*

▶ Remark 9. The literature often defines $P_{i,j}$ as the probability $\mathbf{Pr}(i \to j)$, so their P is the transpose of ours. However, we work with column (right) eigenvectors to analyze the spectrum of P, while this alternate convention uses row (left) eigenvectors, so both conventions yield the same results.

▶ **Definition 10.** *We can view any Markov chain* M *as a weighted, directed graph* G*, defined by* $V(G) = \mathsf{States}(M)$*,* $E(G) \coloneqq \{(i, j) : i, j \in V(G), M[i \to j] > 0\}$*, and* $w((i, j)) = M[j \to i]$*.*

The transition matrix of M *is* $\mathrm{Adj}(G)$*, and we also refer to* $\mathsf{Spectrum}(G)$ *as the spectrum of* M*. Every adjacency matrix has* $\lambda_1 = 1$*, so transition matrix of* M *has an eigenvector* π_M *(normalized so that entries sum to* 1*) for the eigenvalue* 1*. We call* π_M *a stationary distribution of* M*.*

▶ Remark 11. We may use the term "graph" in lieu of "chain" when we want to indicate the random walk defined by the transition matrix $\mathrm{Adj}(G)$.

The next property we introduce is present for every Markov chain we analyze.

▶ **Definition 12.** *The Markov chain $M = (\Omega, P)$ is time-reversible if for any integer $k \geqslant 1$:*

$$\pi_M(x_0)M[x_0 \to x_1] \cdots M[x_{k-1} \to x_k] = \pi_M(x_k)M[x_k \to x_{k-1}] \cdots M[x_1 \to x_0]$$

Intuitively, it means that if start at the stationary distribution and run the chain for a sequence of time states, the reverse sequence has the same probability of occurring. Time reversibility helps us compute stationary distributions via the *detailed balance equations*. (This is especially helpful when there are a huge number of symmetric states.)

▶ **Fact 13.** *The Markov chain $M = (\Omega, P)$ is time-reversible if and only if it satisfies the detailed balance equations: for all $x, y \in \Omega$,*

$$\pi_M(x)M[x \to y] = \pi_M(y)M[y \to x]$$

▶ **Definition 14.** *The ε-mixing time of a Markov chain M is the smallest t such that for any distribution ν over the states of M,*

$$\|\pi_M - P^t\nu\|_1 \leqslant \varepsilon$$

where π_M is the stationary distribution of M.

▶ **Theorem 15.** *For any Markov chain M, the ε-mixing time $t_{\text{mix}}(\varepsilon)$ satisfies:*

$$t_{\text{mix}}(\varepsilon) \leqslant \log\left(\frac{1}{\varepsilon \min \pi_M}\right) \cdot \frac{1}{\mathsf{TwoSidedGap}(M)}.$$

2.2.1 Decomposing Markov Chains

Consider a finite-state time reversible Markov chain M whose structure gives rise to natural state-space partition, M can be decomposed into a number of restriction chains and a projection chain. [11] show that the spectral gap for the original chain can be lower bounded in terms of the spectral gaps for the restriction and projection chains.

We now formally define the decomposition of a Markov chain. Consider an ergodic Markov chain on finite state space Ω with transition probability $P\colon \Omega^2 \to [0,1]$. Let $\pi\colon \Omega \to [0,1]$ denote its stationary distribution, and let $\{\Omega_i\}_{i\in[m]}$ be a partition of the state space into m disjoint sets, where $[m] \coloneqq \{1, \ldots, m\}$.

The *projection chain* induced by the partition $\{\Omega_i\}$ has state space $[m]$ and transitions

$$\overline{P}(i,j) = \left(\sum_{x\in\Omega_i} \pi(x)\right)^{-1} \sum_{x\in\Omega_i y\in\Omega_j} \pi(x)P(x,y).$$

The above expression corresponds to the probability of moving from any state in Ω_i to any state in Ω_j in the original Markov chain.

For each $i \in [m]$, the *restriction chain* induced by Ω_i has state space Ω_i and transitions

$$P_i(x,y) = \begin{cases} P(x,y), & x \neq y, \\ 1 - \sum_{z\in\Omega_i\setminus\{x\}} P(x,z), & x = y. \end{cases}$$

$P_i(x,y)$ is the probability of moving from state $x \in \Omega_i$ to state y when leaving Ω_i is not allowed.

Regardless of how we define the projection and restriction chains for a time reversible Markov chain, they all inherit one useful property from the original chain.

▶ **Fact 16.** *Let $M = (\Omega, P)$ be a time-reversible Markov chain. Then, for any decomposition of M, the projection and restriction chains are also time-reversible.*

We ultimately want to study the spectral gap of random walks. Luckily, the original Markov chain's spectral gap is related to the restriction and projection chains' spectral gaps in the following way:

▶ **Theorem 17** ([11, Theorem 1])**.** *Consider a finite-state time-reversible Markov chain decomposed into a projection chain and m restriction chains as above. Define γ to be maximum probability in the Markov chain that some state leaves its partition block,*

$$\gamma \coloneqq \max_{i\in[m]} \max_{x\in\Omega_i} \sum_{y\in\Omega\setminus\Omega_i} P(x,y).$$

Suppose the projection chain satisfies a Poincaré inequality with constant $\bar{\lambda}$, and the restriction chains satisfy inequalities with uniform constant $\lambda_{\min}$. Then the original Markov chain satisfies a Poincaré inequality with constant

$$\lambda \coloneqq \min\left\{\frac{\bar{\lambda}}{3}, \frac{\bar{\lambda}\lambda_{\min}}{3\gamma+\bar{\lambda}}\right\}.$$

Recall that if λ satisfies a Poincaré inequality, it is a lower bound on the spectral gap (cf. [16]).

2.3 High-Dimensional Expanders

The generalization from expander graphs to hypergraphs (more specifically, simplicial complexes) requires great care. We now formally establish the high dimensional notions of "neighborhood", "expansion," and "random walk."

▶ **Definition 18.** *A simplicial complex $\mathcal{S}$ is specified by vertex set $V(\mathcal{S})$ and a collection $\mathcal{F}(\mathcal{S})$ of subsets of $V(\mathcal{S})$, known as faces, that satisfy the "downward closure" property: if $A \in \mathcal{F}(\mathcal{S})$ and $B \subseteq A$, then $B \in \mathcal{F}(\mathcal{S})$. Any face $S \in \mathcal{F}(\mathcal{S})$ of cardinality $(k+1)$ is called a k-face of $\mathcal{S}$. We use k-$\mathsf{faces}(\mathcal{S})$ to denote the subcollection of k-faces of $\mathcal{F}(\mathcal{S})$. We say $\mathcal{S}$ has dimension d, where $d = \max\{|F| : F \in \mathcal{F}(\mathcal{S})\} - 1$.*

▶ **Example 19.** A 1-dimensional complex $\mathcal{S}$ is a graph with vertex set $V(\mathcal{S})$ and edge set 1-$\mathsf{faces}(\mathcal{S})$.

▶ **Definition 20.** *To formally define random walks and Markov chains on a $\mathcal{S}$, we need to associate $\mathcal{S}$ with a weight function $w : \mathcal{F}(\mathcal{S}) \to \mathbb{R}_+$. We want our weight function to be balanced, meaning for $F \in k$-$\mathsf{faces}(\mathcal{S})$:*

$$w(F) = \sum_{J\in(k+1)\text{-}\mathsf{faces}(\mathcal{S}):J\supset F} w(J)$$

If we restrict ourselves to balanced w, it suffices to only define w over d-$\mathsf{faces}(\mathcal{S})$ and propagate the weights downward to the lower order faces.

▶ **Definition 21.** *The (weighted) k-skeleton of $\mathcal{S}$ is the complex with vertex set $V(\mathcal{S})$ and all faces in $\mathcal{F}(\mathcal{S})$ of cardinality at most $k+1$, with weights inherited from $\mathcal{S}$.*

▶ **Example 22.** The 1-skeleton of $\mathcal{S}$ only contains its vertices (0-faces) and edges (1-faces). It can be characterized as a graph with edge weights, so we can also compute $\mathsf{OneSidedGap}(1\text{-}\mathsf{skeleton}(\mathcal{S}))$ and $\mathsf{TwoSidedGap}(1\text{-}\mathsf{skeleton}(\mathcal{S}))$.

▶ **Definition 23.** *For $S \in k\text{-faces}(\mathcal{S})$ for $k \leqslant H-1$, we associate a particular $(H-k)$-dimensional complex known as the link of S defined below.*

$$\mathsf{link}(S) := \{T \setminus S : T \in \mathcal{F}(\mathcal{S}), S \subseteq T\}$$

If $\mathcal{S}$ was equipped with weight function w, then $\mathsf{link}(S)$ "inherits" it. We associate $\mathsf{link}(S)$ with weight function w_S given by $w_S(T) = w(S \cup T)$. If w is balanced, then w_S is also balanced. We call a $\mathsf{link}(S)$ a t-link if $|S|$ has cardinality t.

▶ **Example 24.** In a graph, the link of a vertex is simply its neighborhood.

▶ **Definition 25.** *The global expansion of $\mathcal{S}$, denoted $\mathsf{GlobalExp}(\mathcal{S})$, is the expansion of its weighted 1-skeleton.*

▶ **Definition 26.** *The local expansion of $\mathcal{S}$, denoted $\mathsf{LocalExp}(\mathcal{S})$ is*

$$\mathsf{LocalExp}(\mathcal{S}) := \min_{0 \leqslant k \leqslant H-1} \min_{S \in k\text{-faces}(\mathcal{S})} \mathsf{TwoSidedGap}(1\text{-skeleton}(\mathsf{link}(S))).$$

In words, it is equal to the expansion of the worst expanding link.

▶ **Example 27.** We use $\mathcal{K}^{(H)}_{H+2}$ to denote the *complete H-dimensional complex* on vertex set $[H+2]$, i.e., the pure H-dimensional simplicial complex obtained by making the set of $(H+1)$-faces equal to all subsets of $[H+2]$ of size $H+1$. The 1-skeleton is then a clique on $H+2$ vertices whose expansion is $1-\frac{1}{H+1}$ and the 1-skeleton of a t-link is a clique on $H+2-t$ vertices, which has expansion $1-\frac{1}{H+1-(t-1)}$. As a result, $\mathsf{TwoSidedGap}\left(\mathcal{K}^{(H)}_{H+2}\right) = \frac{1}{2}$.

▶ Remark 28. We often use $\mathrm{Adj}(\mathcal{S})$ to refer to the adjacency matrix of the 1-skeleton of $\mathcal{S}$, and we may also use $\lambda_i(\mathcal{S})$ to refer to the i-th largest eigenvalue of $\mathrm{Adj}(\mathcal{S})$.

Previously, we mentioned that there are several different notions of high dimensional expansion: some geometric or topological, some combinatorial. We now formally define high dimensional spectral expansion, which is a more combinatorial and graph theoretic notion:

▶ **Definition 29.** *$\mathcal{S}$ is a two-sided λ-local spectral expander if $\mathsf{GlobalExp}(\mathcal{S}) \geqslant \lambda$ and $\mathsf{LocalExp}(\mathcal{S}) \geqslant \lambda$.*

2.3.1 High Order Walks on Simplicial Complexes

Let $\mathcal{S}$ be a H-dimensional simplicial complex and with weight function $w : k\text{-faces}(\mathcal{S}) \to \mathbb{R}_{\geqslant 0}$ on the k-faces of $\mathcal{S}$, for $k \leqslant H$. For each $k < H$, we can define a natural (periodic) Markov chain on a state space consisting of k-faces and $(k+1)$-faces of $\mathcal{S}$.

- At a $(k+1)$-face J, there are exactly $(k+2)$ faces $F \in k\text{-faces}(\mathcal{S})$ such that $F \subset J$, due to the downward closure property. We transition from J to each k-face F with probability $\frac{1}{k+2}$.
- At a k-face F, we transition to each $(k+1)$-face J satisfying $J \supset F$ with probability $\frac{w(J)}{w(F)}$. (Note that w must be balanced for these transitions to be well-defined.)

Restricting the above chain to only odd or even time steps gives us two new random walks: one entirely on $k\text{-faces}(\mathcal{S})$ and one entirely on $(k+1)\text{-faces}(\mathcal{S})$.

▶ **Definition 30** (Down-up walk on k-faces of $\mathcal{S}$). *Let $\mathcal{S}^{\downarrow\uparrow}_{k+1}$ be the Markov chain with state space equal to k-faces$(\mathcal{S})$ and transition probabilities $\mathcal{S}_{\downarrow\uparrow}[J \to J']$ described by the process above, where there is an implicit transition down to a k-face and back up to a $(k+1)$-face. Then:*

$$\mathcal{S}_{\downarrow\uparrow}[J \to J'] = \begin{cases} \frac{1}{k+1} \sum_{F \in k\text{-faces}(\mathcal{S}): F \subset J} \frac{w(J)}{w(F)} & \text{if } J = J' \\ \frac{1}{k+1} \cdot \frac{w(J')}{w(J \cap J')} & \text{if } J \cap J' \in k\text{-faces}(\mathcal{S}) \\ 0 & \text{otherwise} \end{cases}$$

▶ **Definition 31** (Up-down walk on k-faces of $\mathcal{S}$). *Let $\mathcal{S}_{\uparrow\downarrow}$ be the Markov chain with state space equal to k-faces$(\mathcal{S})$ and transition probabilities $\mathcal{S}_{\uparrow\downarrow}[F \to F']$ described by the process above, where there is an implicit transition up to a $(k+1)$-face and back down to a k-face. Then:*

$$\mathcal{S}_{\downarrow\uparrow}[F \to F'] = \begin{cases} \frac{1}{k+1} & \text{if } F = F' \\ \frac{w(F \cup F')}{w(F)} & \text{if } F \cup F' \in (k+1)\text{-faces}(\mathcal{S}) \\ 0 & \text{otherwise} \end{cases}$$

▶ Remark 32. In the literature, we also see $\mathcal{S}^{\downarrow\uparrow}_{k+1}$ written as $\mathcal{S}^{\vee}_{k+1}$, and $\mathcal{S}^{\uparrow\downarrow}_{k}$ written as $\mathcal{S}^{\wedge}_{k}$.

We now present some facts about these high order walks without proof. We refer to [14, 1] for proofs of these facts.

▶ **Fact 33.** *The transition matrices for $\mathcal{S}^{\downarrow\uparrow}_{k+1}$ and $\mathcal{S}^{\uparrow\downarrow}_{k}$ share the same eigenvalues. The nonzero eigenvalues occur with the same multiplicity. A straightforward but important consequence of this fact is*

$$\text{Spectrum}(\mathcal{S}^{\downarrow\uparrow}_{k+1}) = \text{Spectrum}(\mathcal{S}^{\uparrow\downarrow}_{k+1})$$

▶ **Fact 34.** *The Markov chains $\mathcal{S}^{\downarrow\uparrow}_{k}$ and $\mathcal{S}^{\uparrow\downarrow}_{k}$ have the same stationary distribution on k-faces$(\mathcal{S})$, which is proportional to $w(F)$ for each $F \in k$-faces$(\mathcal{S})$. We will call this distribution $\pi_k(\cdot)$.*

For the remainder of the paper, we will assume a uniform weight function on d-faces$(\mathcal{S})$, which is useful for applications like sampling bases of a matroid [1]. When using the uniform weighting scheme, for $F \in k$-faces$(\mathcal{S})$, there is a natural interpretation of $\pi_k(F)$: the fraction of d-faces that contain F as a subface. (We also note that we will use symbolic variables to represent various weight values, and that it is straightforward to adapt our computations to cases where we have uniform weights over k-faces$(\mathcal{S})$ for any k.)

3 Local Densification of Expanders

For a graph G and H-dimensional simplicial complex $\mathcal{S}$, we give a way to combine the two to produce a bounded-degree H-dimensional complex LocalDensifier$(G, \mathcal{S})$ of constant expansion. First, construct a graph G' with

1. vertex set equal to $V(G) \times V(\mathcal{S})$, and
2. edge set equal to $\{\{(v_1, b_1), (v_2, b_2)\} : \{b_1, b_2\} \in 1\text{-faces}(\mathcal{S}), \{v_1, v_2\} \in E(G) \text{ or } v_1 = v_2\}$.

$\mathsf{LocalDensifier}(G, \mathcal{S})$ is then defined as the H-dimensional pure complex whose H-faces are all cliques on $H+1$ vertices $\{(v_1, b_1), (v_2, b_2), \ldots, (v_{H+1}, b_{H+1})\}$ such that there exists an edge $\{a, b\}$ in G for which $v_1, \ldots, v_{H+1} \in \{a, b\}$.

To describe a k-face of $\mathsf{LocalDensifier}(G, \mathcal{S})$, we may also use the ordered pair (F, f), where F is a k-face of $\mathcal{S}$, and f is a function mapping each element of F to a vertex of G. Because of the local densifier's tensor structure, $\mathsf{image}(f)$ is either a single vertex, or a pair of vertices that form an edge in G.

Linear algebraically, we can think of this graph construction as adding a self loop to each vertex of G and then taking the tensor product with the 1-skeleton of $\mathcal{S}$.

Our construction is $\mathcal{Q} := \mathsf{LocalDensifier}(G, \mathcal{B})$, where $\mathcal{B}$ is equal to $\mathcal{K}_s^{(H)}$, the H-dimensional complete complex on some constant $s \geqslant H+1$ vertices, and G is a T-regular triangle-free expander graph on n vertices. We endow $\mathcal{Q}$ with a balanced weight function w induced by setting the weights of all H-faces to 1.

As a first step to understanding this construction, we inspect the weights induced on k-faces for $k < H$. Consider a k-face $F := \{(v_1, b_1), \ldots, (v_{k+1}, b_{k+1})\}$. A short calculation reveals that if $v_1, \ldots, v_{k+1}$ are all equal, then $w(F)$ is equal to $w_{J,k} := \binom{s}{H-k} \cdot [T2^{H-k} - (T-1)]$ and otherwise, $w(F)$ is equal to $w_{I,k} := \binom{s}{H-k} \cdot [2^{H-k}]$. Henceforth, write w_J and w_I instead of $w_{J,k}$ and $w_{I,k}$ when k is understood from context.

We now list out what we prove about $\mathcal{Q}$. Most importantly, we show:

▶ **Theorem 35.** *For every $1 \leqslant k < H$, the Markov transition matrix $\mathcal{Q}_k^{\downarrow\uparrow}$ for down-up (and equivalently up-down) random walks on the k-faces satisfies:*

$$\mathsf{TwoSidedGap}\left(\mathcal{Q}_k^{\downarrow\uparrow}\right) \geqslant \frac{\mathsf{TwoSidedGap}(G)}{64T^2(k+1)^2(s-k)(2^k-1)} \ .$$

We dedicate Section 5 to proving Theorem 35.

As an immediate corollary of Theorem 35 and Theorem 15, we get that

▶ **Corollary 36.** *Let N_k denote the number of k-faces in $\mathcal{Q}$. Then the ϵ-mixing time of $\mathcal{Q}_k^{\downarrow\uparrow}$ satisfies:*

$$t(\varepsilon) \leqslant \frac{64T^2(k+1)^2(s-k)(2^k-1)}{\mathsf{TwoSidedGap}(G)} \cdot \log\left(\frac{2N_k}{\varepsilon}\right) \ .$$

We note that $N_k = \Theta(n)$.

We also derive bounds on the expansion of links of $\mathcal{Q}$. In particular, as a direct consequence of Theorem 41 and the discussion of the expansion properties of the complete complex in Example 27, we conclude:

▶ **Theorem 37.** *We can prove the following bounds on the local and global expansion of $\mathcal{Q}$:*

$$\mathsf{GlobalExp}(\mathcal{Q}) \geqslant \left[\frac{1}{2} - \frac{1}{2 \cdot (T2^H + 1)}\right] \cdot \mathsf{TwoSidedGap}(G), \text{ and}$$
$$\mathsf{LocalExp}(\mathcal{Q}) \geqslant \frac{1}{2}.$$

▶ Remark 38. Suppose $\boldsymbol{G}$ is a random T-regular (triangle-free) graph and $H \geqslant T$. Then the corresponding (random) simplicial complex $\boldsymbol{\mathcal{Q}}$, as a consequence of Friedman's Theorem [8][3], with high probability satisfies

$$\begin{aligned}\mathsf{TwoSidedGap}\left(\boldsymbol{\mathcal{Q}}_k^{\downarrow\uparrow}\right) &\geqslant \frac{T-2\sqrt{T-1}-o_n(1)}{64T^3(k+1)^2(s-k)(2^k-1)}\\ \mathsf{GlobalExp}(\mathcal{S}_{\boldsymbol{\mathcal{Q}}}) &\geqslant \frac{T-2\sqrt{T-1}-o_n(1)}{T+1}, \text{ and}\\ \mathsf{LocalExp}(\mathcal{S}) &\geqslant 1/2.\end{aligned}$$

Thus, $\boldsymbol{\mathcal{Q}}$ endows a natural distribution over simplicial complexes that gives a high-dimensional expander with high probability.

▶ Remark 39. If G is *strongly explicit*, such as an expander from [21, 3], then $\mathcal{Q}$ is also strongly explicit since the tensor product of two strongly explicit graphs is also strongly explicit.

4 Local Expansion

For this entire section, we will mainly work with the complex $\mathsf{LocalDensifier}(G,\mathcal{S})$, so when we use $\mathsf{link}(\cdot)$ without a subscript, it will be with respect to $\mathsf{LocalDensifier}(G,\mathcal{S})$. Next, fix a face $\sigma = (F,f) \in k\text{-}\mathsf{faces}(\mathsf{LocalDensifier}(G,\mathcal{S}))$. In order to study the expansion of the 1-skeleton of $\mathsf{link}(\sigma)$, we need to first compute the weights on its 1-faces.

Let $\tau = \{(v_1,b_1),(v_2,b_2)\} \in 2\text{-}\mathsf{faces}(\mathsf{link}(\sigma))$, where as before, $v_i \in V(G)$ and $b_i \in 1\text{-}\mathsf{faces}(\mathcal{S})$. There are several cases we need to consider:

1. Case 1: $|\mathsf{image}(f)| = 2$.
   Here, $w_\sigma(\tau) = w(\tau \cup \sigma)$, which is proportional to the number of H-faces (F',f') that contain $\tau \cup \sigma$. The face $\tau \cup \sigma$ already has $(k+3)$ vertices, so there are $\binom{S}{H-(k+2)}$ possibilities of F'. There are $2^{H-(k+2)}$ choices for f', since $\mathsf{image}(f')$ must equal $\mathsf{image}(f)$.
2. Case 2: $|\mathsf{image}(f)| = 1$.
   a. Case 2(a): $v_1 = v_2 \in \mathsf{image}(f)$ and $\{b_1,b_2\} \in 2\text{-}\mathsf{faces}(\mathsf{link}_{\mathcal{S}}(F))$.
      Again, there are $\binom{S}{H-(k+2)}$ possibilities for F'. Since $v_1 = v_2 \in \mathsf{image}(f)$, we will have $T \cdot [2^{H-(k+2)} - 1] + 1$ choices for f', as v_1 has T neighbors in G, and when f' is not constant on v_1, there are T choices for the other value it can take.
   b. Case 2(b): $v_1 \neq v_2$ but $(v_1,v_2) \in E(G)$, and $\{b_1,b_2\} \in 2\text{-}\mathsf{faces}(\mathsf{link}_{\mathcal{S}}(F))$.
      Again, we have $\binom{S}{H-(k+2)}$ possibilities for F, but we only have $2^{H-(k+2)}$ choices for f'; the image of f' must be $\{v_1,v_2\}$.
   c. Case 2(c): $v_1 = v_2 \notin \mathsf{image}(f)$ but $v_1 \cup \mathsf{image}(f) \in E(G)$, and $\{b_1,b_2\} \in 2\text{-}\mathsf{faces}(\mathsf{link}_{\mathcal{S}}(F))$.
      The analysis is identical to that of Case 2(b)

For simplicity, we'll assign weights to the elements of $2\text{-}\mathsf{faces}(\mathsf{LocalDensifier}(G,\mathcal{S}))$ as below:

$$w(\{(v_1,b_1),(v_2,b_2)\}) = \begin{cases} w_{S,k} := 2^{H-(k+2)} & \text{for Case 1, 2(b), and 2(c)} \\ w_{C,k} := 1 + T(2^{H-(k+2)} - 1) & \text{for Case 2(a)} \end{cases}$$

(Here, the C and S denote "center" and "satellite," whose meanings will be more natural when discussing $\mathsf{link}(\sigma)$ when $\sigma \neq \emptyset$.)

[3] Friedman's theorem says that a random T-regular graph, whp, has two-sided spectral gap $\frac{T-2\sqrt{T-1}-o_n(1)}{T}$. Additionally, random graphs are triangle-free with constant probability.

▶ Remark 40. Note that if we choose $\sigma = \emptyset$ (so $k = -1$), we simply get the weights of the 1-skeleton of $\mathsf{LocalDensifier}(G, \mathcal{S})$ itself, which will be useful for computing global expansion.

▶ **Theorem 41.** *Let G be a triangle-free T-regular graph and let $\mathcal{S}$ be a pure H-dimensional simplicial complex. Then*

$$\mathsf{GlobalExp}(\mathsf{LocalDensifier}(G, \mathcal{S})) = \min\left\{\frac{T2^{H-1}}{T2^H - (T-1)} \cdot \mathsf{TwoSidedGap}(G), \mathsf{GlobalExp}(\mathcal{S})\right\}, \text{ and}$$

$$\mathsf{LocalExp}(\mathsf{LocalDensifier}(G, \mathcal{S})) = \mathsf{TwoSidedGap}(\mathcal{S}).$$

Proof. Let $\widetilde{G}$ be the graph obtained by adding self-loops to G, with transitions

$$\widetilde{G}[i \to j] = \begin{cases} \frac{w_{C,-1}}{w_{C,-1}+Tw_{S,-1}} & \text{if } i = j \\ \frac{w_{S,-1}}{w_{C,-1}+Tw_{S,-1}} & \text{otherwise} \end{cases}$$

For large H, the self loop probabilities approach $\frac{1}{2}$, while the others approach $\frac{1}{2T}$.

First, observe that $\mathrm{Adj}(\mathsf{LocalDensifier}(G, \mathcal{S})) = \mathrm{Adj}(\widetilde{G}) \otimes \mathrm{Adj}(\mathcal{S})$. Thus,

$$\mathsf{Spectrum}(\mathsf{LocalDensifier}(G, \mathcal{S})) = \{\lambda\mu : \lambda \in \mathsf{Spectrum}(\widetilde{G}), \mu \in \mathsf{Spectrum}(\mathcal{S})\}.$$

and hence the second largest absolute eigenvalue is no more than $\max\{\lambda_1(\widetilde{G})|\lambda|_2(\mathcal{S}), \lambda_1(\mathcal{S})|\lambda|_2(\widetilde{G})\}$, which is simply equal to $\max\{|\lambda|_2(\widetilde{G}), |\lambda|_2(\mathcal{S})\}$. This implies that

$$\mathsf{GlobalExp}(\mathsf{LocalDensifier}(G, \mathcal{S})) = \min\{\mathsf{TwoSidedGap}(\widetilde{G}), \mathsf{GlobalExp}(\mathcal{S})\}.$$

By Lemma 59,

$$\begin{aligned}\mathsf{TwoSidedGap}(\widetilde{G}) &= (1 - \frac{w_{C,-1}}{w_{C,-1} + Tw_{S,-1}}) \cdot \mathsf{TwoSidedGap}(G) \\ &= \frac{T2^{H-1}}{T2^H - (T-1)} \cdot \mathsf{TwoSidedGap}(G)\end{aligned}$$

the first part of the theorem statement follows.

Next, we lower bound $\mathsf{LocalExp}(\mathsf{LocalDensifier}(G, \mathcal{S}))$. For any face S in $\mathsf{LocalDensifier}(G, \mathcal{S})$, there exists an edge $\{u, v\}$ in G such that S is contained in $\{u, v\} \times S'$ where S' is a face of $\mathcal{S}$. If S contains vertices from both $\{u\} \times S'$ and $\{v\} \times S'$, then $\mathsf{link}(S)$ is isomorphic to $\mathsf{LocalDensifier}(\mathsf{edge}, \mathsf{link}(S'))$ where edge denotes a single-edge graph.

$$\mathsf{Spectrum}(\mathsf{LocalDensifier}(\mathsf{edge}, \mathsf{link}(S'))) = \{0\} \cup \mathsf{Spectrum}(\mathsf{link}(S'))$$

and hence

$$\mathsf{TwoSidedGap}(1\text{-}\mathsf{skeleton}(\mathsf{link}(S))) = \mathsf{TwoSidedGap}(\mathcal{S}).$$

Without loss of generality, the remaining case is if S contains vertices from only $\{u\} \times S'$. In this case, $\mathsf{link}(S)$ is isomorphic to $\mathsf{LocalDensifier}(\mathsf{star}, \mathsf{link}(S'))$ where star denotes a star graph with T satellites.

$$\mathsf{Spectrum}(\mathsf{LocalDensifier}(\mathsf{star}, \mathsf{link}(S'))) = \{\lambda\mu : \lambda \in \mathsf{Spectrum}(\mathsf{link}(S')), \mu \in \mathsf{Spectrum}(M)\} \quad (1)$$

where M is star with self loops added on each vertex. We'll call the center vertex of M the "center" vertex, and we'll call the remaining vertices the "satellites."

Using $w_{C,k}$ and $w_{S,k}$ for Cases 2(a), 2(b), and 2(c) computed above, we can also find the appropriate weights for M.

$$M[i \to j] = \begin{cases} \frac{w_{C,k}}{w_{C,k} + T w_{S,k}} & \text{if } i = j,\ i \text{ is the center} \\ \frac{w_{S,k}}{w_{C,k} + T w_{S,k}} & \text{if } i \text{ is the center vertex, } j \text{ is a satellite} \\ \frac{1}{2} & \text{if } i \text{ is a satellite} \end{cases}$$

We can completely classify the eigenspaces of $\mathrm{Adj}(M)$ and determine their corresponding eigenvalues as follows.

1. The vector with value $\frac{w_{C,k}+Tw_{S,k}}{2w_{S,k}}$ on the center of the star and 1 on satellites is an eigenvector of $\mathrm{Adj}(M)$ with eigenvalue 1.
2. The $(T-1)$-dimensional subspace of vectors which are 0 on the center of the star, and whose entries sum to 0 is an eigenspace for eigenvalue $1/2$.
3. The vector with value $-T$ on the center and 1 on the satellites is an eigenvector with eigenvalue $\frac{1}{2} - \frac{w_{C,k}}{w_{C,k}+Tw_{S,k}}$. For large H, this eigenvalue approaches 0.

Since the above classification gives $T+1$ eigenvectors it is complete and it is clear that the second largest absolute eigenvalue of M_2 is bounded by $1/2$ and thus in this case as well, using (1), we can infer

$$\mathsf{TwoSidedGap}(\text{1-skeleton}(\mathsf{link}(S))) \geqslant \min\{\mathsf{TwoSidedGap}(\mathcal{S}), 1/2\}.$$

which means

$$\mathsf{LocalExp}(\mathsf{LocalDensifier}(G, \mathcal{S})) \geqslant \min\{\mathsf{TwoSidedGap}(\mathcal{S}), 1/2\}. \qquad \blacktriangleleft$$

5 Spectral Gap of High Order Walks

5.1 Offsets and Colors

We now inspect the structure of the k-faces of our construction $\mathcal{Q}$ in more detail.

▶ **Definition 42** (k-faces of $\mathcal{Q}$)**.** *The set of k-faces of $\mathcal{Q}$ is exactly equal to the set of tuples (F, f) where F is a k-face of $\mathcal{B}$ and f is a labeling of each element by endpoints of some edge $\{u, v\}$ in G. We call (F, f) t-offset if either $|\{x \in F : f(x) = u\}| = t$ or $|\{x \in F : f(x) = v\}| = t$.*

▶ Remark 43. Suppose $t \leqslant k+1-t$. Note that a $(k+1-t)$-offset state is also t-offset, but we will stick to the convention of describing such states as t-offset. For example, a $(k+1)$-offset state is also 0-offset, but we will only use the term 0-offset.

▶ **Definition 44** (Coloring of k-faces of $\mathcal{Q}$)**.** *We color a k-face (F, f) of $\mathcal{Q}$ with $\mathsf{image}(f)$. Each 0-offset face is then colored with a vertex of G and the remaining faces are each colored with an edge of G.*

In the rest of the section, we study the spectral gap of the Markov chain $\mathcal{Q}_k^{\downarrow\uparrow}$, the down-up random walk on k-faces of $\mathcal{Q}$ induced by certain special weight functions – weight functions $w : k\text{-}\mathsf{faces}(\mathcal{Q}) \to \mathbb{R}_{\geqslant 0}$ with the property that there are two values w_I and w_J such that

$$w((F, f)) = \begin{cases} w_J & \text{if } (F, f) \text{ is 0-offset} \\ w_I & \text{otherwise.} \end{cases}$$

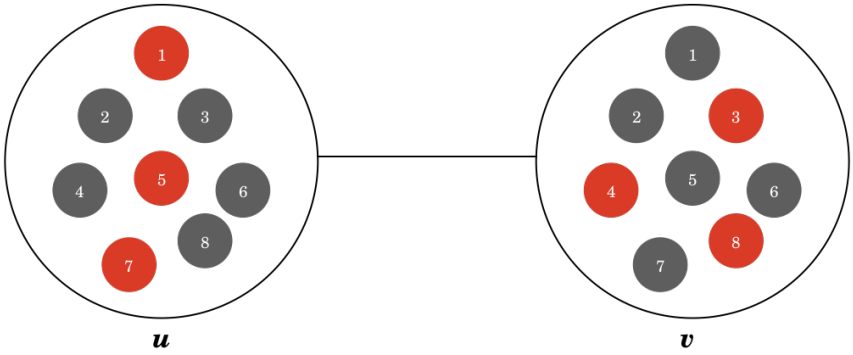


Figure 1 A 5-face in $\mathcal{Q}$. Corresponding 5-face in $\mathcal{B}$ is $\{1, 3, 4, 5, 7, 8\}$ is given by red vertices. Labeling is $(1, u), (3, v), (4, v), (5, u), (7, u), (8, v)$. $\{u, v\}$ is an edge in G. Color of 5-face is $\{u, v\}$.

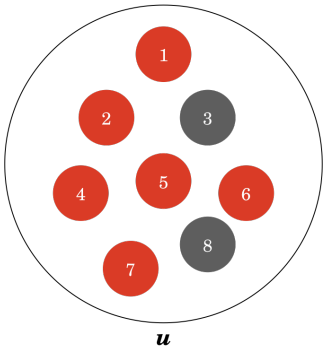


Figure 2 A 0-offset 5-face. Color of 5-face is $\{u\}$.

For instance, if we impose uniform weights on the highest dimensional faces of our complex, the propagated weights on the k-th level will satisfy the above property. The w_I and w_J values for this setup is in Appendix 1.

For the sequel, we use D to refer to the quantity $Tw_I + w_J$. The transition probabilities between states (F, f) and (F', f') depends on a number of conditions such as whether they are 0-offset or 1-offset or a different type, whether they arise from the same k-face in $\mathcal{B}$, and the colors of (F, f) and (F', f') respectively. We provide a detailed treatment of the transition probabilities $\mathcal{Q}_k^{\downarrow\uparrow}[(F, f) \to (F', f')]$ in Table 1 in Appendix A. From the transition probability table we observe that:

▶ **Observation 45.** *For all k-faces in $\mathcal{Q}_k^{\downarrow\uparrow}$, the self-loop probability is at least $\frac{1}{s-k} \cdot \frac{w_J}{D}$. Therefore, the smallest eigenvalue of $\mathcal{Q}_k^{\downarrow\uparrow}$ is at least $\frac{1}{s-k} \cdot \frac{w_J}{D} - 1$.*

5.2 High-Level Picture of $\mathcal{Q}_k^{\downarrow\uparrow}$

As noted in the previous subsection, each k-face can be described by three parameters: a base face $F \in k\text{-faces}(\mathcal{B})$, a "color" set C that is either a single vertex or an edge in $E(G)$, and a function $f : F \to C$. The walk $\mathcal{Q}_k^{\downarrow\uparrow}$ is difficult to analyze directly, but by grouping states based on these three parameters, we can decompose the walk into a projection and restriction chain, and analyze it using the tools from [11].

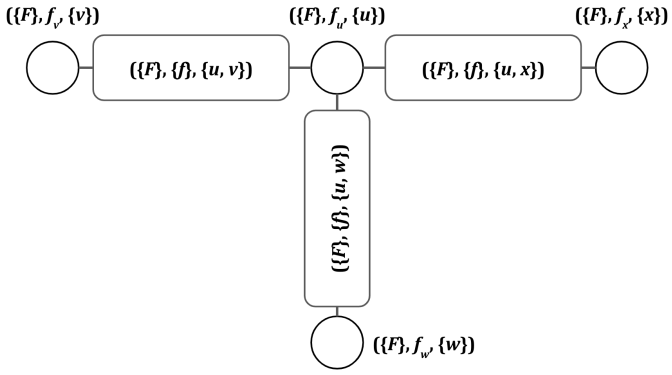


Figure 3 This figure illustrates $\mathcal{Q}_k^{\downarrow\uparrow}$, with states clustered by their color. The rounded rectangles correspond to colors that are edges, while circles correspond to colors that are single vertices. In each cluster, the $\{F\}$ indicates that all F could be represented. Similarly, $\{f\}$ indicates that any f with $\mathsf{image}(f)$ as the color set can be represented. We use f_u to denote the constant function on u.

At the outermost level, we can first group states into subchains based on their color. All subchains whose color is an edge (the rounded rectangles in Figure 3) are isomorphic to each other; similarly, all subchains whose color is a single vertex (the circles in Figure 3) are also isomorphic to each other. At first, it seems promising to partition $\mathcal{Q}_k^{\downarrow\uparrow}$ into these subchains; however, it is inconvenient that these subchains are not *all* isomorphic. To remedy this, we split the single-vertex-colored subchains into T isomorphic copies (with some changes to transition probabilities), and absorb them into the edge-colored subchains. This is detailed in the next section.

If we use this partition, the projection chain resembles a random walk on the *line graph* of G. Each restriction chain corresponds to all states of a single color C. The states are still represented by any base face $F \in k\text{-}\mathsf{faces}(\mathcal{B})$ and any function $f : F \to C$. To analyze each of these restriction chains, it is simplest to apply [11] once more.

Now, we first group states by which base face F they correspond to. The subchains derived from fixing a particular F (the rectangles in Figure 4) are all isomorphic to each other, which leads to a much simplified analysis. Using this partition, the projection chain is simply the k-down-up walk on $\mathcal{B}$. Each restriction chain is thus over states corresponding to a fixed base face F and fixed color C, but the function $f : F \to C$ is allowed to vary. At this point, we may assume $|C| = 2$; thus f corresponds to assigning every element of F one of two elements. The inner restriction chain can be modeled by a hypercube.

Thus, the spectral gap of $\mathcal{Q}_k^{\downarrow\uparrow}$ is a combination of the spectral gaps of (1) the line graph of G, (2) the k-down-up walk on $\mathcal{B}$, and (3) the random walk on a hypercube.

5.3 Splitting 0-Offset Vertices

Towards our end goal of lower bounding the spectral gap of $\mathcal{Q}_k^{\downarrow\uparrow}$, we find it convenient to analyze a related Markov chain $\widetilde{\mathcal{Q}}_k^{\downarrow\uparrow}$, since the related chain has a natural partition into isomorphic subchains. $\widetilde{\mathcal{Q}}_k^{\downarrow\uparrow}$ has the property that its spectrum contains that of $\mathcal{Q}_k^{\downarrow\uparrow}$, which lets us translate a lower bound on the spectral gap of $\widetilde{\mathcal{Q}}_k^{\downarrow\uparrow}$ to a lower bound on the spectral gap of $\mathcal{Q}_k^{\downarrow\uparrow}$.

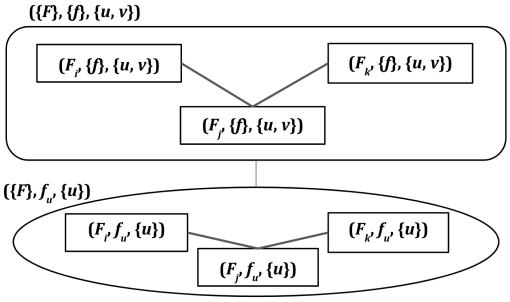


Figure 4 This figure illustrates a subchain of $\mathcal{Q}_k^{\downarrow\uparrow}$, for particular color $\{u, v\}$ and $\{u\}$. We can further cluster the states in this subchain by which face F in $\mathcal{B}$ they represent. Again, $\{f\}$ indicates that f can be any function with $\mathsf{image}(f)$ as the color.

▶ **Definition 46** (Split chain $\widetilde{\mathcal{Q}}_k^{\downarrow\uparrow}$ and coloring of states in $\widetilde{\mathcal{Q}}_k^{\downarrow\uparrow}$). *We identify each state in* $\mathsf{States}(\widetilde{\mathcal{Q}}_k^{\downarrow\uparrow})$ *with a tuple* (F, f, c) *where* (F, f) *is a face in* k-$\mathsf{faces}(\mathcal{Q})$ *and* c *is a color.*

1. *For each* 0*-offset face* (F, f) *in* k-$\mathsf{faces}(\mathcal{Q})$*, let* $\{u\}$ *be the color of* F*, and let the neighbors of* u *in* G *be* $v_1, \ldots, v_T$*.* $\mathsf{States}(\widetilde{\mathcal{Q}}_k^{\downarrow\uparrow})$ *contains the states* $(F, f, \{u, v_1\}), \ldots, (F, f, \{u, v_T\})$ *in place of the state* (F, f, u)*.*
2. *For each remaining* k*-face* (F, f) *of* $\mathcal{Q}$ *(i.e. each* k*-face that isn't* 0*-offset),* $\mathsf{States}(\widetilde{\mathcal{Q}}_k^{\downarrow\uparrow})$ *contains* $(F, f, \mathsf{image}(f))$*.*

For each pair of states $(F, f, c), (F', f', c')$ *in* $\mathsf{States}(\widetilde{\mathcal{Q}}_k^{\downarrow\uparrow})$*,*

$$\widetilde{\mathcal{Q}}_k^{\downarrow\uparrow}[(F, f, c) \to (F', f', c')] = \begin{cases} \frac{\mathcal{Q}_k^{\downarrow\uparrow}[(F,f)\to(F',f')]}{T} & \text{if } (F', f') \text{ is 0-offset} \\ \mathcal{Q}_k^{\downarrow\uparrow}[(F, f) \to (F', f')] & \text{otherwise.} \end{cases}$$

Intuitively, we want to split any transition to a 0-offset face in $\mathcal{Q}$ into T separate transitions in $\widetilde{\mathcal{Q}}_k^{\downarrow\uparrow}$, since each 0-offset face is also split into T new states.

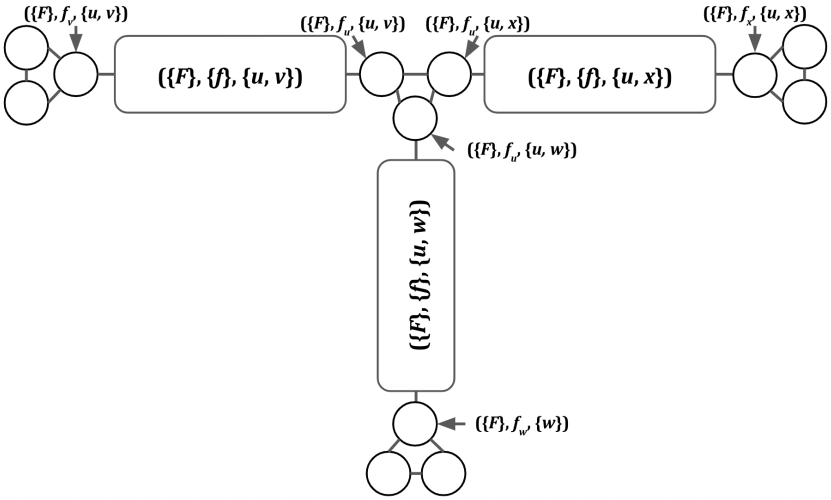


Figure 5 This figure illustrates the post-split vertices of Definition 46. The new vertices can take on any F, but their mappings f will be constant functions.

▶ **Definition 47.** *We say two* k*-faces* (F, f, e) *and* (F', f', e') *have identical base* k*-faces if* $F = F'$ *and different base* k*-faces if* $F \neq F'$*.*

▶ **Definition 48.** *Given a state (F, f, e) such that (F, f) is a 1-offset face, there is a single vertex v such that $f(v)$ is different from $f(u)$ for all u in $F \setminus \{v\}$. We call this vertex v a lonely vertex.*

In the next lemma, we show that the spectrum of the original Markov chain $\mathcal{Q}_k^{\downarrow\uparrow}$ is contained in that of $\widetilde{\mathcal{Q}}_k^{\downarrow\uparrow}$.

▶ **Lemma 49.** $\mathrm{Spec}\left(\mathcal{Q}_k^{\downarrow\uparrow}\right) \subseteq \mathrm{Spec}\left(\widetilde{\mathcal{Q}}_k^{\downarrow\uparrow}\right)$*, and therefore,* $\lambda_2(\mathcal{Q}_k^{\downarrow\uparrow}) \leqslant \lambda_2(\widetilde{\mathcal{Q}}_k^{\downarrow\uparrow})$.

The proof can be found in Appendix B.

5.3.1 Stationary Distribution of $\widetilde{\mathcal{Q}}_k^{\downarrow\uparrow}$

If we want to apply the projection and restriction framework to $\widetilde{\mathcal{Q}}_k^{\downarrow\uparrow}$, we first need to compute its stationary distribution. To do this, we take advantage of the time-reversibility of the high order random walks, and apply the detailed balance equations. The transition probabilities in $\widetilde{\mathcal{Q}}_k^{\downarrow\uparrow}$ are laid out in detail in Appendix A.

▶ **Lemma 50.** *The stationary distribution of the Markov chain $\widetilde{\mathcal{Q}}_k^{\downarrow\uparrow}$ is given by:*

$$\pi_{\widetilde{\mathcal{Q}}_k^{\downarrow\uparrow}}(x) = \begin{cases} \dfrac{1}{|E(G)|} \cdot \dfrac{1}{\binom{s}{k+1}} \cdot \dfrac{1}{2} \cdot \dfrac{w_J}{(2^k-1)Tw_I + w_J} & \text{for } x \text{ 0-offset} \\ \dfrac{1}{|E(G)|} \cdot \dfrac{1}{\binom{s}{k+1}} \cdot \dfrac{1}{2} \cdot \dfrac{Tw_I}{(2^k-1)Tw_I + w_J} & \text{otherwise} \end{cases}$$

Proof. Via the detailed balance equations, we first observe that all vertices with the same offset have the same stationary distribution. Now, let x be any 0-offset vertex and y be any 1-offset vertex. Using the detailed balance equations, we have:

$$\pi_{\widetilde{\mathcal{Q}}_k^{\downarrow\uparrow}}(x) \cdot \frac{w_I}{(k+1)(s-k)D} = \pi_{\widetilde{\mathcal{Q}}_k^{\downarrow\uparrow}}(y) \cdot \frac{w_J}{(k+1)(s-k)DT}$$

Now, let x be any t-offset vertex, with $t \geqslant 1$, and let y be any $(t+1)$-offset vertex. Again, using the detailed balance equations:

$$\pi_{\widetilde{\mathcal{Q}}_k^{\downarrow\uparrow}}(x) \cdot \frac{1}{2(k+1)(s-k)} = \pi_{\widetilde{\mathcal{Q}}_k^{\downarrow\uparrow}}(y) \cdot \frac{1}{2(k+1)(s-k)}$$

From here, we see that all 0-offset faces have one stationary distribution probability, and all other faces also share the same stationary probability. The relations above tell us that for a 0-offset vertex x, and a t-offset vertex y with $t \geqslant 1$:

$$\frac{\pi_{\widetilde{\mathcal{Q}}_k^{\downarrow\uparrow}}(x)}{\pi_{\widetilde{\mathcal{Q}}_k^{\downarrow\uparrow}}(y)} = \frac{w_J}{Tw_I}$$

Normalizing so that $\sum_{x \in \widetilde{\mathcal{Q}}_k^{\downarrow\uparrow}} \pi_{\widetilde{\mathcal{Q}}_k^{\downarrow\uparrow}}(x) = 1$ gives the desired result. ◀

5.4 Outer Projection and Restriction Chains

Now, we can further decompose $\widetilde{\mathcal{Q}}_k^{\downarrow\uparrow}$ into a projection chain and m isomorphic restriction chains, where $m = |E(G)|$, since we will have one partition element for each edge in G. Formally, we partition $\mathsf{States}(\widetilde{\mathcal{Q}}_k^{\downarrow\uparrow})$ into m disjoint sets $\Omega_1 \cup \cdots \cup \Omega_m$, where $\Omega_i = \{(F, f, c) \mid c = e_i\}$.

5.4.1 The Outer Projection Chain

The partition Ω induces a projection chain $([m], P_o)$. The state space is $[m]$. The edge set is

$$E(P_o) = \{\{i,j\} \mid \exists (F,f,e_i) \in \Omega_i \text{ and } (G,g,e_j) \in \Omega_j \text{ s.t. } \widetilde{\mathcal{Q}}_k^{\downarrow\uparrow}[(F,f,e_i) \to (G,g,e_j)] > 0\}$$

In words, we have an edge between i and j if there are transitions from Ω_i to Ω_j.
We obtain the following lower bound on the spectral gap of P_o.

▶ **Lemma 51.** *The spectral gap of P_o is*

$$\frac{\mathsf{TwoSidedGap}(G)}{2} \cdot \frac{w_J + Tw_I}{w_J + (2^k-1)Tw_I} \geqslant \frac{\mathsf{TwoSidedGap}(G)}{2(2^k-1)} \ .$$

A detailed account of the transitional probabilities of the projection chain $([m], P_o)$ can be found in Appendix C.1 and the proof of the lemma can be found in Appendix C.2.

5.4.2 The Outer Restriction Chain

Each partition block Ω_i induces a restriction chain $R_{o,i}$. We show that all restriction chains $R_{o,i}$ for $i \in [m]$ are isomorphic.

▶ **Lemma 52.** *For any $i \neq j$, $i,j \in [m]$, the restriction chains $R_{o,i}$ and $R_{o,j}$ are isomorphic.*

The proof is in Appendix D.1. The transition probabilities of a restriction chain $R_{o,i}$ is deduced in Appendix D.2.

5.4.3 Stationary Distribution of $R_{o,1}$

To compute the spectral gap of $R_{o,1}$, we will further decompose the chain in the next section. In order to apply the projection and restriction framework once more to $R_{o,1}$, we must again compute a stationary distribution.

▶ **Lemma 53.** *The stationary distribution of the outer restriction chain is given by:*

$$\pi_{R_{o,1}}(x) = \begin{cases} \dfrac{1}{\binom{s}{k+1}} \cdot \dfrac{1}{2} \cdot \dfrac{w_J}{(2^k-1)Tw_I + w_J} & \text{for } x \text{ 0-offset} \\ \dfrac{1}{\binom{s}{k+1}} \cdot \dfrac{1}{2} \cdot \dfrac{Tw_I}{(2^k-1)Tw_I + w_J} & \text{otherwise} \end{cases}$$

Proof. By Fact 16, $R_{o,1}$ is also time-reversible. We proceed using the same analysis we used for Lemma 50. At the very end, we use a slightly different normalization to get the desired result. ◀

5.5 Inner Projection and Restriction Chains

Now, we are left to study the outer restriction chain, which, for a fixed $e \in E(G)$, is composed of all (F,f,e) in $\mathsf{States}(\widetilde{\mathcal{Q}}_k^{\downarrow\uparrow})$. Again, we further decompose this chain into projection and restriction chains which are easier to analyze.

We group all (F,f,e) with the same $F \in k\text{-}\mathsf{faces}(\mathcal{B})$ into the same restriction state space Ω_F, which induces a projection chain resembling $\mathcal{B}_{\downarrow\uparrow}$, the down-up walk on k-faces of $\mathcal{B}$, and a restriction chain resembling a lazy random walk on a $(k+1)$-dimensional hypercube.

5.5.1 The Projection Chain

By defining the projection restriction chains as above, we end up with isomorphic restriction chains for each $F \in k\text{-faces}(\mathcal{B})$. Thus, we can identify each of the states of the inner projection chain P_I with some face $F \in k\text{-faces}(\mathcal{S})$. Let $\{F_i\}$ be this partition based on face.

Given $F, F' \in k\text{-faces}(\mathcal{S})$, we can only transition from F to F' either when $F = F'$, or when $F \cap F' \in (k-1)\text{-faces}$. This coincides with the feasible transitions in $\mathcal{B}_{\downarrow\uparrow}$.

We detail the transition probabilities in $\mathcal{B}_{\downarrow\uparrow}$ in Appendix E.2 and are able to obtain the following bounds on the spectral gap of the outer projection chain:

▶ **Lemma 54.** $\mathsf{OneSidedGap}(P_I) \geqslant \dfrac{1}{2T(k+1)}$.

The proof of Lemma 54 can be found in Appendix E.3.

5.5.2 The Restriction Chain

Each restriction chain R_I can be treated as a $(k+1)$-dimensional hypercube with self loops. To see this, note that each restriction chain is a set of states (F, f, e) in $\widetilde{\mathcal{Q}}_k^{\downarrow\uparrow}$ where both F and e are the same. There are thus 2^{k+1} states in each restriction chain, since for each x, we have two choices for $f(x)$. Associating x where $f(x) = u$ to a 0-coordinate in a hypercube vertex, and x where $f(x) = v$ to a 1-coordinate, gives us a bijection from the restriction chain to the hypercube.

The transition probabilities are summarized in Appendix F.1, and we show:

▶ **Lemma 55.** *If we impose uniform weights on the highest order faces,*

$$\mathsf{OneSidedGap}(R_I) \geqslant \frac{w_J}{2Tw_I} \cdot \frac{2w_J}{D(k+1)(s-k)} \geqslant \frac{1}{(k+1)(s-k)} \ .$$

We defer the proof of Lemma 55 to Appendix F.2. We also give relevant background in Appendix F.3.

5.6 Rapid Mixing for High Order Random Walks

Now we put together the decomposition theorem, the lower bounds for the spectral gaps of the project and restriction chains, and Observation 45 to obtain the following lower bound on the two-sided spectral gap of $\widetilde{\mathcal{Q}}_k^{\downarrow\uparrow}$:

▶ **Theorem 56** (Restatement of Theorem 35)**.** *The k down-up random walk $\mathcal{Q}_k^{\downarrow\uparrow}$ has one-sided spectral gap,*

$$\mathsf{TwoSidedGap}(\mathcal{Q}_k^{\downarrow\uparrow}) \geqslant \frac{\mathsf{TwoSidedGap}(G)}{64T(k+1)^2(s-k)(2^k-1)} \ . \tag{2}$$

Proof. Use $\mathsf{OneSidedGap}(M)$ to denote the spectral gap of a Markov chain M. We deduce from Lemma 49 and Theorem 17 that

$$
\begin{aligned}
&\mathsf{OneSidedGap}(\mathcal{Q}_k^{\downarrow\uparrow}) \\
&\quad \geqslant \mathsf{OneSidedGap}(\widetilde{\mathcal{Q}}_k^{\downarrow\uparrow}) && \text{(Lemma 49)} \\
&\quad \geqslant \min\left\{ \frac{\mathsf{OneSidedGap}(P_o)}{3}, \frac{\mathsf{OneSidedGap}(P_o)\mathsf{OneSidedGap}(R_{o,1})}{3\gamma_o + \mathsf{OneSidedGap}(P_o)} \right\} && \text{(Theorem 17 on } \widetilde{\mathcal{Q}}_k^{\downarrow\uparrow}) \\
&\quad \geqslant \min\left\{ \frac{\mathsf{OneSidedGap}(P_o)}{3}, \right. \\
&\qquad \frac{\mathsf{OneSidedGap}(P_o)}{3\gamma_o + \mathsf{OneSidedGap}(P_o)} \cdot \frac{\mathsf{OneSidedGap}(P_I)}{3}, \\
&\qquad \left. \frac{\mathsf{OneSidedGap}(P_o)}{3\gamma_o + \mathsf{OneSidedGap}(P_o)} \cdot \frac{\mathsf{OneSidedGap}(P_I)\mathsf{OneSidedGap}(R_I)}{3\gamma_I + \mathsf{OneSidedGap}(P_I)} \right\} && \text{(Theorem 17 on } R_{o,1}), \\
&\text{where}
\end{aligned}
$$

$$\gamma_o = \max_{i\in[m]} \max_{x\in\Omega_i} \sum_{y\in\Omega\setminus\Omega_i} \widetilde{\mathcal{Q}}_k^{\downarrow\uparrow}(x,y) < 1$$

$$\gamma_I = \max_{F\in k\text{-}\mathsf{faces}(\mathcal{S})} \max_{x\in V(R_I)} \sum_{y\in V(R_{o,1})\setminus V(R_I)} R_{o,1}(x,y) < 1.$$

Furthermore, Lemma 51, Lemma 54 and Lemma 55 provide lower bounds for $\mathsf{OneSidedGap}(P_o)$, $\mathsf{OneSidedGap}(P_I)$, and $\mathsf{OneSidedGap}(R_I)$. If we substitute the spectral-gap lower bounds, and an upper bound of 1 for both γ_o and γ_I, we obtain a lower bound on $\mathsf{OneSidedGap}(\mathcal{Q}_k^{\downarrow\uparrow})$:

$$\mathsf{OneSidedGap}(\mathcal{Q}_k^{\downarrow\uparrow}) \geqslant \frac{\mathsf{TwoSidedGap}(G)}{64T(k+1)^2(s-k)(2^k-1)}. \tag{3}$$

Observation 45 gives a lower bound on $1 - |\lambda_{\min}(\mathcal{Q}_k^{\downarrow\uparrow})|$ larger than the right hand side of (3), which immediately lets us upgrade the statement (3) to (2), thus proving the theorem. ◀

References

1. Nima Anari, Kuikui Liu, Shayan Oveis Gharan, and Cynthia Vinzant. Log-concave polynomials II: high-dimensional walks and an FPRAS for counting bases of a matroid. *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, June 2019.
2. László Babai. Spectra of Cayley graphs. *Journal of Combinatorial Theory, Series B*, 27(2):180–189, 1979.
3. Avraham Ben-Aroya and Amnon Ta-Shma. A combinatorial construction of almost-Ramanujan graphs using the zig-zag product. *SIAM Journal on Computing*, 40(2):267–290, 2011.
4. Nathanaël Berestycki. Lectures on mixing times. *Cambridge University*, 2014.
5. Michael Chapman, Nati Linial, and Yuval Peled. Expander Graphs–Both Local and Global. *arXiv preprint*, 2018. `arXiv:1812.11558`.
6. Irit Dinur and Tali Kaufman. High dimensional expanders imply agreement expanders. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 974–985. IEEE, 2017.
7. Shai Evra and Tali Kaufman. Bounded degree cosystolic expanders of every dimension. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 36–48. ACM, 2016.
8. Joel Friedman. A proof of Alon's second eigenvalue conjecture. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 720–724. ACM, 2003.
9. Mikhail Gromov. Singularities, expanders and topology of maps. Part 2: From combinatorics to topology via algebraic isoperimetry. *Geometric and Functional Analysis*, 20(2):416–526, 2010.

10 Shlomo Hoory, Nathan Linial, and Avi Wigderson. Expander graphs and their applications. *Bulletin of the American Mathematical Society*, 43(4):439–561, 2006.

11 Mark Jerrum, Jung-Bae Son, Prasad Tetali, Eric Vigoda, et al. Elementary bounds on Poincaré and log-Sobolev constants for decomposable Markov chains. *The Annals of Applied Probability*, 14(4):1741–1765, 2004.

12 Tali Kaufman, David Kazhdan, and Alexander Lubotzky. Ramanujan complexes and bounded degree topological expanders. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, pages 484–493. IEEE, 2014.

13 Tali Kaufman and David Mass. High dimensional random walks and colorful expansion. *arXiv preprint*, 2016. `arXiv:1604.02947`.

14 Tali Kaufman and Izhar Oppenheim. High order random walks: Beyond spectral gap. *arXiv preprint*, 2017. `arXiv:1707.02799`.

15 Tali Kaufman and Izhar Oppenheim. Construction of new local spectral high dimensional expanders. *arXiv preprint*, 2019. `arXiv:1710.05304`.

16 David A Levin and Yuval Peres. *Markov chains and mixing times*, volume 107. American Mathematical Soc., 2017.

17 Nathan Linial and Roy Meshulam. Homological connectivity of random 2-complexes. *Combinatorica*, 26(4):475–487, 2006.

18 Alexander Lubotzky. High dimensional expanders. *arXiv preprint*, 2017. `arXiv:1712.02526`.

19 Alexander Lubotzky, Ralph Phillips, and Peter Sarnak. Ramanujan graphs. *Combinatorica*, 8(3):261–277, 1988.

20 Alexander Lubotzky, Beth Samuels, and Uzi Vishne. Explicit constructions of Ramanujan complexes of type Ad. *European Journal of Combinatorics*, 26(6):965–993, 2005.

21 Omer Reingold, Salil Vadhan, and Avi Wigderson. Entropy waves, the zig-zag graph product, and new constant-degree expanders. *Annals of mathematics*, pages 157–187, 2002.

22 Horst Sachs. Über teiler, faktoren und charakteristische polynome von graphen. *Teil I. Wiss. Z. TH Ilmenau*, 12:7–12, 1966.

23 Michael Sipser and Daniel A Spielman. Expander codes. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 566–576. IEEE, 1994.

A Transition Probabilities of the Down-Up Walk

If we impose uniform weights at the highest order faces, then

$$w_I = 2^{H-k}$$
$$w_J = T2^{H-k} - (T-1)$$

For ease of notation, we use define another variable $D = Tw_I + w_J$, which will arise very often. Note that for the uniform weights case, w_J is only slightly smaller than Tw_I, which will help with some of our asymptotics.

Table 1 Transition Probabilities in $\mathcal{Q}_k^{\downarrow\uparrow}$.

Source	Delete	Target	Same k-face	Same edge	Probability	Count
0-offset	anything	0-offset	Yes	N/A	$\frac{1}{s-k}\cdot\frac{w_J}{D}$	1
			No	N/A	$\frac{1}{k+1}\cdot\frac{1}{s-k}\cdot\frac{w_J}{D}$	$(s-(k+1))(k+1)$
		1-offset	Yes	N/A	$\frac{1}{k+1}\cdot\frac{1}{s-k}\cdot\frac{w_I}{D}$	$(k+1)T$
			No	N/A	$\frac{1}{k+1}\cdot\frac{1}{s-k}\cdot\frac{w_I}{D}$	$(k+1)T(s-(k+1))$
1-offset	minority	0-offset	Yes	N/A	$\frac{1}{k+1}\cdot\frac{1}{s-k}\cdot\frac{w_J}{D}$	1
			No	N/A	$\frac{1}{k+1}\cdot\frac{1}{s-k}\cdot\frac{w_J}{D}$	$s-(k+1)$
		1-offset	Yes	Yes	$\frac{1}{k+1}\cdot\frac{1}{s-k}\cdot\frac{w_I}{D}$	1
			No	Yes	$\frac{1}{k+1}\cdot\frac{1}{s-k}\cdot\frac{w_I}{D}$	$s-(k+1)$
			Yes	No	$\frac{1}{k+1}\cdot\frac{1}{s-k}\cdot\frac{w_I}{D}$	$T-1$
			No	No	$\frac{1}{k+1}\cdot\frac{1}{s-k}\cdot\frac{w_I}{D}$	$(T-1)(s-(k+1))$
	majority	1-offset	Yes	Yes	$\frac{k}{k+1}\cdot\frac{1}{s-k}\cdot\frac{1}{2}$	1
			No	Yes	$\frac{1}{k+1}\cdot\frac{1}{s-k}\cdot\frac{1}{2}$	$(s-(k+1))k$
		2-offset	Yes	Yes	$\frac{1}{k+1}\cdot\frac{1}{s-k}\cdot\frac{1}{2}$	k
			No	Yes	$\frac{1}{k+1}\cdot\frac{1}{s-k}\cdot\frac{1}{2}$	$k(s-(k+1))$
t-offset	minority	t-offset	Yes	Yes	$\frac{t}{k+1}\cdot\frac{1}{s-k}\cdot\frac{1}{2}$	1
			No	Yes	$\frac{1}{k+1}\cdot\frac{1}{s-k}\cdot\frac{1}{2}$	$t(s-(k+1))$
		$t-1$-offset	Yes	Yes	$\frac{1}{k+1}\cdot\frac{1}{s-k}\cdot\frac{1}{2}$	t
			No	Yes	$\frac{1}{k+1}\cdot\frac{1}{s-k}\cdot\frac{1}{2}$	$t(s-(k+1))$
	majority	t-offset	Yes	Yes	$\frac{k+1-t}{k+1}\cdot\frac{1}{s-k}\cdot\frac{1}{2}$	1
			No	Yes	$\frac{1}{k+1}\cdot\frac{1}{s-k}\cdot\frac{1}{2}$	$(k+1-t)(s-(k+1))$
		$t+1$-offset	Yes	Yes	$\frac{1}{k+1}\cdot\frac{1}{s-k}\cdot\frac{1}{2}$	$k+1-t$
			No	Yes	$\frac{1}{k+1}\cdot\frac{1}{s-k}\cdot\frac{1}{2}$	$(k+1-t)(s-(k+1))$

B Spectrum of $\widetilde{\mathcal{Q}}_k^{\downarrow\uparrow}$: Proof of Lemma 49

Given a right eigenvector v of $\mathcal{Q}_k^{\downarrow\uparrow}$ for eigenvalue λ, we exhibit a right eigenvector $\widetilde{v}$ of $\widetilde{\mathcal{Q}}_k^{\downarrow\uparrow}$, also for eigenvalue λ. Let

$$\widetilde{v}[(F,f,c)] = \begin{cases} \frac{v[(F,f)]}{T} & \text{if } (F,f) \text{ is 0-offset} \\ v[(F,f)] & \text{otherwise.} \end{cases}$$

We now verify that $\widetilde{v}$ is indeed a right eigenvector of $\widetilde{P}$.

$$\begin{aligned}
\widetilde{P}\widetilde{v}[(F,f,c)] &= \sum_{(F',f',c')\in\mathsf{States}(\widetilde{\mathcal{Q}}_k^{\downarrow\uparrow})} \widetilde{\mathcal{Q}}_k^{\downarrow\uparrow}[(F',f',c')\to(F,f,c)]\widetilde{v}[F',f',c'] \\
&= \sum_{\substack{(F',f',c')\in\mathsf{States}(\widetilde{\mathcal{Q}}_k^{\downarrow\uparrow}) \\ (F',f') \text{ 0-offset}}} \widetilde{\mathcal{Q}}_k^{\downarrow\uparrow}[(F',f',c')\to(F,f,c)]\frac{v[F',f']}{T} + \\
&\quad \sum_{\substack{(F',f',c')\in\mathsf{States}(\widetilde{\mathcal{Q}}_k^{\downarrow\uparrow}) \\ (F',f') \text{ not 0-offset}}} \widetilde{\mathcal{Q}}_k^{\downarrow\uparrow}[(F',f',c')\to(F,f,c)]v[F',f']
\end{aligned}$$

Table 2 Transition Probabilities in $\widetilde{\mathcal{Q}}_k^{\downarrow\uparrow}$.

Source	Delete	Target	Same k-face	Same edge	Probability	Count
0-offset	anything	0-offset	Yes	Yes	$\frac{1}{s-k}\cdot\frac{w_J}{DT}$	1
				No		$T-1$
		0-offset	No	Yes	$\frac{1}{k+1}\cdot\frac{1}{s-k}\cdot\frac{w_J}{DT}$	$(s-(k+1))(k+1)$
				No		$(T-1)(s-(k+1))(k+1)$
		1-offset	Yes	Yes	$\frac{1}{k+1}\cdot\frac{1}{s-k}\cdot\frac{w_I}{D}$	$(k+1)$
				No		$(k+1)(T-1)$
		1-offset	No	Yes		$(k+1)(s-(k+1))$
				No		$(k+1)(T-1)(s-(k+1))$
1-offset	minority	0-offset	Yes	Yes	$\frac{1}{k+1}\cdot\frac{1}{s-k}\cdot\frac{w_J}{DT}$	1
				No		$T-1$
			No	Yes		$s-(k+1)$
				No		$(s-(k+1))(T-1)$
		1-offset	Yes	Yes	$\frac{1}{k+1}\cdot\frac{1}{s-k}\cdot\frac{w_I}{D}$	1
			No	Yes		$s-(k+1)$
			Yes	No		$T-1$
			No	No		$(T-1)(s-(k+1))$
	majority	1-offset	Yes	Yes	$\frac{k}{k+1}\cdot\frac{1}{s-k}\cdot\frac{1}{2}$	1
			No	Yes	$\frac{1}{k+1}\cdot\frac{1}{s-k}\cdot\frac{1}{2}$	$(s-(k+1))k$
		2-offset	Yes	Yes		k
			No	Yes		$k(s-(k+1))$
t-offset	minority	t-offset	Yes	Yes	$\frac{t}{k+1}\cdot\frac{1}{s-k}\cdot\frac{1}{2}$	1
			No	Yes	$\frac{1}{k+1}\cdot\frac{1}{s-k}\cdot\frac{1}{2}$	$t(s-(k+1))$
		$t-1$-offset	Yes	Yes		t
			No	Yes		$t(s-(k+1))$
	majority	t-offset	Yes	Yes	$\frac{k+1-t}{k+1}\cdot\frac{1}{s-k}\cdot\frac{1}{2}$	1
			No	Yes	$\frac{1}{k+1}\cdot\frac{1}{s-k}\cdot\frac{1}{2}$	$(k+1-t)(s-(k+1))$
		$t+1$-offset	Yes	Yes		$k+1-t$
			No	Yes		$(k+1-t)(s-(k+1))$

If (F,f,c) is a 0-offset face, then the above quantity is equal to

$$\begin{aligned}
&\sum_{\substack{(F',f')\in k\text{-faces}(\mathcal{Q})\\(F',f')\ 0\text{-offset}}} \frac{\mathcal{Q}_k^{\downarrow\uparrow}[(F',f')\to(F,f)]}{T}\cdot\frac{v[F',f']}{T}\cdot T + \sum_{\substack{(F',f')\in k\text{-faces}(\mathcal{Q})\\(F',f')\text{ not }0\text{-offset}}} \frac{\mathcal{Q}_k^{\downarrow\uparrow}[(F',f')\to(F,f)]}{T}v[(F',f')]\\
&\qquad = \frac{1}{T}\sum_{(F',f')\in k\text{-faces}(\mathcal{Q})} \mathcal{Q}_k^{\downarrow\uparrow}[(F',f')\to(F,f)]v[(F',f')]\\
&\qquad = \frac{1}{T}\lambda v[(F,f)]\\
&\qquad = \lambda\widetilde{v}[(F,f,c)].
\end{aligned}$$

And if (F,f,c) is not a 0-offset face, then the quantity is equal to

$$\begin{aligned}
&\sum_{\substack{(F',f')\in k\text{-faces}(\mathcal{Q})\\(F',f')\ 0\text{-offset}}} \mathcal{Q}_k^{\downarrow\uparrow}[(F',f')\to(F,f)]\cdot\frac{v[F',f']}{T}\cdot T + \sum_{\substack{(F',f')\in k\text{-faces}(\mathcal{Q})\\(F',f')\text{ not }0\text{-offset}}} \mathcal{Q}_k^{\downarrow\uparrow}[(F',f')\to(F,f)]v[(F',f')]\\
&\qquad = \frac{1}{T}\sum_{(F',f')\in k\text{-faces}(\mathcal{Q})} \mathcal{Q}_k^{\downarrow\uparrow}[(F',f')\to(F,f)]v[(F',f')]\\
&\qquad = \lambda v[(F,f)]\\
&\qquad = \lambda\widetilde{v}[(F,f,c)].
\end{aligned}$$

Since for every right eigenvector v of P, we can exhibit a right eigenvector $\widetilde{v}$ of $\widetilde{P}$, we can conclude that $\mathrm{Spec}\left(\mathcal{Q}_k^{\downarrow\uparrow}\right)\subseteq\mathrm{Spec}\left(\widetilde{\mathcal{Q}}_k^{\downarrow\uparrow}\right)$.

C Spectral Gap of Outer Projection Chain

C.1 Transition Probabilities of Outer Projection Chain

The table below summarizes the types of transition probabilities that occur between i and j in P_o. Each row corresponds to a specific vertex of "Source" type, and provides (1) the transition probability to a specific vertex of "Target" type (where "Same k-face" denotes a transition from (F, f) to (F, f')), and (2) the number of such transitions that occur from the source.

Source	Target	Same k-face	Probability	Count in Ω_j, $j \neq i$, $(i,j) \in E(G)$
0-offset	0-offset	Yes	$\frac{1}{s-k} \cdot \frac{w_J}{DT}$	1
		No	$\frac{1}{k+1} \cdot \frac{1}{s-k} \cdot \frac{w_J}{DT}$	$(k+1)(s-(k+1))$
	1-offset	Yes	$\frac{1}{k+1} \cdot \frac{1}{s-k} \cdot \frac{w_I}{D}$	$(k+1)$
		No	$\frac{1}{k+1} \cdot \frac{1}{s-k} \cdot \frac{w_I}{D}$	$(k+1)(s-(k+1))$
1-offset	0-offset	Yes	$\frac{1}{k+1} \cdot \frac{1}{s-k} \cdot \frac{w_J}{DT}$	1
		No	$\frac{1}{k+1} \cdot \frac{1}{s-k} \cdot \frac{w_J}{DT}$	$s-(k+1)$
	1-offset	Yes	$\frac{1}{k+1} \cdot \frac{1}{s-k} \cdot \frac{w_I}{D}$	1
		No	$\frac{1}{k+1} \cdot \frac{1}{s-k} \cdot \frac{w_I}{D}$	$s-(k+1)$

Using the table, Lemma 50, and the definition of projection chain from [11], the transition probabilities of P_o are:

$$P_o[i \to j] = \begin{cases} \frac{1}{2T} \cdot \frac{Tw_I + w_J}{[(2^k-1)Tw_I + w_J]}, & i \neq j, \text{ and } (i,j) \in E(P_o), \\ 1 - \left(\frac{T-1}{T}\right) \cdot \left(\frac{Tw_I + w_J}{(2^k-1)Tw_I + w_J}\right), & i = j, \\ 0 & \text{otherwise.} \end{cases}$$

C.2 Proof of Lemma 51

Due to the symmetry of the transition probabilities and the partition Ω, the spectrum of of P_o is easily computed from the spectrum of the following graph L:

- $V(L) = [m]$,
- $E(L) = \{(i,j) \in E(P_o) \mid i \neq j\}$.

▶ **Observation 57.** *L is the line graph of the base expander G.*

Proof. By definition of the partition Ω, there is a natural bijection between vertices in $V(L)$ and edges in $E(G)$. By construction, $(i,j) \in E(L)$ if and only if there exists $\{(F,f,e_i),(G,g,e_j)\} \in E(\widetilde{\mathcal{Q}}_k^{\downarrow\uparrow})$ such that $(F,f,e_i) \in \Omega_i$ and $(G,g,e_j) \in \Omega_j$. In the chain $\widetilde{\mathcal{Q}}_k^{\downarrow\uparrow}$, two states (F,f,e_i) and (G,g,e_j) from different partition sets are connected only if they share a common endpoint in G. Thus, $\{i,j\} \in E(P_o)$ only if e_i, e_j are adjacent in G. The if direction is straightforward from the construction of P_o. So L is the line graph of G. ◀

The relationship between $\mathsf{Spectrum}(L(G))$ and $\mathsf{Spectrum}(G)$ is also well understood.

▶ **Theorem 58** ([22]). *If G is a graph of degree d with n vertices and $L(G)$ its line graph, then the characteristic polynomials $\chi(G,\lambda)$ and $\chi(L(G),\lambda)$ satisfy*

$$\chi(L(G),\lambda) = (\lambda+2)^{n(\frac{d}{2}-1)}\chi(G,\lambda+2-d).$$

Proof of Lemma 51. Using Observation 57 and Theorem 58, we relate the spectrum of L to the spectrum of G. Specifically, if λ is an eigenvalue of the normalized adjacency matrix of G, then $\frac{\lambda T+T-2}{2T-2}$ is an eigenvalue of the normalized adjacency matrix of L. From this, one can deduce that $\mathsf{TwoSidedGap}(L) = \frac{T}{2T-2} \cdot \mathsf{TwoSidedGap}(G)$.

$$P_o = \left(1 - \left(\frac{T-1}{T}\right) \cdot \frac{(w_J + Tw_I)}{(w_J + (2^k-1)Tw_I)}\right) \cdot I + \frac{T-1}{T} \cdot \frac{(w_J + Tw_I)}{w_J + (2^k-1)Tw_I} \cdot \mathrm{Adj}(L)$$

It follows that if v, λ is an eigenvector, eigenvalue pair of $\mathrm{Adj}(L)$, then

$$v, 1 - \left(\frac{T-1}{T}\right) \cdot \frac{(w_J + Tw_I)}{(w_J + (2^k-1)Tw_I)} + \lambda \cdot \frac{T-1}{T} \cdot \frac{(w_J + Tw_I)}{w_J + (2^k-1)Tw_I}$$

is an eigenvector, eigenvalue pair of P_o. Therefore,

$$\begin{aligned}\mathsf{TwoSidedGap}(P_o) &= \mathsf{TwoSidedGap}(L) \cdot \frac{T-1}{T} \cdot \frac{w_J + Tw_I}{w_J + (2^k-1)Tw_I} \\ &= \frac{\mathsf{TwoSidedGap}(G)}{2} \cdot \frac{w_J + Tw_I}{w_J + (2^k-1)Tw_I}.\end{aligned}$$

◀

D Outer Restriction Chains

D.1 Proof of Lemma 52

Let $e_i, e_j \in E(G)$ be the edges corresponding to Ω_i, Ω_j respectively. Suppose $e_i = \{u_i, v_i\}$ and $e_j = \{u_j, v_j\}$. Define a map $t_{ij} \colon e_i \to e_j$ to be $t_{ij}(u_i) = u_j, t_{ij}(v_i) = v_j$. Then $R_{o,i}$ and $R_{o,j}$ are isomorphic under the map $M_{ij} \colon \Omega_i \to \Omega_j, (F, f, e_i) \to (F, t_{ij} \circ f, e_j)$.

D.2 Transition Probabilities of Outer Restriction Chains

Since the restriction chains are isomorphic, we can focus on $R_{o,1}$ without loss of generality. Using the decomposition rule given in Section 2.2.1, we can compute the transition probabilities of $R_{o,1}$:

- For all 0-offset (F, f, e_1), the self loop probability is

  $$\frac{T-1}{T} + \frac{w_J}{DT(s-k)}.$$

  The transition probability to each of its $(k+1)(s-k-1)$ adjacent 0-offset neighbors (F', f, e_1) is

  $$\frac{w_J}{DT(k+1)(s-k)}.$$

  The transition probability to each of its $(k+1)(s-k)$ non-0-offset neighbors (F', f') is

  $$\frac{w_I}{D(k+1)(s-k)}.$$

- For all 1-offset (F, f, e_1), the self loop probability is

  $$\frac{(T-1)}{T(k+1)} + \frac{w_I}{D(k+1)(s-k)} + \frac{k}{2(k+1)(s-k)}.$$

The transition probability to each of its $(s-k)$ 0-offset neighbors (F', f') is

$$\frac{w_J}{DT(k+1)(s-k)}.$$

The transition probability to each of its k non-0-offset neighbors with identical base k-face (F, f', e_1) is

$$\frac{1}{2(k+1)(s-k)}.$$

The transition probability to each of its $(s-k-1)$ non-0-offset neighbors with a different base k-face (F', f', e_1) reached by deleting the lonely[4] vertex and adding back a different lonely vertex is

$$\frac{w_I}{D(k+1)(s-k)}.$$

The transition probability to each of its $2k(s-k-1)$ non-0-offset neighbors with a different base k-face (F', f', e_1) reached by deleting a non-lonely vertex and adding back any other vertex is

$$\frac{1}{2(k+1)(s-k)}.$$

- For the remaining (F, f, e_1), the self loop probability is

$$\frac{1}{2(s-k)}.$$

The transition probability to each of its $(k+1)$ neighbors with an identical base k-face (F, f', e_1) is

$$\frac{1}{2(k+1)(s-k)}.$$

The transition probability to each of its $2(k+1)(s-k-1)$ neighbors with a different base k-face (F', f', e_1) is also

$$\frac{1}{2(k+1)(s-k)}.$$

E Spectral Gap of Inner Projection Chain

E.1 Lazy Random Walks

Both the inner projection and the inner restriction chains have self-loops, so it will be useful to first present some preliminary results on lazy random walks. If we start with Markov chain $\widetilde{M} = (\Omega, \widetilde{P})$ and wish to add a uniform self loop probability to each state to get Markov chain $M = (\Omega, P)$, we write P as a convex combination of $\widetilde{P}$ and I:

$$P = c \cdot I + (1-c) \cdot \widetilde{P}, \text{ where } 0 \leqslant c \leqslant 1$$

Since this convex combination will appear a few different times throughout this paper, we'll prove a basic fact about the spectral gap of P:

[4] Recall that "lonely" was defined in Definition 48

▶ **Lemma 59.** *For $M = (\Omega, P)$ as defined above:*

$$\mathsf{OneSidedGap}(M) = (1-c) \cdot \mathsf{OneSidedGap}(\widetilde{M})$$

Proof. Let λ be any eigenvalue of $\widetilde{P}$, with associated eigenvector v. Then, v is also an eigenvector for P for eigenvalue:

$$c + (1-c) \cdot \lambda(M)$$

To see this, $Pv = \left[c \cdot I + (1-c) \cdot \widetilde{P}\right] v = cv + (1-c)\left(\widetilde{P}v\right) = [c + (1-c) \cdot \lambda]\, v$. The spectrum of $\widetilde{M}$ is a linear shift and scaling of the spectrum of M, and the spectral gap scales by $(1-c)$. ◀

E.2 Transition Probabilities of Outer Projection Chain

The table below indicates the transition probabilities from a specific face of "Source" type in F_i to various "Target" faces in F_j for $j \neq i$. In the last column, we count transitions to *any* F_j, rather than a specific F_j; this made our computations much easier. Due to the symmetry of the $\{F_i\}$ partition elements, to get the transition from F_i to a specific F_j, we simply divide the transition probability to $\bigcup_{j \neq i} F_j$ by the number of F_j adjacent to F_i, which is $(k+1)(s-(k+1))$.

Source	Delete	Target	Probability	Count in F_j, $j \neq i$
0-offset	anything	0-offset	$\frac{1}{k+1} \cdot \frac{1}{s-k} \cdot \frac{w_J}{DT}$	1
		1-offset	$\frac{1}{k+1} \cdot \frac{1}{s-k} \cdot \frac{w_I}{D}$	1
1-offset	minority	0-offset	$\frac{1}{k+1} \cdot \frac{1}{s-k} \cdot \frac{w_J}{DT}$	1
		1-offset	$\frac{1}{k+1} \cdot \frac{1}{s-k} \cdot \frac{w_I}{D}$	1
	majority	1-offset	$\frac{1}{k+1} \cdot \frac{1}{s-k} \cdot \frac{1}{2}$	1
		2-offset	$\frac{1}{k+1} \cdot \frac{1}{s-k} \cdot \frac{1}{2}$	1
t-offset	minority	t-offset	$\frac{1}{k+1} \cdot \frac{1}{s-k} \cdot \frac{1}{2}$	1
		$(t-1)$-offset	$\frac{1}{k+1} \cdot \frac{1}{s-k} \cdot \frac{1}{2}$	1
	majority	t-offset	$\frac{1}{k+1} \cdot \frac{1}{s-k} \cdot \frac{1}{2}$	1
		$(t+1)$-offset	$\frac{1}{k+1} \cdot \frac{1}{s-k} \cdot \frac{1}{2}$	1

Using the table above, Lemma 53, and the framework of [11], the specific transition probabilities for each state in the projection chain are:

- $p := \dfrac{1}{T(k+1)(s-k)} \cdot \dfrac{[(2^k-2)T+1]Tw_I + w_J}{(2^k-1)Tw_I + w_J}$ to each of its $(k+1)(s-(k+1))$ neighbors.
- $1-(k+1)(s-(k+1))p$ for self loops, which can be verified to be nonzero.

E.3 Proof of Lemma 54

Let $\widetilde{\mathcal{B}_{\downarrow\uparrow}}$ be the non-lazy version (i.e. no self loops) of $\mathcal{B}_{\downarrow\uparrow}$. Since in our construction, $\mathcal{B}_{\downarrow\uparrow}$ is a complete complex, $w(F)$ is uniform over $F \in k$-faces, so all transitions in $\widetilde{\mathcal{B}_{\downarrow\uparrow}}$ are also uniform. To understand the spectrum of P_I, we can express the transition matrix of P_I as:

$$(k+1)(s-(k+1))p \cdot \widetilde{\mathcal{B}_{\downarrow\uparrow}} + [1-(k+1)(s-(k+1))p] \cdot \nVdash$$

Luckily, for $\mathcal{B}_{\downarrow\uparrow}$ a complete complex, the spectrum of $\widetilde{\mathcal{B}_{\downarrow\uparrow}}$ is well understood. The following can be deduced from the main theorem of [14].

▶ **Theorem 60.** $\mathsf{OneSidedGap}(\mathcal{B}_{\downarrow\uparrow}) \geqslant \dfrac{1}{(k+1)}$.

We can now compute the second largest eigenvalue of the non-lazy walk $\widetilde{\mathcal{B}_{\downarrow\uparrow}}$.

▶ **Corollary 61.** $\mathsf{OneSidedGap}(\widetilde{\mathcal{B}_{\downarrow\uparrow}}) \geqslant \dfrac{1}{(k+1)} + \dfrac{1}{(k+1)(s-k-1)}$.

Proof. Since we are working with a complete complex, all weights on sets of a given size are uniform. Thus, the self-loop probability of $\mathcal{B}_{\downarrow\uparrow}$ is $\frac{1}{s-k}$.

We can next write $\mathcal{B}_{\downarrow\uparrow} = \frac{1}{s-k} \cdot I + (1 - \frac{1}{s-k}) \cdot \widetilde{\mathcal{B}_{\downarrow\uparrow}}$. Using Lemma 59, we conclude that

$$\mathsf{OneSidedGap}(\widetilde{\mathcal{B}_{\downarrow\uparrow}}) = \frac{\mathsf{OneSidedGap}(\mathcal{B}_{\downarrow\uparrow})}{(1 - \frac{1}{s-k})}$$

We get the desired result after substituting $\frac{1}{(k+1)}$ as a lower bound for $\mathsf{OneSidedGap}(\widetilde{\mathcal{B}_{\downarrow\uparrow}})$. ◀

Proof of Lemma 54. By Lemma 59 again, we have

$$\mathsf{OneSidedGap}(P_I) = \mathsf{OneSidedGap}(\widetilde{\mathcal{B}_{\downarrow\uparrow}}) \cdot (k+1)(s-(k+1))p.$$

Substituting for p:

$$\begin{aligned}
\mathsf{OneSidedGap}(P_I) &\geqslant \left[\frac{1}{(k+1)} + \frac{1}{(k+1)(s-k-1)}\right] \cdot \left[\frac{s-k-1}{T(s-k)} \cdot \frac{[(2^k-2)T+1]Tw_I + w_J}{(2^k-1)Tw_I + w_J}\right] \\
&\geqslant \left[\frac{1}{(k+1)} + \frac{1}{(k+1)(s-k-1)}\right] \cdot \frac{1}{2T} \\
&\geqslant \frac{1}{2T(k+1)}
\end{aligned}$$

It can be verified that $\dfrac{1}{2T}$ is a lower bound on $\dfrac{s-k-1}{T(s-k)} \cdot \dfrac{[(2^k-2)T+1]Tw_I + w_J}{(2^k-1)Tw_I + w_J}$. ◀

F Spectral Gap of Inner Restriction Chain

F.1 Transition Probabilities of Inner Restriction chain

The transition probabilities can be summarized succinctly:

Source	Delete	Target	Probability
0-offset	anything	1-offset	$\frac{1}{k+1} \cdot \frac{1}{s-k} \cdot \frac{w_I}{D}$
1-offset	minority	0-offset	$\frac{1}{k+1} \cdot \frac{1}{s-k} \cdot \frac{w_J}{DT}$
	majority	2-offset	$\frac{1}{k+1} \cdot \frac{1}{s-k} \cdot \frac{1}{2}$
t-offset	minority	$(t-1)$-offset	$\frac{1}{k+1} \cdot \frac{1}{s-k} \cdot \frac{1}{2}$
	majority	$(t+1)$-offset	$\frac{1}{k+1} \cdot \frac{1}{s-k} \cdot \frac{1}{2}$

F.2 Proof of Lemma 55

We can also define a related chain U, that has the same state space and transitions as R, but the self loop probabilities are uniform across all vertices. More precisely:

- For *all* hypercube vertices, the self loop probability is $1 - \dfrac{w_I}{D(S-k)}$. The transition probability to each of their $(k+1)$ neighbors in the hypercube is

$$\frac{w_I}{D(k+1)(s-k)}.$$

The goal of this section is to bound on the spectral gap of R_I. Our approach relates the spectrum of R to the spectrum of U. Due to the uniformity of the self loop probabilities, the spectrum of U is easy to compute.
For ease, we will write $\widetilde{D} = 2w_J + w_I T(2^{k+1} - 2)$. The key bound on $\mathsf{OneSidedGap}(R_I)$, which we provide a proof of in Appendix F.4 is the following:

▶ **Lemma 62.**

$$\mathsf{OneSidedGap}(R_I) \geqslant \frac{w_J \widetilde{D}}{2^{k+1} \cdot (Tw_I)^2} \cdot \mathsf{OneSidedGap}(U) \geqslant \frac{w_J}{2Tw_I} \cdot \mathsf{OneSidedGap}(U) \ .$$

We are able to explicitly compute $\mathsf{OneSidedGap}(U) = \frac{2w_I}{D(k+1)(s-k)}$ (see Lemma 67). The conclusion of Lemma 55 is then immediate.

F.3 Variational Characterization of Spectral Gap

We will also use a different, variational characterization of the spectral gap of a time-reversible Markov chain (Ω, P), which will prove useful when working with self loops that have different probabilities. This characterization provides bounds on λ_2 without forcing us to analyze the chain's entire spectrum [4].

▶ **Definition 63.** *Let $M = (\Omega, P)$ be a time-reversible Markov chain. For functions $f, g : \Omega \to \mathbb{R}$, the Dirichlet form corresponding to M is:*

$$\mathcal{E}_M(f, g) = \frac{1}{2} \sum_{x \in \Omega} \sum_{y \in \Omega} \pi_M(x) M[x \to y] \cdot [f(x) - f(y)][g(x) - g(y)]$$

We may omit the subscript M when there is no ambiguity.

▶ **Definition 64.** *Again, let (Ω, P) be a time-reversible Markov chain with stationary distribution π_M. For a functions $f : \Omega \to \mathbb{R}$, the variance corresponding to M is:*

$$\mathbf{Var}_M(f) = \frac{1}{2} \sum_{x \in \Omega} \sum_{y \in \Omega} \pi_M(x) \pi_M(y) \cdot [f(x) - f(y)]^2$$

We may omit the subscript M when there is no ambiguity. This definition is equivalent to

$$\mathbf{Var}_M(f) = \mathbf{E}_{\pi_M}[f^2] - \mathbf{E}_{\pi_M}[f]^2$$

These definitions are equivalent because for X, Y i.i.d, $\mathbf{Var}(X) = \frac{1}{2}\mathbf{E}[(X - Y)^2]$.

▶ **Theorem 65.** *Let $M = (\Omega, P)$ be a time-reversible Markov Chain. Then:*

$$\mathsf{OneSidedGap}(M) = \inf \left\{ \frac{\mathcal{E}_M(f, f)}{\mathbf{Var}_M(f)} \mid f : \Omega \to \mathbb{R}, \mathbf{Var}_M(f) \neq 0 \right\}$$

Often, we will not be able to compute the exact spectral gap of a chain, but it will suffice to have a lower bound on it. We can determine whether λ is a lower bound on $\mathsf{OneSidedGap}(M)$ by checking if it satisfies the *Poincaré inequality*:

▶ **Definition 66.** *We say $\lambda \geqslant 0$ satisfies the Poincaré inequality if for all $f : \Omega \to \mathbb{R}$:*

$$\lambda \cdot \mathbf{Var}_M(f) \leqslant \mathcal{E}_M(f, f)$$

By the variational characterization of spectral gap, we would also have $\lambda \leqslant \mathsf{OneSidedGap}(M)$.

F.4 Proof of Lemma 62

▶ **Lemma 67.** *Let H be the uniform, non-lazy walk on the $(k+1)$-dim. hypercube. Then $\lambda_2(H) = \frac{2}{k+1}$.*

Proof. See [2] for a thorough treatment of Cayley graphs. The $(k+1)$-dimensional hypercube is the Cayley graph derived from the cyclic group $\mathbb{Z}_2^{k+1}$. ◀

▶ **Observation 68.** *$\lambda(U)$ is $\frac{2w_I}{D(k+1)(S-k)}$.*

Proof. Let P_H denote the transition matrix of a uniform random walk on a $(k+1)$-dimensional hypercube, with no self loops. Then, the transition matrix P_U of U can be expressed as:

$$P_U = \left(1 - \frac{w_I}{D(S-k)}\right) \cdot I + \frac{w_I}{D(S-k)} \cdot P_H$$

By Lemma 59, we have $\lambda(U) = \lambda(H) \cdot \frac{w_I}{D(S-k)}$, so we get the desired result via Lemma 67. ◀

We also observe that the stationary distribution of U, which we will call π_U, is uniform over the 2^{k+1} states. The stationary distribution of R_I, denoted π_{R_I} can also be described explicitly.

▶ **Observation 69.** *The stationary distribution π_{R_I} of chain R_I is*

$$\pi_{R_I}(x) = \begin{cases} \frac{w_J}{2w_J + w_I T(2^{k+1}-2)} & \text{if } x \in \{\vec{0}, \vec{1}\} \\ \frac{Tw_I}{2w_J + Tw_I(2^{k+1}-2)} & \text{otherwise} \end{cases}$$

Proof. By time reversibility of R_I [11], the detailed balance equations imply that for all y that are t-offset, for $t \geqslant 1$, the stationary probability $\pi_{R_I}(y)$ is the same, and $\pi_{R_I}(\vec{0}) = \pi_{R_I}(\vec{1})$.

Let x be 0-offset and y be 1-offset. Again, by time-reversibility of R_i and detailed balance:

$$\pi_{R_I}(x) \cdot \frac{w_I}{D(k+1)(S-k)} = \pi_{R_I}(y) \cdot \frac{w_J}{DT(k+1)(S-k)}$$

This tells us $\pi_{R_I}(x) = \frac{w_J}{Tw_I} \cdot \pi_{R_I}(y)$. Solving for $\sum_{x \in \{0,1\}^{k+1}} \pi_{R_I}(x) = 1$ gives the desired result. ◀

Recall that we write $\widetilde{D} = 2w_J + w_I T(2^{k+1} - 2)$.

Proof of Lemma 62. Let g be a real-valued function over the k-faces of LocalDensifier$(G, \mathcal{S})$. Using Theorem 65, it suffices to prove that for all g,

$$\frac{\mathcal{E}_{R_I}(g,g)}{\mathbf{Var}_{R_I}(g,g)} \geqslant \frac{w_J \widetilde{D}}{2^{k+1} \cdot (Tw_I)^2} \cdot \frac{\mathcal{E}_U(g,g)}{\mathbf{Var}_U(g,g)}$$

First, we compute both $\mathcal{E}_U(g,g)$ and $\mathcal{E}_R(g,g)$.

$$\begin{aligned} \mathcal{E}_U(g,g) &= \frac{1}{2} \sum_{x,y \in \{0,1\}^{(k+1)}} \pi_U(x) \cdot [g(x) - g(y)]^2 \cdot P_U(x,y) \\ &= \frac{1}{2} \cdot \frac{1}{2^{(k+1)}} \cdot \frac{w_I}{D(k+1)(S-k)} \sum_{x,y \in \{0,1\}^{(k+1)}} [g(x) - g(y)]^2 \end{aligned}$$

$$\begin{aligned}
\mathcal{E}_{R_I}(g,g) &= \frac{1}{2}\sum_{x,y\in\{0,1\}^{(k+1)}:x\in\{\vec{0},\vec{1}\}} \pi_{R_I}(x)\cdot[g(x)-g(y)]^2\cdot P_{R_I}(x,y)\\
&\quad+\frac{1}{2}\sum_{x,y\in\{0,1\}^{(k+1)}:x\notin\{\vec{0},\vec{1}\}} \pi_{R_I}(x)\cdot[g(x)-g(y)]^2\cdot P_{R_I}(x,y)\\
&= \frac{1}{2}\sum_{x,y\in\{0,1\}^{(k+1)}:x\in\{\vec{0},\vec{1}\}} \frac{w_J}{\widetilde{D}}\cdot[g(x)-g(y)]^2\cdot\frac{w_I}{D(k+1)(S-k)}\\
&\quad+\frac{1}{2}\sum_{x\in\{0,1\}^{(k+1)}:x\text{ 1-balanced}} \frac{Tw_I}{\widetilde{D}}\cdot\left(\sum_{y\in\{0,1\}^{(k+1)}:y\text{ 0-balanced}}[g(x)-g(y)]^2\cdot\frac{w_J}{DT(k+1)(S-k)}\right.\\
&\quad\left.+\sum_{y\in\{0,1\}^{(k+1)}:y\text{ not 0-balanced}}[g(x)-g(y)]^2\cdot\frac{1}{2(k+1)(S-k)}\right)\\
&\quad+\frac{1}{2}\sum_{x,y\in\{0,1\}^{(k+1)}:x,y\notin\{\vec{0},\vec{1}\}} \frac{Tw_I}{\widetilde{D}}\cdot[g(x)-g(y)]^2\cdot\frac{1}{2(k+1)(S-k)}\\
&\geqslant \frac{1}{2}\cdot\frac{w_Iw_J}{\widetilde{D}D(k+1)(S-k)}\sum_{x,y\in\{0,1\}^{(k+1)}}[g(x)-g(y)]^2
\end{aligned}$$

From the above computations, we can conclude that

$$\mathcal{E}_{R_I}(g,g)\geqslant\frac{2^{(k+1)}\cdot w_J}{\widetilde{D}}\cdot\mathcal{E}_U(g,g)$$

Similarly, we can compute both $\mathbf{Var}_U(g)$ and $\mathbf{Var}_R(g)$:

$$\begin{aligned}
\mathbf{Var}_U(g) &= \frac{1}{2}\sum_{x,y\in\{0,1\}^{k+1}}\pi_U(x)\pi_U(y)[f(x)-f(y)]^2\\
&= \frac{1}{2}\cdot\frac{1}{2^{2(k+1)}}\sum_{x,y\in\{0,1\}^{k+1}}[f(x)-f(y)]^2\\
\mathbf{Var}_{R_I}(g,g) &= \frac{1}{2}\sum_{x,y\in\{\vec{0},\vec{1}\}}\pi_{R_I}(x)\pi_{R_I}(y)[f(x)-f(y)]^2\\
&\quad+\frac{1}{2}\sum_{\substack{x\in\{\vec{0},\vec{1}\},\ y\in\{0,1\}^{k+1}\setminus\{\vec{0},\vec{1}\}\text{ or}\\ x\in\{0,1\}^{k+1}\setminus\{\vec{0},\vec{1}\},\ y\in\{\vec{0},\vec{1}\}}}\pi_{R_I}(x)\pi_{R_I}(y)[f(x)-f(y)]^2\\
&\quad+\frac{1}{2}\sum_{x,y\in\{\vec{0},\vec{1}\}}\pi_{R_I}(x)\pi_{R_I}(y)[f(x)-f(y)]^2\\
&= \frac{1}{2}\sum_{x,y\in\{\vec{0},\vec{1}\}}\frac{w_J^2}{\widetilde{D}^2}[f(x)-f(y)]^2+\frac{1}{2}\sum_{\substack{x\in\{\vec{0},\vec{1}\},\ y\in\{0,1\}^{k+1}\setminus\{\vec{0},\vec{1}\}\text{ or}\\ x\in\{0,1\}^{k+1}\setminus\{\vec{0},\vec{1}\},\ y\in\{\vec{0},\vec{1}\}}}\frac{Tw_Iw_J}{\widetilde{D}^2}[f(x)-f(y)]^2\\
&\quad+\frac{1}{2}\sum_{x,y\in\{\vec{0},\vec{1}\}}\frac{(Tw_I)^2}{\widetilde{D}^2}[f(x)-f(y)]^2\\
&\leqslant \frac{1}{2}\cdot\frac{(Tw_I)^2}{\widetilde{D}^2}\sum_{x,y\in\{0,1\}^{k+1}}[f(x)-f(y)]^2
\end{aligned}$$

From the above computations, we can conclude that

$$\mathbf{Var}_{R_I}(g)\leqslant\frac{2^{2(k+1)}\cdot(Tw_I)^2}{\widetilde{D}^2}\mathbf{Var}_U(g)$$

Combining this with what we know about $\mathcal{E}_{R_I}(g,g)$ and $\mathcal{E}_U(g,g)$, we conclude the lemma. ◀

Approximating Cumulative Pebbling Cost Is Unique Games Hard

Jeremiah Blocki
Department of Computer Science, Purdue University, West Lafayette, IN, USA
https://www.cs.purdue.edu/homes/jblocki
jblocki@purdue.edu

Seunghoon Lee
Department of Computer Science, Purdue University, West Lafayette, IN, USA
https://www.cs.purdue.edu/homes/lee2856
lee2856@purdue.edu

Samson Zhou
School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA
https://samsonzhou.github.io/
samsonzhou@gmail.com

Abstract

The cumulative pebbling complexity of a directed acyclic graph G is defined as $\mathsf{cc}(G) = \min_P \sum_i |P_i|$, where the minimum is taken over all legal (parallel) black pebblings of G and $|P_i|$ denotes the number of pebbles on the graph during round i. Intuitively, $\mathsf{cc}(G)$ captures the amortized Space-Time complexity of pebbling m copies of G in parallel. The cumulative pebbling complexity of a graph G is of particular interest in the field of cryptography as $\mathsf{cc}(G)$ is tightly related to the amortized Area-Time complexity of the Data-Independent Memory-Hard Function (iMHF) $f_{G,H}$ [7] defined using a constant indegree directed acyclic graph (DAG) G and a random oracle $H(\cdot)$. A secure iMHF should have amortized Space-Time complexity as high as possible, e.g., to deter brute-force password attacker who wants to find x such that $f_{G,H}(x) = h$. Thus, to analyze the (in)security of a candidate iMHF $f_{G,H}$, it is crucial to estimate the value $\mathsf{cc}(G)$ but currently, upper and lower bounds for leading iMHF candidates differ by several orders of magnitude. Blocki and Zhou recently showed that it is NP-Hard to compute $\mathsf{cc}(G)$, but their techniques do not even rule out an efficient $(1+\varepsilon)$-approximation algorithm for any constant $\varepsilon > 0$. We show that for *any* constant $c > 0$, it is Unique Games hard to approximate $\mathsf{cc}(G)$ to within a factor of c.

Along the way, we show the hardness of approximation of the DAG Vertex Deletion problem on DAGs of constant indegree. Namely, we show that for any $k, \varepsilon > 0$ and given a DAG G with N nodes and constant indegree, it is Unique Games hard to distinguish between the case that G is (e_1, d_1)-reducible with $e_1 = N^{1/(1+2\varepsilon)}/k$ and $d_1 = kN^{2\varepsilon/(1+2\varepsilon)}$, and the case that G is (e_2, d_2)-depth-robust with $e_2 = (1-\varepsilon)ke_1$ and $d_2 = 0.9N^{(1+\varepsilon)/(1+2\varepsilon)}$, which may be of independent interest. Our result generalizes a result of Svensson who proved an analogous result for DAGs with indegree $\mathcal{O}(N)$.

2012 ACM Subject Classification Theory of computation → Computational complexity and cryptography; Security and privacy → Hash functions and message authentication codes

Keywords and phrases Cumulative Pebbling Cost, Approximation Algorithm, Unique Games Conjecture, γ-Extreme Depth Robust Graph, Superconcentrator, Memory-Hard Function

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.13

Related Version https://arxiv.org/pdf/1904.08078.pdf

Funding The opinions in this paper are those of the authors and do not necessarily reflect the position of the National Science Foundation.
Jeremiah Blocki: Research supported in part by NSF Award #1755708.
Seunghoon Lee: Research supported in part by NSF Award #1755708 and by the Center for Science of Information at Purdue University (NSF CCF-0939370).

Acknowledgements Part of this work was done while Samson Zhou was a postdoctoral fellow at Indiana University.

1 Introduction

The black pebbling game is a powerful abstraction that allows us to analyze the complexity of functions f_G with a static data-dependency graph G. In particular, a directed acyclic graph (DAG) $G = (V, E)$ can be used to encode data-dependencies between intermediate values produced during computation e.g., if L_v is the v^{th} intermediate value and $L_v := L_j \times L_i$ then the DAG G would include directed edges (i, v) and (j, v) indicating that L_v depends on the previously computed values L_i and L_j. A black pebbling of G is a sequence $P = (P_0, \ldots, P_t) \subseteq V$ of pebbling configurations. Intuitively, a pebbling configuration P_i describes the set of data labels that have been computed and stored in memory at time i. The rules of the pebbling game stipulate that we must have $\mathsf{parents}(v) = \{u \ : \ (u, v) \in E\} \subseteq P_i$ for each newly pebbled node $v \in P_{i+1} \setminus P_i$ i.e., before we can compute a new data value L_v, we must first have the labels of each dependent data value L_u available in memory.

Historically, much of the literature has focused on the sequential black pebbling game where we require that $|P_{i+1} \setminus P_i| \leq 1$ for all round i. In recent years, the parallel black pebbling game has seen renewed interest due to the rapid expansion of parallel computing, e.g., GPUs, FPGAs. In the more general parallel black pebbling game, there is no such restriction on the number of new pebbles in each round, i.e., on a parallel architecture, it is possible to determine L_v for each node $v \in P_{i+1} \setminus P_i$ simultaneously since the dependent data-values are already in memory.

There are several natural ways to measure the cost of a pebbling. The space complexity of a DAG G asks for a legal pebbling $P = (P_0, \ldots, P_t)$ that minimizes the maximum space usage $\max_{i \leq t} |P_i|$ – even if the time t is exponential in the number of nodes N. Space-time complexity asks for a legal pebbling $P = (P_0, \ldots, P_t)$ that minimizes the space-time product $t \times \max_{i \leq t} |P_i|$. Alwen and Serbinenko [7] observed that in the parallel black pebbling game, the space-time of pebbling $G^{\times m}$, m independent copies of a DAG G, does not always scale linearly with m. In particular, for some DAGs G the total space-time cost of pebbling $G^{\times m}$ is roughly equal to the space-time cost of pebbling a single instance of G for $m = \tilde{\mathcal{O}}(\sqrt{N})$!

Alwen and Serbinenko [7] introduced the notion of the cumulative pebbling cost $\mathsf{cc}(G)$ of a DAG G to model the amortized space-time costs in the parallel black pebbling game. Formally, the cumulative pebbling cost of a pebbling P is given by $\mathsf{cc}(P) = \sum_i |P_i|$ and $\mathsf{cc}(G) = \min_P \mathsf{cc}(P)$, where the minimum is taken over all legal (parallel) black pebblings of G. The cumulative pebbling cost is a fundamental metric that is worth studying. It captures the amortized space-time cost of pebbling m copies of G in parallel, i.e., in the limit we have $\mathsf{cc}(G) = \lim_{m \to \infty} \mathtt{ST}(G^{\times m})/m$ where the space-time cost of a pebbling $P = (P_1, \ldots, P_t)$ is $\mathtt{ST}(P) = t \times \max_i |P_i|$ and the notation $G^{\times m}$ denotes a new graph consisting of m disjoint copies of G.

In this paper, we address the following question:

Given a DAG G, can we (approximately) compute $\mathsf{cc}(G)$?

This is a natural question in settings where we want to evaluate the function f_G (with data-dependency DAG G) on many distinct inputs – $\mathsf{cc}(G)$ models the amortized cost of computing f_G. The question is also highly relevant to the cryptanalysis of Data-Independent Memory-Hard Functions (iMHFs). In the context of password hashing we want to find a (constant indegree) DAG G with maximum cumulative pebbling complexity, e.g., to maximize

the cost of a brute-force attacker who wants to evaluate the function f_G on every input in a password cracking dictionary. Thus, given a DAG G one might wish to lower-bound $\mathsf{cc}(G)$ before using G in the design of a memory-hard password hashing algorithm.

Cumulative Pebbling Complexity in Cryptography

In many natural contexts such as password hashing and Proofs of Work, it is desirable to lower bound the amortized space-time cost, e.g., in the random oracle model it is known that the cumulative memory complexity of a (side-channel resistant) iMHF $f_{G,H}$ is $\Omega(\mathsf{cc}(G))$, where $f_{G,H}$ is a labeling function defined in terms of the DAG G and a random oracle H [7]. Thus, in the field of cryptography there has been a lot of interest in designing constant indegree graphs with cumulative pebbling cost $\mathsf{cc}(G)$ as large as possible and in analyzing the pebbling cost $\mathsf{cc}(G)$ of candidate iMHF constructions $f_{G,H}$, e.g., see [2, 5, 3, 6, 4, 14].

From an asymptotic standpoint many of the open questions have been (nearly) resolved. Alwen and Blocki [2] showed that for any DAG G with N nodes and constant indegree we have $\mathsf{cc}(G) = \mathcal{O}\left(N^2 \log\log N/\log N\right)$, while Alwen et al. [5, 4] gave constructions with $\mathsf{cc}(G) = \Omega(N^2/\log N)$. For Argon2i, the winner of the password hashing competition, we have the upper bound $\mathsf{cc}(G) = \mathcal{O}\left(N^{1.767}\right)$ and the lower bound $\mathsf{cc}(G) = \tilde{\Omega}\left(N^{1.75}\right)$ [14].

Most of these upper/lower bounds exploited a relationship between $\mathsf{cc}(G)$ and a combinatorial property called depth-robustness. A DAG $G = (V, E)$ is (e, d)-reducible if we can find a subset $S \subseteq V$ with $|S| \leq e$ such that any directed path P in G of length d contains at least one node in S. On the other hand, if G is not (e, d)-reducible, then we say that G is (e, d)-depth robust. Depth-robustness is known to be both necessary [2] and sufficient [5] for secure iMHFs. In particular, any (e, d)-reducible DAG G with N nodes and indegree $\mathsf{indeg}(G)$ has $\mathsf{cc}(G) \leq \min_{g \geq d}\left(eN + gN \times \mathsf{indeg}(G) + \frac{N^2 d}{g}\right)$ [2] while any (e, d)-depth robust DAG G has $\mathsf{cc}(G) \geq ed$ [5]. The later observation was used to build a constant indegree graph G with $\mathsf{cc}(G) = \Omega(N^2/\log N)$ by showing that the constructed G is $(\Omega(N/\log N), \Omega(N))$-depth robust. The former observation was used to prove that any constant indegree graph has $\mathsf{cc}(G) = \mathcal{O}\left(N^2 \log\log N/\log N\right)$ by exploiting the observation that any such DAG G is $\left(\mathcal{O}\left(N \log\log N/\log N\right), \Omega(N/\log^2 N)\right)$-reducible (simply set $g = \mathcal{O}\left(N \log\log N/\log N\right)$ in the above [2] bound).

Although many of the open questions have been (nearly) resolved from an asymptotic standpoint, from a concrete security standpoint for all practical iMHF candidates G, the best known upper and lower bounds on $\mathsf{cc}(G)$ differ by several orders of magnitude. In fact, Blocki et al. [11] recently found that for practical parameter settings ($N \leq 2^{24}$), Argon2i provides better resistance to known pebbling attacks than DRSample [4] despite the fact that DRSample ($\mathsf{cc}(G) = \Omega(N^2/\log N)$) is asymptotically superior to Argon2i ($\mathsf{cc}(G) = \tilde{\Omega}\left(N^{1.75}\right)$). Of course it is certainly possible that an improved pebbling strategy for Argon2i will reverse this finding tomorrow making it difficult to provide definitive recommendations about which construction is superior in practice.

Given a DAG G, one might try to resolve these questions directly by (approximately) computing $\mathsf{cc}(G)$. Blocki and Zhou [15] previously showed that the problem of computing $\mathsf{cc}(G)$ is NP-Hard. However, their result does not even rule out the existence of a $(1+\varepsilon)$-approximation algorithm for any constant $\varepsilon > 0$.

1.1 Our Contributions

Our main result is the hardness of any constant factor approximation to the cost of graph pebbling even for DAGs with constant indegree[1].

▶ **Theorem 1.** *Given a DAG G with constant indegree, it is Unique Games hard to approximate $\mathsf{cc}(G)$ within any constant factor. (See Theorem 13.)*

Along the way to proving our main result, we show that for any constant $k > 0, \varepsilon > 0$, given a constant indegree graph G, it is Unique Games hard to distinguish between the following two cases: (1) G is (e_1, d_1)-reducible with $e_1 = N^{1/(1+2\varepsilon)}/k$ and $d_1 = kN^{2\varepsilon/(1+2\varepsilon)}$ and (2) G is (e_2, d_2)-depth-robust with $e_2 = (1-\varepsilon)ke_1$ and $d_2 = 0.9N^{(1+\varepsilon)/(1+2\varepsilon)}$. This intermediate result (see Corollary 8) generalizes a result of Svensson [45], who proved an analogous result for DAGs G with arbitrarily large indegree $\mathsf{indeg}(G) = \mathcal{O}(N)$.

Corollary 8 may be of independent interest as depth-robust graphs have found many other applications in cryptography including Proofs of Sequential Work [37], Proofs of Space [24], Proofs of Replication [39, 25] and (relaxed) locally correctable codes for computationally bounded channels [10, 12]. Testing the depth-robustness of a DAG G is especially relevant to the analysis of (tight) Proofs of Space/Replication – several constructions rely on (unproven) conjectures about the concrete depth-robustness of particular DAGs e.g., see [16, 25].

1.2 Technical Ingredients

To prove our result we use three technical ingredients. The first ingredient is a reduction of Svensson [45] that it is Unique Games hard to distinguish between a DAG G (with $\mathsf{indeg}(G) = \mathcal{O}(N)$) that is (e_1, d_1)-reducible or (e_2, d_2)-depth-robust. The second technical ingredient is γ-Extreme Depth-Robust Graphs [6] with bounded indegree. We use γ-Extreme Depth-Robust Graphs to modify the construction of Svensson [45] and show that the same result holds for graphs with much smaller indegree. Finally, we use low depth superconcentrators to boost the lower bound on cc to $\min\{e_2N, d_2N\}/8$ instead of e_2d_2 in the case the graph is (e_2, d_2)-depth robust. We prove that this can be done without significantly increasing the pebbling cost in the case the graph is (e_1, d_1)-reducible.

1.2.1 Technical Ingredient 1

Our first technical ingredient is a result of Svensson [45], who proved that for any constant $k > 0, \varepsilon > 0$, it is Unique Games hard to distinguish between the following two cases (1) G is (e_1, d_1)-reducible with $e_1 = N/k$ and $d_1 = k$, or (2) G is (e_2, d_2)-depth robust with $e_2 = N(1-1/k)$ and $d_2 = \Omega(N^{1-\varepsilon})$. To prove this, Svensson gave a reduction that transforms from any instance of Unique Games $\mathcal{U}$ to a directed acyclic graph $G_{\mathcal{U}}$ on N nodes such that $G_{\mathcal{U}}$ is (e_1, d_1)-reducible for $e_1 \approx N/k$ and $d = k$ if $\mathcal{U}$ is satisfiable. Otherwise, if $\mathcal{U}$ is unsatisfiable, it can be shown that $G_{\mathcal{U}}$ is (e_2, d_2)-depth robust. This is a potentially useful starting point because the pebbling complexity of a graph $G_{\mathcal{U}}$ is closely related to its depth-robustness. In particular, in the second case, a result of Alwen et al. [5] establishes that $\mathsf{cc}(G_{\mathcal{U}}) \geq e_2d_2$ and in the first case, a result of Alwen and Blocki shows that $\mathsf{cc}(G_{\mathcal{U}}) \leq \min_{g \geq d_1} \left(e_1N + gN \times \mathsf{indeg}(G_{\mathcal{U}}) + \frac{N^2 d_1}{g} \right)$ [2].

[1] Each node v in a data-dependency DAG G model an atomic unit of computation. Thus, in practice we expect G to have indegree 2 or 3. If $L_v = g(L_{v_1}, \ldots, L_{v_k})$ is a function of $k \gg 2$ previously computed values $L_{v_1}, \ldots, L_{v_k}$ then we would have generated several additional intermediate data-values while evaluating $g(\cdot)$. These data-values should have been included as nodes in G which is supposed to have a node for every intermediate data-value.

Challenges of Applying Svensson's Construction

While the pebbling complexity of $G_{\mathcal{U}}$ is related to depth-robustness, there is still a vast gap between the upper/lower bounds. In particular, in Svensson's construction we have $\mathsf{indeg}(G_{\mathcal{U}}) = \mathcal{O}(N)$, so the $gN \times \mathsf{indeg}(G_{\mathcal{U}})$ term could be as large as $gN^2 \gg e_2 d_2$. Thus, we would need to be able to reduce the indegree significantly to obtain a gap between $\mathsf{cc}(G_{\mathcal{U}})$ in the two cases. (In fact, we can show the that pebbling cost is exactly $\mathsf{cc}(G_{\mathcal{U}}) = \frac{N(L+1)}{2}$ independent of the Unique Games instance $\mathcal{U}$ – see Lemma 17 in the appendix.) We remark that a naïve attempt to reduce indegree in Svensson's construction $G_{\mathcal{U}}$ by replacing every node v (as in [5]) with a path of length $N + \mathsf{indeg}(v)$ would result in a constant indegree graph $G'_{\mathcal{U}}$ with $N' \approx 2N^2$ nodes that will not be useful for our purposes. The new graph $G'_{\mathcal{U}}$ would be (e_1, d_1)-reducible in the first case with $e_1 = N/k = \mathcal{O}\left(\sqrt{N'}/k\right)$ and $d_1 = 2kN = \mathcal{O}\left(\sqrt{N'}k\right)$. In the second case, the DAG $G'_{\mathcal{U}}$ would be (e_2, d_2)-depth robust with $e_2 \approx ke_1$ and $d_2 = \mathcal{O}\left(N'^{1-\varepsilon/2}\right)$. We would now have $\mathsf{cc}(G'_{\mathcal{U}}) \leq \min_{g \geq d_1}\left(e_1 N' + 2gN' + \frac{N'^2 d_1}{g}\right) = \omega(e_1 N')$ for our upper bound while the lower bound is at most $e_2 d_2 \approx ke_1 N'^{1-\varepsilon/2}$. At the end of the day, the graph $G_{\mathcal{U}}$ is still quite far from what we need.

1.2.2 Technical Ingredient 2: γ-Extreme Depth-Robust Graphs

It does not seem to be possible to obtain a suitable graph $G_{\mathcal{U}}$ by applying indegree reduction techniques to Svensson's Construction in a black-box manner. Instead, we open up the black-box and show how to reduce the indegree using a recent technical result of Alwen et al. [6]. A DAG $G_{\gamma,N}$ on N nodes is said to be γ-extreme depth-robust if it is (e, d)-depth robust for any $e, d > 0$ such that $e + d \leq (1-\gamma)N$. Alwen et al. [6] showed that for any constant $\gamma > 0$, there exists a family $\{G_{\gamma,N}\}_{N=1}^{\infty}$ of γ-extreme depth robust DAGs with maximum indegree $\mathcal{O}(\log N)$. While Alwen et al. [6] were not focused on outdegree, it is not too difficult to see that their construction yields a single family of DAGs with maximum indegree and outdegree $\mathcal{O}(\log N)$.

In Svensson's construction, the DAG $G_{\mathcal{U}}$ is partitioned into $L = N^{1-\varepsilon}$ symmetric layers i.e., if u_ℓ (the copy of node u in layer ℓ_1) is connected to v_{ℓ_2} (the copy of node v in layer $\ell_2 > \ell_1$) then for *any* layers $i < j \leq L$, the directed edge (u_i, v_j) exists. The fact that this edge is "copied" $\mathcal{O}\left(L^2\right)$ times for every pair of layers $i < j$ significantly increases the indegree. However, Svensson's argument that $G_{\mathcal{U}}$ is depth-robust in the second case relies on the existence of each of these edges. To reduce the indegree we start with a γ-extreme depth robust DAG $G_{\gamma,L}$ on L nodes and only keep edges between nodes u_i and v_j in layers i and j if there is a path of length ≤ 2 between nodes i and j in $G_{\gamma,L}$. The new graph can also be shown to have degree at most $\mathcal{O}\left(\mathsf{indeg}(G_L) \times \mathsf{outdeg}(G_L) \times N/L\right) = \mathcal{O}\left(N^{\varepsilon} \log^2 N\right)$. Despite the fact that the indegree is vastly reduced, we are still able to modify Svensson's argument to prove that (for a suitable constant $\gamma > 0$) our new graph is still (e_2, d_2)-depth robust with $e_2 \approx ke_1$ and $d_2 = \mathcal{O}\left(N^{1-\varepsilon}\right)$ – note that the new graph is clearly still (e_1, d_1)-reducible if $\mathcal{U}$ is satisfiable since we only remove edges from Svensson's construction.

We can then apply the generic black-box indegree reduction of [5] to reduce the indegree to 2 by replacing every node with a path of length $N^{2\varepsilon}$. This established our first technical result that even for constant indegree DAGs, it is Unique Games hard to distinguish between the following two cases: (1) G is (e_1, d_1)-reducible with $e_1 = N^{1/(1+2\varepsilon)}/k$ and $d_1 = kN^{2\varepsilon/(1+2\varepsilon)}$, and (2) G is (e_2, d_2)-depth-robust with $e_2 = (1-\varepsilon)ke_1$ and $d_2 = 0.9N^{(1+\varepsilon)/(1+2\varepsilon)}$.

1.2.3 Technical Ingredient 3: Superconcentrators

Although indegree reduction is a crucial step toward showing hardness of approximation for graph pebbling complexity, we still cannot apply known results that relate (e_1, d_1)-reducibility and (e_2, d_2)-depth robustness to pebbling complexity, since there is still no gap between the pebbling complexity of the two cases. In particular, we are always stuck with the $e_1 N$ term in the upper bound of [2] which is *already* much larger than the lower bound $e_2 d_2$ from [6]. To overcome this result we rely on superconcentrators. A *superconcentrator* is a graph that connects N input nodes to N output nodes so that any subset of k inputs and k outputs are connected by k vertex disjoint paths. Moreover, the total number of edges in the graph should be $\mathcal{O}(N)$.

Blocki et al. [11] recently proved that G', the *superconcentrator overlay* of an (e, d)-depth robust graph, has pebbling cost $\mathsf{cc}(G') \geq \max\{eN, dN\}/8$, which is a significant improvement on the lower bound $\mathsf{cc}(G') \geq ed$ when $e = o(N)$ and $d = o(N)$. This allows us to increase the lower-bound in case 2, but we need to be careful that we do not significantly increase the pebbling cost in case 1. To do this we rely on the existence of superconcentrators with depth $\mathcal{O}(\log N)$ [40] and we give a significantly improved pebbling attack on the superconcentrator overlay DAG G' in case 1 when the original graph is (e_1, d_1)-reducible. With the improved pebbling attack, we are able to show that $\mathsf{cc}(G) \geq e_1 kN/16$ in case 2 and that $\mathsf{cc}(G) \leq 16 e_1 N$ in case 1. Since k is an arbitrary constant, this implies that it is Unique Games hard to approximate $\mathsf{cc}(G)$ to within any constant factor $c > 0$.

2 Related Work

Pebbling games have found a number of applications under various formulations and models (see the survey [38] for a more thorough review). The sequential black pebbling game was introduced by Hewitt and Paterson [29], and by Cook [19] and has been particularly useful in exploring space/time trade-offs for various problems like matrix multiplication [47], fast fourier transformations [43, 47], integer multiplication [46] and many others [17, 44]. In cryptography it has been used to construct/analyze Proofs of Space [24, 41], Proofs of Work [23, 37] and Memory-Hard Functions [26]. Alwen and Serbinenko [7] argued that the parallel version of the black pebbling game was more appropriate for Memory-Hard Functions and they proved that any iMHF attacker in the parallel random oracle model corresponds to a pebbling strategy with equivalent cumulative memory cost.

The space cost of the black pebbling game is defined to be $\max_i |P_i|$, which intuitively corresponds to minimizing the maximum space required during computation of the associated function. Gilbert et al. [27] studied the space-complexity of the black-pebbling game and showed that this problem is PSPACE-Complete by reducing from the truly quantified boolean formula (TQBF) problem. In our case, the decision problem is $\mathsf{cc}(G) \leq k$ is in NP because the optimal pebbling strategy cannot last for more than N^2 steps since *any* graph with N nodes has $\mathsf{cc}(G) \leq N^2$.

Red-Blue Pebbling

Given a DAG $G = (V, E)$, the goal of the red-blue pebbling game [30] is to place pebbles on all sink nodes of G (not necessarily simultaneously) from an empty starting configuration. Intuitively, red pebbles represent values in cache and blue pebbles represent values stored in memory. Blue pebbles must be converted to red pebbles (e.g., loaded into cache) before

they can be used in computation, but there is a limit m (cache-size) on the number of red-pebbles that can be used. Red-blue pebbling games have been used to study memory-bound functions [22] (functions that incur many expensive cache-misses [1]).

Ren and Devadas introduced the notion of bandwidth hard functions and used the red-blue pebbling game to analyze the energy cost of a memory hard function [42]. In their model, red-moves (representing computation performed using data in cache) have a smaller cost c_r than blue-moves c_b (representing data movements to/from memory) and a DAG G on N nodes is said to be bandwidth hard if any red-blue pebbling has cost $\Omega(N \cdot c_b)$. Ren and Devadas showed that the bit reversal graph [35], which forms the core of iMHF candidate Catena-BRG [26], is maximally bandwidth hard. Subsequently, Blocki et al. [13] gave a pebbling reduction showing that any attacker random oracle model (pROM) can indeed be viewed as a red-blue pebbling with equivalent cost. They also show that it is NP-Hard to compute the minimum cost red-blue pebbling of a DAG G i.e., the decision problem "is the red-blue pebbling cost $\leq k$?" is NP-Complete (A result of Demaine and Liu [20, 36] implies that the problem is PSPACE-Hard to compute the red-blue pebbling cost when $c_r = 0$ i.e., computation is free). In general, the red-blue cost of G is always lower bounded by $c_r N$ and upper-bounded by $2c_b N + c_r N$. The question of a more efficient c-approximation algorithm for $c = o(c_b/c_r)$ remains open.

Unique Games

Recently, the Unique Games Conjecture and related conjectures have received a lot of attention for their applications in proving hardness of approximation. Khot et al. [32] showed that the Goemans-Williamson approximation algorithm for Max-Cut [28] is optimal, assuming the Unique Games Conjecture. Khot and Regev [34] showed that Minimum Vertex Cover problem is Unique Games hard to solve within a factor of $2 - \varepsilon$, which is nearly tight from the guarantee that a simple greedy algorithm gives. The Unique Games Conjecture also leads to tighter approximation hardness for other problems including Max 2-SAT [32] and Betweenness [18]. Although a previous stronger version of the conjecture asked whether Unique Games instances required exponential time algorithms in the worst case, Arora et al. [8] gave a subexponential time algorithm for Unique Games. Lately, focus has also been drawn toward studying the related Label Cover Problem, such as the 2-Prover-1-Round Games, i.e. the 2-to-1 Games Conjecture [21] and the 2-to-2 Games Conjecture [33].

3 Preliminaries

We use the notation $[N]$ to denote the set $\{0, 1, \ldots, N-1\}$. Given a directed acyclic graph $G = (V, E)$ and a node $v \in V$, we use $\mathsf{parents}(v) = \{u : (u, v) \in E\}$ (resp. $\mathsf{children}(v) = \{u : (v, u) \in E\}$) to denote the parents (resp. children) of node v. We use $\mathsf{indeg}(v) = |\mathsf{parents}(v)|$ (resp. $\mathsf{outdeg}(v) = |\mathsf{children}(v)|$) to denote the number of incoming (resp. outgoing) edges into (resp. out of) the vertex v. We also define $\mathsf{indeg}(G) = \max_{v \in V} \mathsf{indeg}(v)$ and $\mathsf{outdeg}(G) = \max_{v \in V} \mathsf{outdeg}(v)$. Given a set $S \subseteq V$ of nodes, we use $G - S$ to refer to the graph obtained by deleting all nodes in S and all edges incident to S. We also use $G[S] = G - (V \setminus S)$ to refer to the subgraph induced by the nodes S, i.e., deleting every other node in $V \setminus S$. Given a node $v \notin S$, we use $\mathsf{depth}(v, G - S)$ to refer to the longest directed path in $G - S$ ending at node v and we use $\mathsf{depth}(G - S) = \max_{v \notin S} \mathsf{depth}(v, G - S)$ to refer to the longest directed path in $G - S$. Given a subset B, we will also use $\mathsf{depth}_B(v, G - S)$ to refer to the maximum number of nodes in the set B contained in any directed path in $G - S$ that ends at node v. We define $\mathsf{depth}_B(G - S) = \max_{v \notin S} \mathsf{depth}_B(v, G - S)$ analogously.

▶ **Definition 2** (Unique Games). *An instance* $\mathcal{U} = (G = (V, W, E), [R], \{\pi_{v,w}\}_{v,w})$ *of Unique Games consists of a regular bipartite graph* $G(V, W, E)$ *and a set* $[R]$ *of labels. Each edge* $(v, w) \in E$ *has a constraint given by a permutation* $\pi_{v,w} : [R] \to [R]$*. The goal is to output a labeling* $\rho : (V \cup W) \to [R]$ *that maximizes the number of satisfied edges, where an edge is satisfied if* $\rho(v) = \pi_{v,w}(\rho(w))$*.*

▶ **Conjecture 3** (Unique Games Conjecture, [31]). *For any constants* $\alpha, \beta > 0$*, there exists a sufficiently large integer* R *(as a function of* α, β*) such that for Unique Games instances with label set* $[R]$*, no polynomial time algorithm can distinguish whether: (1) the maximum fraction of satisfied edges of any labeling is at least* $1 - \alpha$*, or (2) the maximum fraction of satisfied edges of any labeling is less than* β*.*

Graph Pebbling

The goal of the (black) pebbling game is to place pebbles on all sink nodes of some input directed acyclic graph (DAG) $G = (V, E)$. The game proceeds in rounds, and each round i consists of a number of pebbles $P_i \subseteq V$ placed on a subset of the vertices. Initially, the graph is unpebbled, $P_0 = \emptyset$, and in each round $i \geq 1$, we may place a pebble on $v \in P_i$ if either all parents of v contained pebbles in the previous round ($\mathsf{parents}(v) \subseteq P_{i-1}$) or if v already contained a pebble in the previous round ($v \in P_{i-1}$). In the sequential pebbling game, at most one new pebble can be placed on the graph in any round (i.e., $|P_i \backslash P_{i-1}| \leq 1$), but this restriction does not apply in the parallel pebbling game.

We use $\mathcal{P}_G^{\parallel}$ to denote the set of all valid parallel pebblings of G. The *cumulative cost* of a pebbling $P = (P_1, \ldots, P_t) \in \mathcal{P}_G^{\parallel}$ is the quantity $\mathsf{cc}(P) := |P_1| + \ldots + |P_t|$ that represents the sum of the number of pebbles on the graph during every round. The (parallel) *cumulative pebbling cost* of G, denoted $\mathsf{cc}(G) := \min_{P \in \mathcal{P}_G^{\parallel}} \mathsf{cc}(P)$, is the cumulative cost of the best legal pebbling of G.

A DAG G is (e, d)-*reducible* if there exists a subset $S \subseteq V$ of size $|S| \leq e$ such that $\mathsf{depth}(G - S) < d$. That is, there are no directed paths containing d vertices remaining, once the vertices in the set S are removed from G. If G is not (e, d)-reducible, we say that it is (e, d)-depth robust.

4 Reduction

Svensson [45] showed that for any constant $k, \epsilon > 0$ it is Unique Games hard to distinguish between whether a DAG G is (e_1, d_1)-reducible for $e_1 = N/k$ and $d_1 = k$ or G is (e_2, d_2)-depth robust with $e_2 = N(1 - 1/k)$ and $d_2 = \Omega(N^{1-\varepsilon})$. To prove this, Svensson showed how to transform a Unique Games instance $\mathcal{U} = (G = (V, W, E), [R], \{\pi_{v,w}\}_{v,w})$ into a graph $G_{\mathcal{U}}$ such that $G_{\mathcal{U}}$ is (e_1, d_1)-reducible if it is possible to satisfy $1 - \alpha$ fraction of the edges and $G_{\mathcal{U}}$ is (e_2, d_2)-depth robust if it is not possible to satisfy β-fraction of the edges. To obtain inapproximability results for cc, it is crucial to substantially reduce the indegree of this construction.

4.1 Review of Svensson's Construction

To construct $G_{\mathcal{U}}$, Svensson first constructs a layered bipartite DAG $\hat{G}_{\mathcal{U}}$, which encodes the unique games instance $\mathcal{U}$ and later transforms $\hat{G}_{\mathcal{U}}$ into the required DAG $G_{\mathcal{U}}$. For completeness, we provide a full description of the DAG $\hat{G}_{\mathcal{U}}$ in the appendix. We will focus our discussion here on the essential properties of the DAG $\hat{G}_{\mathcal{U}}$.

The graph $\hat{G}_{\mathcal{U}}$ has a number of *bit-vertices* B partitioned into *bit-layers* $B = B_0 \cup \ldots \cup B_L$, where B_i is the set of bit-vertices in bit-layer i. Each B_i can be partitioned into sets $B_{i,w}$ for $w \in W$. Similarly, $\hat{G}_{\mathcal{U}}$ has a number of *test-vertices* T partitioned into *test-layers* $T = T_0 \cup \ldots \cup T_{L-1}$, where T_i is the set of test-vertices in test-layer i. Outgoing edges for test-layer T_ℓ must be directed into a bit vertex in layer $B_{\ell'}$ with $\ell' > \ell$. Similarly, outgoing edges from B_ℓ must be directed into a test vertex in layer $T_{\ell'}$ with $\ell' \geq \ell$. Each T_i can be partitioned into sets $T_{i,v}$ for $v \in V$. The constraints in our Unique Games instance $\mathcal{U}$ are encoded as edges between the bit vertices and test vertices. We use $N = |T|$ to denote the total number of test nodes and remark that the parameter L is set such that $L \geq N^{1-\epsilon}$.

$\hat{G}_{\mathcal{U}}$ also displays symmetry between the layers in the sense that $B_\ell = \{b_1^\ell, \ldots, b_m^\ell\}$ and $T_\ell = \{t_1^\ell, \ldots, t_p^\ell\}$, so that the number of bit-vertices in each bit-layer is the same and the number of test-vertices in each test-layer is the same.

Symmetry

In Svensson's construction, we have exactly m bit vertices in every layer $B_\ell = \{b_1^\ell, \ldots, b_m^\ell\}$ and exactly p test vertices in every layer $T_\ell = \{t_1^\ell, \ldots, t_p^\ell\}$. The edges between B_ℓ and T_ℓ (resp. T_ℓ and $B_{\ell+1}$) encode the edge constraints in the unique games instance $\mathcal{U}$. Furthermore, the construction is symmetric so that directed edge (b_i^ℓ, t_j^ℓ) exists if and only if for *every* $\ell' \geq \ell$ the edge $(b_i^\ell, t_j^{\ell'})$ exists. Thus for any $\ell' \geq \ell$, the edges between B_ℓ and $T_{\ell'}$ encode the constraints in $\mathcal{U}$. Similarly, the directed edge $(t_j^\ell, b_i^{\ell+1})$ exists if and only if any $\ell' > \ell$ the edge $(t_j^\ell, b_i^{\ell'})$ exists. We remark that this means that the indegree of the graph $\hat{G}_{\mathcal{U}}$ is at least L (and can be as large as $\Omega(N)$ in general).

Robustness of $\hat{G}_{\mathcal{U}}$

Svensson argues that if it is possible to satisfy a $1-\alpha$ fraction of the constraints in $\mathcal{U}$, then there exists a subset $S \subseteq T$ of at most $|S| \leq e_1$ test-vertices such that $\mathsf{depth}_B(\hat{G}_{\mathcal{U}} - S) \leq d_1$. Similarly, if it is not possible to satisfy a β-fraction of the constraints, then for any subset $S \subseteq T$ of at most $|S| \leq e_2$ test-vertices, we have $\mathsf{depth}_B(\hat{G}_{\mathcal{U}} - S) \geq d_2$. This does not directly show that $\hat{G}_{\mathcal{U}}$ is depth-robust since we are not allowed to delete bit-vertices. However, one can easily transform $\hat{G}_{\mathcal{U}}$ into a graph $G_{\mathcal{U}}$ on the $N = |T|$ test nodes such that $G_{\mathcal{U}}$ is (e, d)-depth robust if and only if for all subsets $S \subseteq T$ of $|S| \leq e$ test vertices in $\hat{G}_{\mathcal{U}}$, we have $\mathsf{depth}_B(\hat{G}_{\mathcal{U}} - S) \geq d$. It is worth mentioning that we can view these guarantees as a form of *weighted* depth-robustness where all test-vertices have weight 1 and all bit-vertices have weight ∞, i.e., if $1-\alpha$ fraction of the constraints in $\mathcal{U}$, then we can find a subset S of nodes with weight $\mathsf{weight}(S) \leq e_1$ such that $\mathsf{depth}(\hat{G}_{\mathcal{U}} - S) \leq d_1$, and if it is not possible to satisfy β-fraction of the constraints, then for any subset S with $\mathsf{weight}(S) \leq e_2$ we have $\mathsf{depth}(\hat{G}_{\mathcal{U}} - S) \geq d_1$.

Graph Coloring and Robustness

An equivalent way to view the problem of *weighted* reducibility (resp. depth-robustness) is in terms of graph coloring. This view is central to Svensson's argument. In particular, if we can find a depth reducing set $S \subseteq T$ of size $|S| \leq e$ such that $\mathsf{depth}_B(\hat{G}_{\mathcal{U}} - S) \leq d$, then we can define a d-coloring $\chi : B \to [d]$ of each of the bit-vertices such that the coloring χ is consistent with every remaining test node $v \in T \setminus S$. Here, consistency means that $\max_{b \in \mathsf{parents}(v)} \chi(b) < \min_{b \in \mathsf{children}(v)} \chi(b)$. In fact, it is not too difficult to see that there is a subset $S \subseteq T$ of $|S| \leq e$ test-vertices such that $\mathsf{depth}_B(\hat{G}_{\mathcal{U}} - S) \leq d$ if and only if there

is a d-coloring χ such that $\left|\{v \ : \ \max_{b\in\mathsf{parents}(v)}\chi(b) \geq \min_{b\in\mathsf{children}(v)}\chi(b)\}\right| \leq e$, i.e., given a d-coloring χ of the bit vertices, we can simply select $S = \{v \ : \ \max_{b\in\mathsf{parents}(v)}\chi(b) \geq \min_{b\in\mathsf{children}(v)}\chi(b)\}$ of inconsistent test-vertices and then for every $u \in B$ we can inductively show that $\mathsf{depth}_B(u, \hat{G}_{\mathcal{U}} - S) \leq \chi(u)$.

Brief Overview of Svensson's Proof

Svensson defines $\chi(w,i)$ to denote the largest color that is smaller than the colors of at least $(1-\delta)$ fraction of the bit-vertices in $B_{i,w}$, i.e., $\chi(w,i) = \max\{\text{color } c : \Pr_{b\in B_{i,w}}[\chi(b) \geq c] \geq 1-\delta\}$. Suppose that it is not possible to satisfy a $\beta = \frac{\delta\eta^2}{t^2k^2}$-fraction of the constraints in $\mathcal{U}$ for tunable parameters $t, \eta > 0$ that are part of Svensson's construction. The core piece of Svensson's proof is demonstrating that if the set $S = \{v \ : \ \max_{b\in\mathsf{parents}(v)}\chi(b) \geq \min_{b\in\mathsf{children}(v)}\chi(b)\}$ of inconsistent test-vertices has size $|S| \leq (1-32\delta)|T|$, then we can find some $w \in W$ such that $\Pr[\chi(w,i) > \chi(w,i+1)] \geq 32\delta^2$ for some constant c that depends on various parameters of the construction. Svensson notes that by symmetry of the construction $\hat{G}_{\mathcal{U}}$, we can assume without loss of generality that $\chi(i,w) \leq \chi(i+1,w)$ for any $i \leq L$. We remark that this will *not* necessarily be the case after our indegree reduction step. Thus, it immediately follows that χ uses more than $32|T|\delta^2$ colors, i.e., $\mathsf{depth}_B(u, \hat{G}_{\mathcal{U}} - S) \geq 32|T|\delta^2$.

4.2 Reducing the Indegree

As previously discussed, Svensson's construction has indegree that is too large for the purposes of bounding the pebbling complexity by finding a gap between known results implied by (e_1, d_1)-reducibility and (e_2, d_2)-depth robustness. To perform indegree reduction, we use a γ-extreme depth-robust graph $G_{\gamma,L+1}$ with $L+1$ vertices in a procedure $\mathsf{Sparsify}_{G_{\gamma,L+1}}(\hat{G}_{\mathcal{U}})$ to decide which edges in $\hat{G}_{\mathcal{U}}$ to keep and which edges to discard. Intuitively, we will keep the edge $(b_\ell, t_{\ell'})$ from a bit vertex $b_\ell \in B_\ell$ on layer $\ell \leq \ell'$ to test vertex $t_{\ell'} \in T_{\ell'}$ on layer ℓ' if and only if $\ell = \ell'$ or $G_{\gamma,L+1}$ contains the edge (ℓ, ℓ'). Similarly, we will keep the edge $(t_\ell, b_{\ell'})$ from a test vertex $t_\ell \in T_\ell$ on layer $\ell < \ell'$ to bit vertex $b_{\ell'} \in B_{\ell'}$ on layer ℓ' if and only if $(\ell, \ell') \in G_{\gamma,L+1}$. The result is a new DAG $\mathsf{Sparsify}_{G_{\gamma,L+1}}(\hat{G}_{\mathcal{U}})$ with substantially smaller indegree and outdegree $\mathcal{O}\left(N^\varepsilon \log^2 N\right)$ instead of $\mathcal{O}(N)$.

Transformation $\mathsf{Sparsify}_{G_{\gamma,L+1}}(\hat{G}_{\mathcal{U}})$

Input: An instance $\hat{G}_{\mathcal{U}} = (V, E)$ of the Svensson's construction, whose vertices are partitioned into $L+1$ bit-layers $B_0, \ldots, B_L$ and L test-layers $T_0, \ldots, T_{L-1}$, a γ-extreme depth robust graph $G_{\gamma,L+1} = (V_\gamma = [L+1], E_\gamma)$.

1. Let $G' = (V, E)$ be a copy of $\hat{G}_{\mathcal{U}}$.
2. If $e = (b, t)$ is an edge in G, where $b \in B_i$ and $t \in T_j$, delete e from G' if $i \neq j$ and $(i, j) \notin E_\gamma$.
3. If $e = (t, b)$ is an edge in G, where $b \in B_i$ and $t \in T_j$, delete e from G' if $(j, i) \notin E_\gamma$.

Output: G'

We remark that we only delete edges from $\hat{G}_{\mathcal{U}}$. Thus for any subset $S \subseteq T$ of $|S| \leq e_1$ test vertices, we have $\mathsf{depth}_B(\hat{G}_{\mathcal{U}} - S) \geq \mathsf{depth}_B(\mathsf{Sparsify}_{G_{\gamma,L+1}}(\hat{G}_{\mathcal{U}}) - S)$. Hence, $\mathsf{Sparsify}_{G_{\gamma,L+1}}(\hat{G}_{\mathcal{U}})$ is certainly not more depth-robust than $\hat{G}_{\mathcal{U}}$. The harder argument is showing that the graph $\mathsf{Sparsify}_{G_{\gamma,L+1}}(\hat{G}_{\mathcal{U}})$ is still depth-robust when our unique games instance $\mathcal{U}$ has no assignment satisfying a β fraction of the edges.

Assuming that the Unique Games instance is unsatisfiable, Lemma 4 implies that as long as $32\delta^2|T|$ test-vertices are consistent with our coloring, we can find some $w \in W$ such that w is *locally consistent* on at least $32\delta^2 L$ layers, i.e., w is locally consistent on layer ℓ if $\forall \ell' > \ell$ we have $\chi(w,\ell') > \chi(w,\ell)$.

The parameters η, t in Lemma 4 are tunable parameters of the reduction.

▶ **Lemma 4.** *Let χ be any coloring of $\mathsf{Sparsify}_{G_{\gamma,L+1}}(\hat{G}_{\mathcal{U}})$. If the Unique Games instance has no labeling that satisfies a fraction $\frac{\delta\eta^2}{t^2k^2}$ of the constraints and at least $32\delta^2|T|$ test vertices are consistent with χ, then there exists $w \in W$ with*

$$\Pr_{\ell\in[L]}[\chi(w,\ell') > \chi(w,\ell) \text{ for all } \ell' > \ell \text{ with } (\ell,\ell') \in E_\gamma] \geq 32\delta^2.$$

We remark that the proof of Lemma 4 closely follows Svensson's argument with a few modifications. While the modifications are relatively minor, specifying these modifications requires a complete description of Svensson's construction. We refer an interested reader to Appendix B for details and for the formal proof of Lemma 4.

Lemma 5 now shows that $\mathsf{Sparsify}_{G_{\gamma,L+1}}(\hat{G}_{\mathcal{U}})$ is still depth-robust in case 2. The main challenge is that after we sparsify the graph, we can no longer assume that $\chi(w,\ell') > \chi(w,\ell)$ for all $\ell' > \ell$ without loss of generality, e.g., even if there are many i's for which $\chi(w,i+1) > \chi(w,i)$ we could have a sequence like $\chi(w,1) = 1, \chi(w,2) = 2, \chi(w,3) = 2, \chi(w,4) = 2, \chi(w,5) = 1, \chi(w,6) = 2, \ldots$. We rely on the fact that $G_{\gamma,L+1}$ is extremely depth-robust to show that for any sufficiently large subset $\mathtt{LC} \subseteq [L]$ of layers for which w is *locally consistent*, there must be a subsequence $\mathtt{P}_w \subseteq \mathtt{LC}$ of length $|\mathtt{P}_w| \geq |\mathtt{LC}| - \gamma L$ over which $\chi(w,\cdot)$ is strictly increasing.

▶ **Lemma 5.** *If the Unique Games instance has no labeling that satisfies a fraction $\frac{\delta\eta^2}{t^2k^2}$ of the constraints and $\gamma \leq 31\delta^2$, then for every set $S \subseteq T$ of at most $|S| \leq (1-32\delta)|T|$ test-vertices the graph $G' = \mathsf{Sparsify}_{G_{\gamma,L+1}}(\hat{G}_{\mathcal{U}})$ has a path of length $\delta^2 L$.*

Proof. Suppose the Unique Games instance has no labeling that satisfies a fraction $\frac{\delta\eta^2}{t^2k^2}$ of the constraints. Let S contain at most $(1-32\delta)|T|$ and define the labeling $\chi(b) = \mathsf{depth}_B(b, G'-S)$. By Lemma 4, there exists $w \in W$ with

$$\Pr_{\ell\in[L]}[\chi(w,\ell') > \chi(w,\ell) \text{ for all } \ell' > \ell \text{ with } (\ell,\ell') \in E_\gamma] \geq 32\delta^2.$$

Let $\mathtt{LC} \subseteq [L]$ denote the subset of layers over which w is locally consistent. We remark that each $\ell \in \mathtt{LC}$ corresponds to a node in $G_{\gamma,L+1}$ and that $G_{\gamma,L+1}[\mathtt{LC}]$ contains a path $\mathtt{P}_w = (\ell_1, \ldots, \ell_k)$ of length $k \geq |\mathtt{LC}| - \gamma L \geq (32\delta^2 - \gamma)L$. We also note that $\chi(w,\ell_{i+1}) > \chi(w,\ell_i)$ for each $i < k$. Hence, $\chi(w,\ell_k) \geq k$, which means that $\mathsf{depth}_B(b, G'-S) \geq (32\delta^2-\gamma)L \geq \delta^2 L$ as long as $\gamma \leq 31\delta^2$. ◀

Theorem 6, our main technical result in this section, states that it is Unique Games hard to distinguish between (e_1,d_1)-reducible and (e_2,d_2)-depth robust graphs even for a DAG G with N vertices and $\mathsf{indeg}(G) = \mathcal{O}\left(N^\varepsilon \log^2 N\right)$.

▶ **Theorem 6.** *For any integer $k \geq 2$ and constant $\varepsilon > 0$, given a DAG G with N vertices and $\mathsf{indeg}(G) = \mathcal{O}\left(N^\varepsilon \log^2 N\right)$, it is Unique Games hard to distinguish between the following cases: (1) (Completeness): G is $\left(\left(\frac{1-\varepsilon}{k}\right)N, k\right)$-reducible, and (2) (Soundness): G is $\left((1-\varepsilon)N, N^{1-\varepsilon}\right)$-depth robust.*

Proof. Recall that we can transform $G' = \mathsf{Sparsify}_{G_{\gamma,L+1}}(\hat{G}_{\mathcal{U}})$ into an unweighted graph G over the $N = |T|$ test-vertices. In particular, we add the edge (u, v) to G if and only if there was a path of length 2 from u to v in G'. We remark that the indegree is $\mathsf{indeg}(G) = \mathcal{O}\left(N^{\varepsilon} \log^2 N\right)$ and that for any $S \subseteq T$, we have $\mathsf{depth}(G - S) \leq \mathsf{depth}_B(G' - S) \leq \mathsf{depth}(G - S) + 1$. Completeness now follows immediately from Theorem 20 under the observation that we only removed edges from Svensson's construction. Soundness follows immediately from Theorem 20 and Lemma 5. ◀

Obtaining DAGs with Constant Degree

We can now apply a second indegree reduction procedure $\mathsf{IDR}(G, \gamma)$. For a graph $G = (V, E)$, the procedure $\mathsf{IDR}(G, \gamma)$ replaces each node $v \in V$ with a path $P_v = v_1, \ldots, v_{\delta+\gamma}$, where δ is the indegree of G. For each edge $(u, v) \in E$, we add the edge $(u_{\delta+\gamma}, v_j)$ whenever (u, v) is the j^{th} incoming edge of v, according to some fixed ordering. [5] give parameters e_2 and d_2 so that $\mathsf{IDR}(G, \gamma)$ is (e_2, d_2)-depth robust if G is (e, d)-depth robust. For a formal description of $\mathsf{IDR}(G, \gamma)$, see Appendix B. We complete the reduction by giving parameters e_1 and d_1 so that $\mathsf{IDR}(G, \gamma)$ is (e_1, d_1)-reducible if G is (e, d)-reducible.

▶ **Lemma 7.** *There exists a polynomial time procedure* $\mathsf{IDR}(G, \gamma)$ *that takes as input a DAG* G *with* N *vertices and* $\mathsf{indeg}(G) = \delta$ *and outputs a graph* $G' = \mathsf{IDR}(G, \gamma)$ *with* $(\delta + \gamma)N$ *vertices and* $\mathsf{indeg}(G') = 2$*. Moreover, the following properties hold: (1) If* G *is* (e, d)*-reducible, then* $\mathsf{IDR}(G, \gamma)$ *is* $(e, (\delta + \gamma) \cdot d)$*-reducible, and (2) If* G *is* (e, d)*-depth robust, then* $\mathsf{IDR}(G, \gamma)$ *is* $(e, \gamma \cdot d)$*-depth robust.*

▶ **Corollary 8.** *For any integer* $k \geq 2$ *and constant* $\varepsilon > 0$*, given a DAG* G *with* N *vertices and maximum indegree* $\mathsf{indeg}(G) = 2$*, it is Unique Games hard to decide whether* G *is* (e_1, d_1)*-reducible or* (e_2, d_2)*-depth robust for (Completeness):* $e_1 = \frac{1}{k} N^{\frac{1}{1+2\varepsilon}}$ *and* $d_1 = kN^{\frac{2\varepsilon}{1+2\varepsilon}}$*, and (Soundness):* $e_2 = (1 - \varepsilon)N^{\frac{1}{1+2\varepsilon}}$ *and* $d_2 = 0.9N^{\frac{1+\varepsilon}{1+2\varepsilon}}$*.*

5 Putting the Pieces Together

We would now like to apply Theorem 18 and Theorem 19. However, the upper bound on $\mathsf{cc}(G)$ that we obtain from Theorem 18 will not be better than $e_1 N = \frac{1}{k} N^{\frac{2+2\varepsilon}{1+2\varepsilon}}$, while the lower bound we obtain from Theorem 19 is just $(1 - \varepsilon)N^{\frac{2+\varepsilon}{1+2\varepsilon}}$, so we do not get our desirable gap between the upper and lower bounds. We therefore discard Theorem 18 and Theorem 19 altogether and instead apply a graph transformation with explicit bounds on pebbling complexity.

▶ **Definition 9** (Superconcentrator)**.** *A graph* G *with* $\mathcal{O}(N)$ *vertices is called a* superconcentrator *if there exists* N *input vertices, denoted* $\mathsf{input}(G)$*, and* N *output vertices, denoted* $\mathsf{output}(G)$*, such that for all* $S_1 \subseteq \mathsf{input}(G), S_2 \subseteq \mathsf{output}(G)$ *with* $|S_1| = |S_2| = k$*, there are* k *vertex disjoint paths from* S_1 *to* S_2*.*

Pippenger gives a superconcentrator construction with depth $\mathcal{O}(\log N)$.

▶ **Lemma 10** ([40])**.** *There exists a superconcentrator* G *with at most* $42N$ *vertices, containing* N *input vertices and* N *output vertices, such that* $\mathsf{indeg}(G) \leq 16$ *and* $\mathsf{depth}(G) \leq \log(42N)$*.*

Now we define the overlay of a superconcentrator on a graph G (see Figure 1).

▶ **Definition 11** (Superconcentrator Overlay)**.** *Let* $G = (V(G), E(G))$ *be a fixed DAG with* N *vertices and* $G_S = (V(G_S), E(G_S))$ *be a (priori fixed) superconcentrator with* N *input vertices* $\mathsf{input}(G_S) = \{i_1, \cdots, i_N\} \subseteq V(G_S)$ *and* N *output vertices* $\mathsf{output}(G_S) = \{o_1, \cdots, o_N\} \subseteq$

$V(G_S)$. *We call a graph* $G' = (V(G_S), E(G_S) \cup E_I \cup E_O)$ *a* superconcentrator overlay *where* $E_I = \{(i_u, i_v) : (u, v) \in E(G)\}$ *and* $E_O = \{(o_i, o_{i+1}) : 1 \leq i < N\}$ *and denote as* $G' = \mathsf{superconc}(G)$.

We will denote the interior nodes as $\mathsf{interior}(G') = G' \setminus (\mathsf{input}(G') \cup \mathsf{output}(G'))$ where $\mathsf{input}(G') = \mathsf{input}(G_S)$ and $\mathsf{output}(G') = \mathsf{output}(G_S)$. We remark that when using Pippenger's construction of superconcentrators, it is easy to show that $\mathsf{superconc}(G)$ is $\left(e + \frac{N}{d}, 2d + \log(42N)\right)$-reducible whenever G is (e, d)-reducible, which implies that

$$\mathsf{cc}(\mathsf{superconc}(G)) \leq \min_{g \geq d} \left(e + \frac{N}{d}\right) 42N + 2g(42N) + \frac{42N}{g} \left(2d + \log(42N)\right) 42N.$$

For more details, we refer an interested reader to Lemma 24 and Corollary 25 in Appendix D. However, these results are not quite as strong as we would like. By comparison, we have the following lower bound on the pebbling complexity from [11]:

$$\mathsf{cc}(\mathsf{superconc}(G)) \geq \min\left(\frac{eN}{8}, \frac{dN}{8}\right).$$

In Lemma 12 we obtain a *significantly tighter* upper bound on $\mathsf{cc}(\mathsf{superconc}(G))$ with an improved pebbling strategy described at the end of this section.

▶ **Lemma 12.** *Let* G *be an* (e, d)*-reducible graph with* N *vertices with* $\mathsf{indeg}(G) = 2$*. Then* $\mathsf{cc}(\mathsf{superconc}(G)) \leq \min_{g \geq d} \left\{2eN + 4gN + \frac{43dN^2}{g} + \frac{24N^2 \log(42N)}{g} + 42N \log(42N) + N\right\}$.

With the improved attack in Lemma 12, we can tune parameters appropriately to obtain our main result, Theorem 13.

▶ **Theorem 13.** *Given a DAG* G*, it is Unique Games hard to approximate* $\mathsf{cc}(G)$ *within any constant factor.*

Proof. Let $k \geq 2$ be an integer that we shall later fix and similarly, let $\varepsilon > 0$ be a constant that we will later fix. Given a DAG G with N vertices, then it follows by Corollary 8 that it is Unique Games hard to decide whether G is (e_1, d_1)-reducible or (e_2, d_2)-depth robust for $e_1 = \frac{1}{k} N^{\frac{1}{1+2\varepsilon}}$, $d_1 = kN^{\frac{2\varepsilon}{1+2\varepsilon}}$ and $e_2 = (1-\varepsilon)N^{\frac{1}{1+2\varepsilon}}$ and $d_2 = 0.9N^{\frac{1+\varepsilon}{1+2\varepsilon}}$. If G is (e_1, d_1)-reducible, then by Lemma 12, $\mathsf{cc}(\mathsf{superconc}(G)) \leq \min_{g \geq d} \{2e_1 N + 4gN + \frac{43d_1N^2}{g} + \frac{24N^2 \log(42N)}{g} + 42N \log(42N) + N\}$. Observe that $2e_1 N = \frac{2}{k} N^{(2+2\varepsilon)/(1+2\varepsilon)}$, whereas for $g = e_1$ and sufficiently large N, $4gN + \frac{43d_1N^2}{g} + \frac{24N^2 \log(42N)}{g} + 42N \log(42N) + N \leq \frac{5}{k} N^{\frac{2+2\varepsilon}{1+2\varepsilon}}$. Hence for sufficiently large N,

$$\mathsf{cc}(\mathsf{superconc}(G)) \leq \frac{7}{k} N^{\frac{2+2\varepsilon}{1+2\varepsilon}}.$$

On the other hand, if G is (e_2, d_2)-depth robust, then by Lemma 23, $\mathsf{cc}(\mathsf{superconc}(G)) \geq \min\left(\frac{e_2 N}{8}, \frac{d_2 N}{8}\right)$. Specifically,

$$\mathsf{cc}(\mathsf{superconc}(G)) \geq \frac{e_2 N}{8} = \frac{1-\varepsilon}{8} N^{\frac{2+2\varepsilon}{1+2\varepsilon}}.$$

Let $c > 1$ be any constant. Setting $\varepsilon = 0.1$ and $k = \lceil \frac{560}{9} c^2 \rceil$, we get that if G is (e_1, d_1)-reducible, then $\mathsf{cc}(\mathsf{superconc}(G)) \leq \frac{9}{80c^2} N^{\frac{2+2\varepsilon}{1+2\varepsilon}}$ but if G is (e_2, d_2)-reducible, then $\mathsf{cc}(\mathsf{superconc}(G)) \geq \frac{9}{80} N^{\frac{2+2\varepsilon}{1+2\varepsilon}}$. Hence, it is Unique Games hard to approximate $\mathsf{cc}(G)$ with a factor of c. ◀

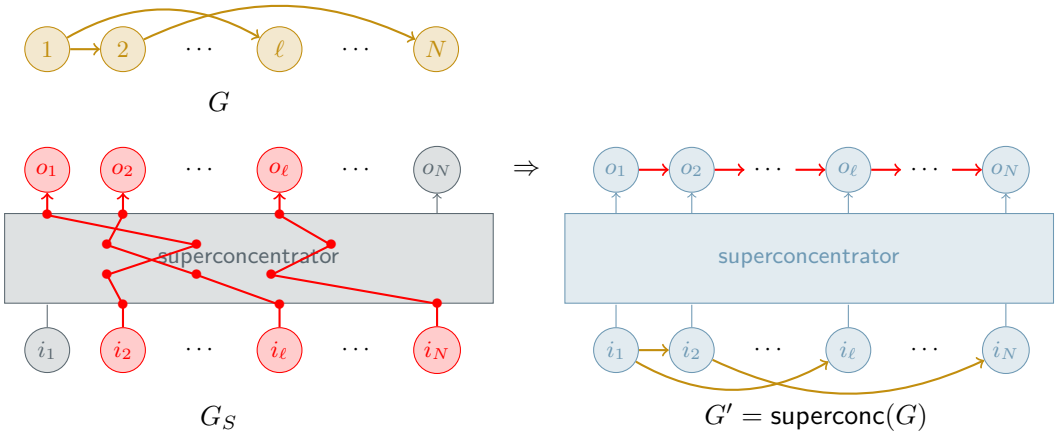


Figure 1 An example of the superconcentrator overlay $G' = \mathsf{superconc}(G)$. We remark that as illustrated in G_S, for any k inputs and k outputs (highlighted in red), there exist k vertex disjoint paths in a superconcentrator.

Improved Pebbling Strategy for $G' = \mathsf{superconc}(G)$

Step 1: Pebble the input nodes $\mathsf{input}(G') = G$.
Step 2: Efficiently pebble $\mathsf{interior}(G')$ using the property of superconcentrator.
Step 3: Pebble all nodes in $\mathsf{output}(G')$ by alternating between light and ballon phases.

- Light Phase - Walk pebble across the interval $I_i = [o_{(i-1)g+1}, o_{ig}]$ in g steps.
  - Precondition for the i^{th} light phase:
    - **(1)** Pebbles on all nodes $v \in \mathsf{parents}(o_{(i-1)g+1})$.
    - **(2)** Pebbles on all nodes $v \in \mathsf{parents}(I_i) \setminus I_i$.
    - **(3)** Pebbles on the set S where S is a (e, d)-depth reducing set for G.
  - Postcondition for the i^{th} light phase:
    - **(1)** Pebbles on the set S and node o_{ig}.
- Balloon Phase - Recover all the missing pebbles in $\mathsf{input}(G') \cup \mathsf{interior}(G')$ for the upcoming light phase.
  - Precondition for the i^{th} balloon phase:
    - **(1)** Pebbles on the set S.
    - **(2)** Pebble on node o_{ig+1} (the first node in the next light phase interval.)
  - Postcondition for the i^{th} balloon phase:
    - **(1)** Pebbles on all nodes in $\mathsf{input}(G') \cup \mathsf{interior}(G') = G' \setminus \mathsf{output}(G')$.
    - **(2)** Pebble on node o_{ig+1} (the first node in the next light phase interval.)

Figure 2 An improved pebbling strategy for $G' = \mathsf{superconc}(G)$. It brought ideas from [2].

Proof of Lemma 12. We will examine the pebbling cost of $\mathsf{superconc}(G)$ for each step shown in Figure 2.

- Step 1: We need to place pebbles on all input nodes in G'. By Theorem 18, the pebbling cost of $\mathsf{input}(G') = G$ will be upper bounded by[2]

$$\mathsf{cc}(G) \leq \min_{g \geq d} \left\{ eN + 2gN + \frac{N^2 d}{g} \right\}.$$

- Step 2: Start with a configuration with pebbles on every node in $\mathsf{input}(G')$. We have that $\mathsf{depth}(G') \setminus \mathsf{input}(G') = \log(42N)$. Therefore, in time $\log(42N)$, we can place pebbles on every node in $\mathsf{input}(G') \cup \mathsf{interior}(G')$. Hence, the total pebbling cost in Step 2 will be at most $42N \log(42N)$.
- Step 3: The goal for step 3 is to walk a pebble across the output nodes starting from o_1 to o_N. To save cost during this step, we should alternate light phases and balloon phases repeatedly N/g times in total since we walk pebble across the interval $I_i = [o_{(i-1)g+1}, o_{ig}]$ of length g in $\mathsf{output}(G')$ in each phase. Let S be a (e, d)-depth reducing set for G. In each light phase, to walk a pebble across the interval I_i, we should keep pebbles on S and $\mathsf{parents}(I_i) \setminus I_i$. Since each node in I_i has two parents outside the interval and we keep one pebble in I_i (the current node) for each step, the maximum number of pebbles to keep would be $|S| + 2g + 1 = e + 2g + 1$ for each step. Hence, the maximum pebbling cost to walk pebble across I_i in i^{th} light phase is $(e + 2g + 1)g$. In each balloon phase, we recover the pebbles in $\mathsf{input}(G') \cup \mathsf{interior}(G')$ for the next light phase. Since S is a (e, d)-depth reducing set, we have that $\mathsf{depth}(G' \setminus (S \cup \mathsf{output}(G'))) \leq d + \log(42N)$. Therefore, recovering the pebbles will cost at most $(d + \log(42N))42N$ for each balloon phase. Hence, the total pebbling cost for Step 3 will be at most $[(e + 2g + 1)g + (d + \log(42N))42N] \frac{N}{g}$.

Taken together, we have that

$$\begin{aligned}\mathsf{cc}(\mathsf{superconc}_2(G)) &\leq \min_{g \geq d} \left\{ \underbrace{eN + 2gN + \frac{N^2 d}{g}}_{\text{Step 1}} + \underbrace{42N \log(42N)}_{\text{Step 2}} + \underbrace{[(e + 2g + 1)g + (d + \log(42N)42N] \frac{N}{g}}_{\text{Step 3}} \right\} \\ &\leq \min_{g \geq d} \left\{ 2eN + 4gN + \frac{43dN^2}{g} + \frac{24N^2 \log(42N)}{g} + 42N \log(42N) + N \right\}\end{aligned}$$

as desired. ◀

References

1 Martín Abadi, Michael Burrows, and Ted Wobber. Moderately Hard, Memory-Bound Functions. In *Proceedings of the Network and Distributed System Security Symposium, NDSS 2003, San Diego, California, USA*, 2003.

2 Joël Alwen and Jeremiah Blocki. Efficiently Computing Data-Independent Memory-Hard Functions. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part II*, volume 9815 of *LNCS*, pages 241–271. Springer, Heidelberg, August 2016. doi:10.1007/978-3-662-53008-5_9.

[2] Note that Theorem 18 shows the upper bound of the pebbling cost to pebble the last node of G. Here, the difference is that we have to pebble all nodes in $\mathsf{input}(G') = G$, not the last node of G only. However, [2] says that we can recover all nodes concurrently by running one more balloon phase and such cost is already contained in the term $N^2 d/g$. Therefore, we have the same upper bound for $\mathsf{cc}(G)$.

3 Joël Alwen and Jeremiah Blocki. Towards practical attacks on argon2i and balloon hashing. In *Security and Privacy (EuroS&P), 2017 IEEE European Symposium on*, pages 142–157. IEEE, 2017.

4 Joël Alwen, Jeremiah Blocki, and Ben Harsha. Practical Graphs for Optimal Side-Channel Resistant Memory-Hard Functions. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 1001–1017. ACM Press, October/November 2017. doi:10.1145/3133956.3134031.

5 Joël Alwen, Jeremiah Blocki, and Krzysztof Pietrzak. Depth-Robust Graphs and Their Cumulative Memory Complexity. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part III*, volume 10212 of *LNCS*, pages 3–32. Springer, Heidelberg, April/May 2017. doi:10.1007/978-3-319-56617-7_1.

6 Joël Alwen, Jeremiah Blocki, and Krzysztof Pietrzak. Sustained Space Complexity. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 99–130. Springer, Heidelberg, April/May 2018. doi:10.1007/978-3-319-78375-8_4.

7 Joël Alwen and Vladimir Serbinenko. High Parallel Complexity Graphs and Memory-Hard Functions. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *47th ACM STOC*, pages 595–603. ACM Press, June 2015. doi:10.1145/2746539.2746622.

8 Sanjeev Arora, Boaz Barak, and David Steurer. Subexponential Algorithms for Unique Games and Related Problems. In *51st FOCS*, pages 563–572. IEEE Computer Society Press, October 2010. doi:10.1109/FOCS.2010.59.

9 Nikhil Bansal and Subhash Khot. Optimal Long Code Test with One Free Bit. In *50th Annual IEEE Symposium on Foundations of Computer Science, FOCS*, pages 453–462, 2009.

10 Jeremiah Blocki, Venkata Gandikota, Elena Grigorescu, and Samson Zhou. Relaxed Locally Correctable Code in Computationally Bounded Channels. In *IEEE International Symposium on Information Theory (ISIT)*, 2019.

11 Jeremiah Blocki, Benjamin Harsha, Siteng Kang, Seunghoon Lee, Lu Xing, and Samson Zhou. Data-Independent Memory Hard Functions: New Attacks and Stronger Constructions. *IACR Cryptology ePrint Archive*, 2018:944, 2018.

12 Jeremiah Blocki, Shubhang Kulkarni, and Samson Zhou. On Locally Decodable Codes in Resource Bounded Channels. *CoRR*, abs/1909.11245, 2019.

13 Jeremiah Blocki, Ling Ren, and Samson Zhou. Bandwidth-Hard Functions: Reductions and Lower Bounds. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018*, pages 1820–1836. ACM Press, October 2018. doi:10.1145/3243734.3243773.

14 Jeremiah Blocki and Samson Zhou. On the Depth-Robustness and Cumulative Pebbling Cost of Argon2i. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 445–465. Springer, Heidelberg, November 2017. doi:10.1007/978-3-319-70500-2_15.

15 Jeremiah Blocki and Samson Zhou. On the Computational Complexity of Minimal Cumulative Cost Graph Pebbling. *Financial Cryptography and Data Security (FC 2018)*, 2018.

16 Ethan Cecchetti, Ian Miers, and Ari Juels. PIEs: Public incompressible encodings for decentralized storage. Cryptology ePrint Archive, Report 2018/684, 2018. URL: https://eprint.iacr.org/2018/684.

17 Ashok K. Chandra. Efficient Compilation of Linear Recursive Programs. In *SWAT (FOCS)*, pages 16–25, 1973.

18 Moses Charikar, Venkatesan Guruswami, and Rajsekar Manokaran. Every Permutation CSP of arity 3 is Approximation Resistant. In *Proceedings of the 24th Annual IEEE Conference on Computational Complexity, CCC 2009, Paris, France, 15-18 July 2009*, pages 62–73, 2009.

19 Stephen A. Cook. An Observation on Time-storage Trade off. In *Proceedings of the Fifth Annual ACM Symposium on Theory of Computing*, STOC '73, pages 29–33, 1973.

20 Erik D. Demaine and Quanquan C. Liu. Inapproximability of the Standard Pebble Game and Hard to Pebble Graphs. In *Algorithms and Data Structures - 15th International Symposium, WADS 2017, St. John's, NL, Canada, July 31 - August 2, 2017, Proceedings*, pages 313–324, 2017.

21 Irit Dinur, Subhash Khot, Guy Kindler, Dor Minzer, and Muli Safra. Towards a proof of the 2-to-1 games conjecture? In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 376–389, 2018.

22 Cynthia Dwork, Andrew Goldberg, and Moni Naor. On Memory-Bound Functions for Fighting Spam. In *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*, volume 2729 of *Lecture Notes in Computer Science*, pages 426–444. Springer, 2003. URL: `http://www.iacr.org/cryptodb/archive/2003/CRYPTO/1266/1266.pdf`.

23 Cynthia Dwork, Moni Naor, and Hoeteck Wee. Pebbling and Proofs of Work. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 37–54. Springer, Heidelberg, August 2005. `doi:10.1007/11535218_3`.

24 Stefan Dziembowski, Sebastian Faust, Vladimir Kolmogorov, and Krzysztof Pietrzak. Proofs of Space. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 585–605. Springer, Heidelberg, August 2015. `doi:10.1007/978-3-662-48000-7_29`.

25 Ben Fisch. Tight Proofs of Space and Replication. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part II*, volume 11477 of *LNCS*, pages 324–348. Springer, Heidelberg, May 2019. `doi:10.1007/978-3-030-17656-3_12`.

26 Christian Forler, Stefan Lucks, and Jakob Wenzel. Memory-Demanding Password Scrambling. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part II*, volume 8874 of *LNCS*, pages 289–305. Springer, Heidelberg, December 2014. `doi:10.1007/978-3-662-45608-8_16`.

27 John R. Gilbert, Thomas Lengauer, and Robert Endre Tarjan. The Pebbling Problem is Complete in Polynomial Space. In *Proceedings of the 11h Annual ACM Symposium on Theory of Computing (STOC)*, pages 237–248, 1979.

28 Michel X. Goemans and David P. Williamson. .879-approximation algorithms for MAX CUT and MAX 2SAT. In *26th ACM STOC*, pages 422–431. ACM Press, May 1994. `doi:10.1145/195058.195216`.

29 Carl E. Hewitt and Michael S. Paterson. Record of the Project MAC Conference on Concurrent Systems and Parallel Computation, 1970.

30 Jia-Wei Hong and H. T. Kung. I/O complexity: The red-blue pebble game. In *Proceedings of the 13th Annual ACM Symposium on Theory of Computing, May 11-13, 1981, Milwaukee, Wisconsin, USA*, pages 326–333, 1981.

31 Subhash Khot. On the power of unique 2-prover 1-round games. In *34th ACM STOC*, pages 767–775. ACM Press, May 2002. `doi:10.1145/509907.510017`.

32 Subhash Khot, Guy Kindler, Elchanan Mossel, and Ryan O'Donnell. Optimal Inapproximability Results for MAX-CUT and Other 2-Variable CSPs? *SIAM J. Comput.*, 37(1):319–357, 2007.

33 Subhash Khot, Dor Minzer, and Muli Safra. On independent sets, 2-to-2 games, and Grassmann graphs. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 576–589, 2017.

34 Subhash Khot and Oded Regev. Vertex cover might be hard to approximate to within 2-epsilon. *J. Comput. Syst. Sci.*, 74(3):335–349, 2008.

35 Thomas Lengauer and Robert E. Tarjan. Asymptotically Tight Bounds on Time-space Trade-offs in a Pebble Game. *J. ACM*, 29(4):1087–1130, October 1982.

36 Quanquan Liu. Red-Blue and Standard Pebble Games: Complexity and Applications in the Sequential and Parallel Models. Master's thesis, Massachusetts Institute of Technology, February 2017. URL: `http://erikdemaine.org/theses/qliuM.pdf`.

37 Mohammad Mahmoody, Tal Moran, and Salil P. Vadhan. Publicly verifiable proofs of sequential work. In Robert D. Kleinberg, editor, *ITCS 2013*, pages 373–388. ACM, January 2013. `doi:10.1145/2422436.2422479`.

38 Jakob Nordström. Pebble Games, Proof Complexity, and Time-Space Trade-offs. *Logical Methods in Computer Science*, 9(3), 2013.

39 Krzysztof Pietrzak. Proofs of Catalytic Space. In Avrim Blum, editor, *ITCS 2019*, volume 124, pages 59:1–59:25. LIPIcs, January 2019. doi:10.4230/LIPIcs.ITCS.2019.59.

40 Nicholas Pippenger. Superconcentrators. *SIAM J. Comput.*, 6(2):298–304, 1977.

41 Ling Ren and Srinivas Devadas. Proof of Space from Stacked Expanders. In *Theory of Cryptography - 14th International Conference, TCC 2016-B, Beijing, China, October 31 - November 3, 2016, Proceedings, Part I*, pages 262–285, 2016.

42 Ling Ren and Srinivas Devadas. Bandwidth Hard Functions for ASIC Resistance. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 466–492. Springer, Heidelberg, November 2017. doi:10.1007/978-3-319-70500-2_16.

43 John E. Savage and Sowmitri Swamy. Space-time trade-offs on the FFT algorithm. *IEEE Transactions on Information Theory*, 24(5):563–568, 1978.

44 John E. Savage and Sowmitri Swamy. Space-Time Tradeoffs for Oblivious Interger Multiplications. In *ICALP*, pages 498–504, 1979.

45 Ola Svensson. Hardness of Vertex Deletion and Project Scheduling. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques - 15th International Workshop, APPROX, and 16th International Workshop, RANDOM. Proceedings*, pages 301–312, 2012.

46 Sowmitri Swamy and John E. Savage. Space-Time Tradeoffs for Linear Recursion. In *POPL*, pages 135–142, 1979.

47 Martin Tompa. Time-space Tradeoffs for Computing Functions, Using Connectivity Properties of Their Circuits. In *Proceedings of the Tenth Annual ACM Symposium on Theory of Computing*, STOC '78, pages 196–204, New York, NY, USA, 1978. ACM. doi:10.1145/800133.804348.

A Svensson's Construction

In this section, we review Svensson's Construction. Given an instance $\mathcal{U}$ of Unique Games, Svensson first constructs a weighted instance $\hat{G}_{\mathcal{U}}$ of the DAG reducibility problem. Recall that in the weighted DAG reducibility problem, we are given a DAG G with weights on each node and a target depth d and the goal is to find a minimum weight subset S such that $(G - S)$ contains no path of length d. Given (e_1, d_1) and (e_2, d_2) with $e_1 < e_2$ and $d_1 > d_2$ a weaker goal is simply to distinguish between the following cases: (1) there is a set S of weight at most e_1 s.t. $(G - S)$ contains no path of length d_1, and (2) for all sets S of weight at most e_2 the graph $(G - S)$ contains a path of length d_2. Svensson constructs $\hat{G}_{\mathcal{U}}$ s.t. distinguishing between these cases allows us to solve the original Unique Games instance $\mathcal{U}$.

A.1 Notation

We first review some notation that is used to describe Svensson's initial construction. For $x \in [k]^R$ and a subset S of not necessarily distinct indices of $[R]$, let

$$C_{x,S} = \{z \in [k]^R \,:\, z_j = x_j \forall j \notin S\}$$

denote the sub-cube whose coordinates not in S are fixed according to x. Let

$$C_{x,S,v,w} = \{z \in [k]^R \,:\, z_j = x_{\pi_{v,w}(j)} \forall \pi_{v,w}(j) \notin S\}$$

denote the image of the sub-cube $C_{x,s}$ under $\pi_{v,w}$. Similarly, let

$$C^{\oplus}_{x,S} = \{z \oplus \mathbf{1} \,:\, z \in C_{x,S}\},$$

where $\oplus$ denotes addition modulo k and $\mathbf{1}$ denotes an R-dimensional vector with all elements 1, and let

$$C^{\oplus}_{x,S,v,w} = \{z \oplus \mathbf{1} \,:\, z \in C_{x,S,v,w}\}.$$

A.2 Construction

In Svensson's initial construction, nodes are divided into two sets: bit-vertices and test-vertices. Bit-vertices are assigned weight infinity to guarantee that these nodes are not deleted. Test-vertices are assigned weight one. Here, we focus on the construction of $\hat{G}_{\mathcal{U}}$ though the graph $\hat{G}_{\mathcal{U}}$ is later transformed into an instance $G_{\mathcal{U}}$ of the unweighted DAG reducibility problem, i.e., distinguishing between the cases that $G_{\mathcal{U}}$ is (e_1, d_1)-reducible and (e_2, d_2)-depth robust is sufficient to solve the original Unique Games instance $\mathcal{U}$. See Figure 3 for a simple example of $\hat{G}_{\mathcal{U}}$ along with the transformation to the unweighted instance $G_{\mathcal{U}}$. The DAG $\hat{G}_{\mathcal{U}}$ is defined formally as follows:

- For some L to be fixed, there are $L+1$ layers of bit-vertices. Each bit-layer ℓ with $0 \le \ell \le L$ the DAG $\hat{G}_{\mathcal{U}}$ contains bit-vertices $b^{\ell}_{w,x}$ for each $w \in W$ and $x \in [k]^R$. Each bit-vertex is assigned weight ∞.
- There are L layers of test-vertices. For each $0 \le \ell \le L-1$, the DAG $\hat{G}_{\mathcal{U}}$ contains test-vertices $t^{\ell}_{x,S,v,w_1,\ldots,w_{2t}}$ for every $x \in [k]^R$, every sequence of indices $S = (s_1, \ldots, s_{\varepsilon R}) \in [R]^{\varepsilon R}$, every $v \in V$ and every sequence $(w_1, \ldots, w_{2t})$ of $2t$ not necessarily distinct neighbors of v. Each test-vertex is assigned weight 1.
- If $\ell \le \ell'$ and $z \in C_{x,S,v,w_j}$, then there is an edge from bit-vertex $b^{\ell}_{w_j,z}$ to test-vertex $t^{\ell'}_{x,S,v,w_1,\ldots,w_{2t}}$ for each $1 \le j \le 2t$.
- If $\ell > \ell'$ and $z \in C^{\oplus}_{x,S,v,w_j}$, then there is an edge from test-vertex $t^{\ell'}_{x,S,v,w_1,\ldots,w_{2t}}$ to bit-vertex $b^{\ell}_{w_j,z}$ for each $1 \le j \le 2t$.
- If T is the total number of test-vertices, then L is selected so that $\delta^2 L \ge T^{1-\delta}$.

A.3 Transformation

As mentioned before, in the Svensson's construction, the bit-vertices are given weight ∞ so that they are never deleted, and the graph can be simplified in the following manner without altering the reduction. The transformation to $G_{\mathcal{U}}$ is defined formally as follows:

- For each $0 \le \ell \le L-1$, there exists a vertex $v^{\ell}_{x,S,v,w_1,\ldots,w_{2t}}$ for every $x \in [k]^R$, every sequence of indices $S = (s_1, \ldots, s_{\varepsilon R}) \in [R]^{\varepsilon R}$, every $v \in V$ and every sequence $(w_1, \ldots, w_{2t})$ of $2t$ not necessarily distinct neighbors of v.
- If γ is the number of vertices in each layer, then L is selected so that $\delta^2 L \ge (\gamma L)^{1-\delta}$.
- There exists an edge between $v^{\ell}_{x,S,v,w_1,\ldots,w_{2t}}$ and $v^{\ell'}_{x',S',v',w'_1,\ldots,w'_{2t}}$ if and only if $\ell < \ell'$ and there exist i, j such that $C^{\oplus}_{x,S,v,w_i} \cap C_{x',S',v',w'_j}$ is nonempty.

▶ **Example 14.** In this example, we will illustrate how to reduce from a Unique Games instance $\mathcal{U}$ to a Svensson's construction $\hat{G}_{\mathcal{U}}$, and a simplification procedure from $\hat{G}_{\mathcal{U}}$ to $G_{\mathcal{U}}$ by examining a simple toy example.

Consider the following Unique Games instance $\mathcal{U} = (G = (V, W, E), [R], \{\pi_{v,w}\}_{v,w})$ with $V = \{v_1\}, W = \{w_1\}, E = \{(v_1, w_1)\}, \pi_{v_1,w_1} : \{1,2\} \to \{2,1\}$, a labeling $\rho : (V \cup W) \to [R]$ such that $\rho(v_1) = 1, \rho(w_1) = 2$, and with the parameters $R = 2, k = 2, t = 1, \delta = 0.1$ and $\epsilon = 0.5$. Then we have the following observations when constructing $\hat{G}_{\mathcal{U}}$:

- Each bit-layer ℓ with $0 \le \ell \le L$ contains bit-vertices $b^{\ell}_{w,x}$ for each $w \in W$ and $x \in [k]^R$. Hence, the number of bit-vertices in each layer is $|W| \times |[k]^R| = 1 \times 2^2 = 4$. That is, for each layer i, we have the following bit-vertices:

  $b^{i}_{w_1,(11)}, b^{i}_{w_1,(12)}, b^{i}_{w_1,(21)}$, and $b^{i}_{w_1,(22)}$.

- Each test-layer ℓ with $0 \leq \ell \leq L-1$ contains test-vertices $t^{\ell}_{x,S,v,w_1,\ldots,w_{2t}}$ for every $x \in [k]^R$, every sequence of indices $S = (s_1, \ldots, s_{\varepsilon R}) \in [R]^{\varepsilon R}$, every $v \in V$ and every sequence $(w_1, \ldots, w_{2t})$ of $2t$ not necessarily distinct neighbors of v. Since $\varepsilon R = 1$ and v_1 has only one neighbor w_1, the number of test-vertices in each layer is $|[k]^R| \times |S| \times |V| \times |N_G(v_1)^{2t}| = 2^2 \times 2 \times 1 \times 1^2 = 8$, where $N_G(v)$ denotes the set of neighbors of v in a graph G. That is, for each layer i, we have the following test-vertices (from now on, we omit the subscript $v, w_1, \ldots, w_{2t}$ in this example since there is only one such case for each test-vertex):

  $$t^i_{(11),(1)}, t^i_{(12),(1)}, t^i_{(21),(1)}, t^i_{(22),(1)}, t^i_{(11),(2)}, t^i_{(12),(2)}, t^i_{(21),(2)}, \text{ and } t^i_{(22),(2)}.$$

- There exists an edge from bit vertex $b^{\ell}_{w_1,z}$ to test-vertex $t^{\ell'}_{x,S}$ if $\ell \leq \ell'$ and $z \in C_{x,S,v_1,w_1}$. We recall that $C_{x,S,v_1,w_1} = \{z \in [k]^R : z_j = x_{\pi_{v_1,w_1}(j)}\ \forall \pi_{v_1,w_1}(j) \notin S\}$. Now it is easy to see that if $S = \{1\}$, $z \in C_{x,S,v_1,w_1}$ if and only if $z_1 = x_2$, and if $S = \{2\}$, $z \in C_{x,S,v_1,w_1}$ if and only if $z_2 = x_1$. Therefore, we have an edge from $b^i_{w_1,(12)}$ to $t^j_{(11),(1)}, t^j_{(21),(1)}, t^j_{(21),(2)}$, and $t^j_{(22),(2)}$ for all $0 \leq i \leq j < L$, and so on.
- There exists an edge from test-vertex $t^{\ell'}_{x,S}$ to bit-vertex $b^{\ell}_{w_1,z}$ if $\ell > \ell'$ and $z \in C^{\oplus}_{x,S,v_1,w_1}$, where $\oplus$ denotes addition modulo $k = 2$ and $C^{\oplus}_{x,S,v_1,w_1} = \{z \oplus 1 : z \in C_{x,S,v_1,w_1}\}$. Hence, for example, if there is an edge from $b^i_{w_1,(12)}$ to $t^j_{x,S}$ then there should be an edge from $t^j_{x,S}$ to $b^{j'}_{w_1,(21)}$ for all $j' > j$ since $(12) \oplus \mathbf{1} = (12) \oplus (11) = (21)$.

When transforming $\hat{G}_{\mathcal{U}}$ into $G_{\mathcal{U}}$, we can observe that $C^{\oplus}_{x,S,v_1,w_1} \cap C_{x',S',v_1,w_1}$ is nonempty if and only if there is a path between two test-vertices through one bit-vertex. Taken together, we have the following structure of graphs reduced from a Unique Games instance $\mathcal{U}$, as shown in Figure 3. ◀

B Modified Construction

Given an instance $\hat{G}_{\mathcal{U}}$ of the Svensson's construction and a γ-extreme depth-robust graph $G_{\gamma,L+1} = (V_\gamma = [L+1], E_\gamma)$, we formally define our modified instance $G' = \mathsf{Sparsify}_{G_{\gamma,L+1}}(\hat{G}_{\mathcal{U}})$ in the following manner.

Transformation $\mathsf{Sparsify}_{G_{\gamma,L+1}}(\hat{G}_{\mathcal{U}})$

Input: An instance $\hat{G}_{\mathcal{U}} = (V, E)$ of the Svensson's construction, whose vertices are partitioned into $L+1$ bit-layers $B_0, \ldots, B_L$ and L test-layers $T_0, \ldots, T_{L-1}$, a γ-extreme depth robust graph $G_{\gamma,L+1} = (V_\gamma = [L+1], E_\gamma)$.

1. Let $G' = (V, E)$ be a copy of $\hat{G}_{\mathcal{U}}$.
2. If $e = (b, t)$ is an edge in $\hat{G}_{\mathcal{U}}$, where $b \in B_i$ and $t \in T_j$, delete e from G' if $i \neq j$ and $(i, j) \notin E_\gamma$.
3. If $e = (t, b)$ is an edge in $\hat{G}_{\mathcal{U}}$, where $b \in B_i$ and $t \in T_j$, delete e from G' if $(j, i) \notin E_\gamma$.

Output: G'

We give an illustration of the Sparsify procedure in Figure 4.

Correspondingly, our modified instance can also be simplified in the following manner without altering the reduction.

- For a input parameter γ, let $G_{\gamma,L+1} = (V_\gamma = [L+1], E_\gamma)$ be an $\frac{\gamma}{2}$-extreme depth robust graph with $L+1$ vertices, which we use $[L+1]$ to represent.

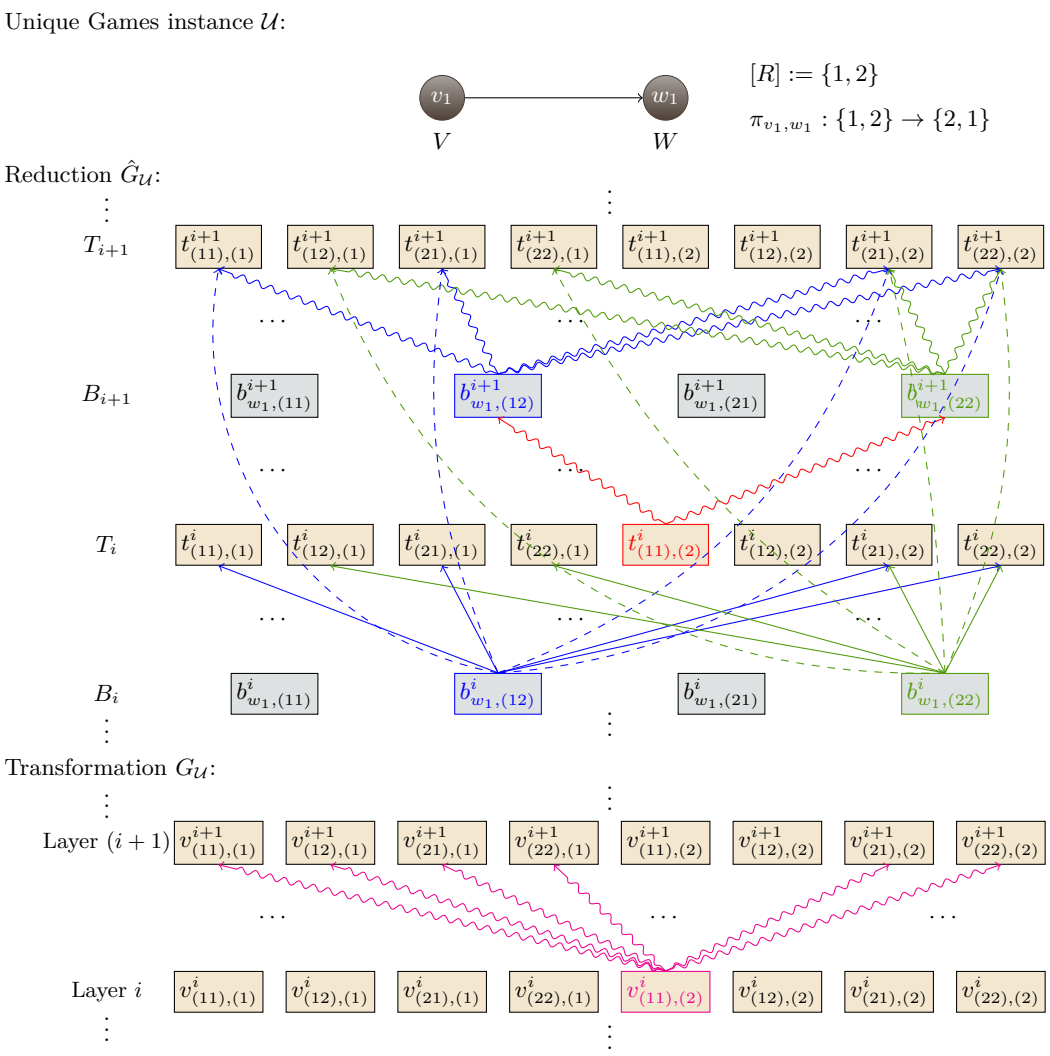


Figure 3 An example of reduction $\hat{G}_{\mathcal{U}}$ from a Unique Games instance $\mathcal{U}$ and a transformation into $G_{\mathcal{U}}$ illutrated in Example 14. We remark that in $\hat{G}_{\mathcal{U}}$ and $G_{\mathcal{U}}$, we only drew edges from the highlighted vertices for simplicity and readability. In $G_{\mathcal{U}}$, we only keep test-vertices from $\hat{G}_{\mathcal{U}}$ and connect two vertices if there is a path between those two through one bit-vertex. Therefore, we can easily check that totally 6 edges (shown in snaked magenta edges) going out from the vertex $v^i_{(11),(2)}$ in $G_{\mathcal{U}}$ comes from the edges $(t^i_{(11),(2)}, b^{i+1}_{w_1,(12)})$ and $(t^i_{(11),(2)}, b^{i+1}_{w_1,(22)})$ (shown in a snaked red edge) and edges from $b^{i+1}_{w_1,(12)}$ and $b^{i+1}_{w_1,(22)}$ to the test layer T_{i+1} (shown in snaked blue/green edges) in $\hat{G}_{\mathcal{U}}$. We also note that the edges starting from the i^{th} bit layer B_i go to every upper test layers T_j for all $j \geq i$. Those edges are represented as dashed ones.

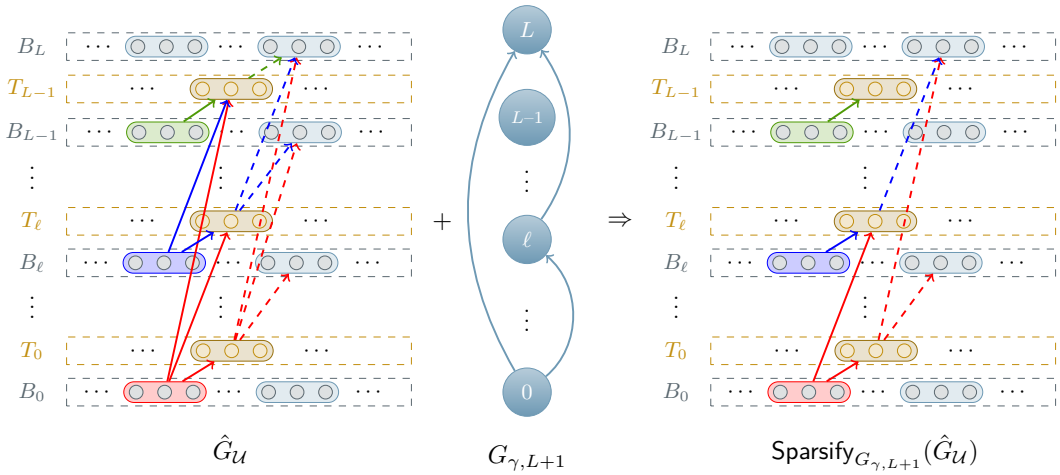


Figure 4 A description of the transformation $\mathsf{Sparsify}_{G_{\gamma,L+1}}(\hat{G}_{\mathcal{U}})$ where $\hat{G}_{\mathcal{U}}$ is Svensson's construction and $G_{\gamma,L+1}$ is a γ-extreme depth-robust graph. We remark that the edges between the subsets of nodes indicate that every node in the input subset is connected to every node in the output subset (in this example, there should be $3 \times 3 = 9$ edges from B_0 to T_0.)

- For each $0 \leq \ell \leq L-1$, there exists a vertex $v^{\ell}_{x,S,v,w_1,\ldots,w_{2t}}$ for every $x \in [k]^R$, every sequence of indices $S = (s_1, \ldots, s_{\varepsilon R}) \in [R]^{\varepsilon R}$, every $v \in V$ and every sequence $(w_1, \ldots, w_{2t})$ of $2t$ not necessarily distinct neighbors of v.
- If γ is the number of vertices in each layer, then L is selected so that $\delta^2 L \geq (\gamma L)^{1-\delta}$.
- There exists an edge between $v^{\ell}_{x,S,v,w_1,\ldots,w_{2t}}$ and $v^{\ell'}_{x',S',v',w'_1,\ldots,w'_{2t}}$ if and only if $\ell < \ell'$, the edge (ℓ, ℓ') is in E_γ, and there exist i, j such that $C^{\oplus}_{x,S,v,w_i} \cap C_{x',S',v',w'_j}$ is nonempty.

We first recall the following definition of influence of the i^{th} coordinate:

$$\mathsf{Infl}_i(f) = \mathbb{E}_x\left[\mathsf{Var}(f) | x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_R\right].$$

We now reference the key theorem used in Svensson's analysis.

▶ **Theorem 15** ([31, 45]). *For every $\varepsilon, \delta > 0$ and integer k, there exists $\eta > 0$ and integers t, d such that any collection of functions $f_1, \ldots, f_t : [k]^R \to \{0,1\}$ that satisfies*

- $\forall j, \mathbb{E}[f_j] \geq \delta$
- $\forall i \in [R]$, $\forall 1 \leq \ell_1 \neq \ell_2 \leq t$: $\min\left\{\mathsf{Infl}^d_i(f_{\ell_1}), \mathsf{Infl}^d_i(f_{\ell_2})\right\} \leq \eta$

has

$$\Pr_{x,S_\varepsilon}\left[\bigwedge_{j=1}^{t} f_j(C_{x,S_\varepsilon}) \equiv 0\right] \leq \delta.$$

We now show that the transformed graph maintains similar properties as Svensson's construction, given an instance of Unique Games. The following statement is analogous to Lemma 4.7 in [45].

▶ **Reminder of Lemma 4.** *Let χ be any coloring of $\mathsf{Sparsify}_{G_{\gamma,L+1}}(\hat{G}_{\mathcal{U}})$. If the Unique Games instance has no labeling that satisfies a fraction $\frac{\delta\eta^2}{t^2k^2}$ of the constraints and at least $32\delta^2|T|$ test vertices are consistent with χ, then there exists $w \in W$ with*

$$\Pr_{\ell \in [L]}\left[\chi(w,\ell') > \chi(w,\ell) \text{ for all } \ell' > \ell \text{ with } (\ell,\ell') \in E_\gamma\right] \geq 32\delta^2.$$

Proof of Lemma 4. As in [45], an equivalent formulation of the problem is finding a coloring χ in $\{1, 2, \ldots, k\}$ to each bit-vertex to minimize the number of unsatisfied test-vertices. Unlike in [45], we say a test-vertex $t^{\ell}_{x,S,v,w_1,\ldots,w_{2t}}$ is satisfied if

$$\max_{\substack{1\le j\le 2t\\ z\in C_{x,S,v,w_j}}} \chi\left(b^{\ell'}_{w_j,z}\right) < \min_{\substack{1\le j\le 2t\\ z\in C^{\oplus}_{x,S,v,w_j}}} \chi\left(b^{\ell''}_{w_j,z}\right),$$

for all $\ell' \le \ell < \ell''$ with $(\ell', \ell'') \in E_\gamma$, so that all the predecessors of ℓ are assigned lower colors than the successors of ℓ.

We also define the color $\chi(w, \ell)$ for $w \in W$ and $0 \le \ell \le L$ as the maximum color that satisfies

$$\Pr_x\left[\chi(b^{\ell}_{w,x}) \ge \chi(w, \ell)\right] \ge 1 - \delta.$$

For $w \in W$ and $0 \le \ell \le L - 1$, define the indicator function $f^{\ell}_w : [k]^R \to \{0, 1\}$ by

$$f^{\ell}_w(x) = \begin{cases} 0 & \text{if } \chi\left(b^{\ell'}_{w,x}\right) > \chi(w, \ell) \text{ for all } \ell' > \ell \text{ with } (\ell, \ell') \in E_\gamma,\\ 1 & \text{otherwise.}\end{cases}$$

We call a test-vertex $w \in W$ good in test-layer ℓ if for $\chi(w, \ell') > \chi(w, \ell)$ for every edge of the form (ℓ, ℓ') in the γ-extreme depth-robust graph $G_{\gamma,L+1} = (V_\gamma = [L + 1], E_\gamma)$.

▷ Claim 16. If the Unique Games instance has no labeling satisfying a fraction $\frac{\delta\eta^2}{t^2k^2}$ of the constraints and a fraction 16δ of the vertices of test-layer ℓ are satisfied, then at least a 2δ fraction of the vertices are good in test-layer ℓ.

Proof. Let A_ℓ be the set of satisfied vertices of test-layer ℓ so that for all $\ell' \le \ell < \ell''$ with $(\ell', \ell'') \in E_\gamma$, it follows that

$$\Pr_{x,S,v,w_1,\ldots,w_{2t}}\left[\max_{\substack{1\le j\le 2t\\ z\in C_{x,S,v,w_j}}} \chi\left(b^{\ell'}_{w_j,z}\right) < \min_{\substack{1\le j\le 2t\\ z\in C^{\oplus}_{x,S,v,w_j}}} \chi\left(b^{\ell''}_{w_j,z}\right)\right] \ge 16\delta,$$

since at least 16δ fraction of the vertices in A_ℓ are satisfied. We call a tuple $(v, w_1, \ldots, w_{2t})$ good if

$$\Pr_{x,S}\left[\max_{\substack{1\le j\le 2t\\ z\in C_{x,S,v,w_j}}} \chi\left(b^{\ell'}_{w_j,z}\right) < \min_{\substack{1\le j\le 2t\\ z\in C^{\oplus}_{x,S,v,w_j}}} \chi\left(b^{\ell''}_{w_j,z}\right)\right] \ge 8\delta,$$

for all $\ell' \le \ell < \ell''$ with $(\ell', \ell'') \in E_\gamma$. Observe that at least 8δ fraction of the tuples are good.

From the definition of $\chi(w, \ell')$, we have that $\Pr_x\left[\chi\left(b^{\ell'}_{w,x}\right)\right] \ge 1 - \delta$. Hence for a good tuple, it follows that

$$7\delta \le \Pr_{x,S}\left[\max_{1\le j\le 2t} \chi(w_j, \ell') < \min_{\substack{1\le j\le 2t\\ z\in C^{\oplus}_{x,S,v,w_j}}} \chi\left(b^{\ell''}_{w_j,z}\right)\right] \le \Pr_{x,S}\left[\bigwedge_{j=1}^{2t} f^{\ell}_{w_j}(C_{x,S,v,w_j}) \equiv 0\right],$$

for all $\ell' \le \ell < \ell''$ with $(\ell', \ell'') \in E_\gamma$.

Therefore by Theorem 15, at least one of the following cases holds:

1. more than t of the functions have $\mathbb{E}\left[f_{w_j}^{\ell}\right] < \delta$ so that $\chi(w_j, \ell') > \chi(w_j, \ell)$ for every edge of the form (ℓ, ℓ') in E_γ, or
2. there exist $1 \leq \ell_1 \neq \ell_2 \leq t$ and $j \in \mathcal{I}[w_{\ell_1}], j' \in \mathcal{I}[w_{\ell_2}]$ such that $\pi_{v,w_{\ell_1}}(j) = \pi_{v,w_{\ell_2}}(j')$, where

$$\mathcal{I}[w] = \{i \in [R] : \mathsf{Infl}_i^d(f_w^{\ell}) \geq \eta\}.$$

If Condition 1 holds for at least $\frac{1}{2}$ of the good tuples, or equivalently a 4δ fraction of all tuples, then at least a 2δ fraction of the test-vertices are good in test-layer ℓ because we can pick a vertex $w_j \in W$ uniformly at random by picking a tuple $(v, w_1, \ldots, w_{2t})$ and then taking one of the vertices $w_1, \ldots, w_{2t}$ at random. Conditioned on the (at least) 2δ probability that the tuple is good and satisfies Condition 1, the probability that $\chi(w_j, \ell') > \chi(w_j, \ell)$ for every edge of the form (ℓ, ℓ') in E_γ for the sampled vertex w_j is at least $\frac{1}{2}$. Therefore, $\Pr_w[\chi(w, \ell') > \chi(w, \ell)$ for all $\ell' > \ell$ with $(\ell, \ell') \in E_\gamma] \geq 2\delta$, so that at least 2δ fraction of the test-vertices are good in test-layer ℓ.

By way of contradiction, we can show that if Condition 2 were to hold for more than half of the good tuples, the assumption that the Unique Games instance has no labeling satisfying a fraction $\frac{\delta\eta^2}{t^2k^2}$ of the constraints is violated. The argument holds exactly as Claim 4.8 in [45], but we repeat it here for completeness.

For every $w \in W$, let $\rho(w)$ be a random label from $\mathcal{I}[w]$. For every $v \in V$, let w be a random neighbor of v and let $\rho(v) = \pi_{v,w}(\rho(w))$. If Condition 2 holds for half of the good tuples, then a random tuple has this property with at least probability 4δ. Thus with probability at least $\frac{1}{4t^2}$, $w = w_{\ell_1}$ and $w' = w_{\ell_2}$ for w, w' randomly picked from the set $\{w_1, \ldots, w_{2t}\}$. Moreover, [45, 9] observes that with probability at least $\frac{\eta^2}{k^2}$, the labeling procedure defines $j = \rho(w)$ and $j' = \rho(w')$. Hence if Condition 2 holds for half of the good tuples,

$$\Pr_{v,w,w'}[\pi_{v,w}(\rho(w)) = \pi_{v,w'}(\rho(w'))] \geq \frac{4\delta\eta^2}{4t^2k^2},$$

so that over the randomness of the labeling procedure,

$$\Pr_{(v,w)}[\rho(v) = \pi_{v,w}(\rho(w))] \geq \frac{\delta\eta^2}{t^2k^2},$$

which contradicts the assumption that the Unique Games instance has no labeling satisfying a fraction $\frac{\delta\eta^2}{t^2k^2}$ of the constraints is violated. ◁

Consider a subgraph induced by all bit-vertices and a fraction 32δ of the test-vertices and consider the minimum number of colors required for a coloring χ to satisfy the 32δ fraction of the test-vertices. Since at least 16δ fraction of the test-vertices are good in at least 16δ fraction of the test-layers, then by Claim 16,

$$\Pr_{\ell \in [L], w \in W}[\chi(w, \ell') > \chi(w, \ell) \text{ for all } \ell' > \ell \text{ with } (\ell, \ell') \in E_\gamma] \geq 16\delta \cdot 2\delta = 32\delta^2.$$

Therefore, there exists $w \in W$ with $\Pr_{\ell \in [L]}[\chi(w, \ell') > \chi(w, \ell)$ for all $\ell' > \ell$ with $(\ell, \ell') \in E_\gamma] \geq 32\delta^2$. ◀

Interestingly, we can exactly compute the pebbling complexity of the simplified Svensson's construction, when the graph is only represented with the test-vertices.

▶ **Lemma 17.** *Given a (simplified) Svensson's construction $G_{\mathcal{U}}$ that consists of N vertices partitioned across L layers,* $\mathsf{cc}(G_{\mathcal{U}}) = \frac{N(L+1)}{2}$.

Proof. We first show that the pebbling complexity of $G_{\mathcal{U}}$ is at least $\frac{N(L+1)}{2}$. Observe that the Svensson's construction contains $\frac{N}{L}$ vertices in each layer and furthermore, for each pair of layers i and j, there is a perfect matching between vertices of layer i and vertices of layer j among the edges connecting layers i and j. Let $\mathcal{M}_{i,j}$ be the subset of edges between i and j that is perfect matching.

For a given pebble u in layer j, let v_i be the vertex in layer i matched to u by $\mathcal{M}_{i,j}$. To pebble a vertex u in layer j, all of its parents must contain pebbles in the previous round. Namely, there must be a pebble on v_i for all $1 \leq i < j$ in the previous round. Since each $\mathcal{M}_{i,j}$ is a perfect matching, there must be $j-1$ pebbles on the graph solely for the purpose of pebbling node u in layer j. Thus pebbling each node in layer j induces a pebbling cost of at least $j-1$. Since there are $\frac{N}{L}$ pebbles in each layer and L layers, then the total pebbling cost is at least $\frac{N}{L}\sum_{j=1}^{L}(j-1) = \frac{N(L+1)}{2}$, which lower bounds $\mathsf{cc}(G_{\mathcal{U}})$.

On the other hand, consider the natural pebbling where all the pebbles in layer j are pebbled in round j, and no pebble is ever removed. Then the graph is completely pebbled in L rounds, since layer L is pebbled in round L. Moreover, the cost of pebbling round j is $\frac{N}{L}(j-1)$. Hence, the pebbling cost is $\frac{N}{L}\sum_{j=1}^{L}(j-1) = \frac{N(L+1)}{2}$, which upper bounds $\mathsf{cc}(G_{\mathcal{U}})$. ◀

Finally, we give a formal description of the procedure $\mathsf{IDR}(G,\gamma)$. Recall that $\mathsf{IDR}(G,\gamma)$ for a graph $G=(V,E)$ replaces each vertex $v \in V$ with a path $P_v = v_1,\ldots,v_{\delta+\gamma}$, where δ is the indegree of G. For each edge $(u,v) \in E$, we add the edge $(u_{\delta+\gamma}, v_i)$ whenever (u,v) is the i^{th} incoming edge of v, according to some fixed ordering. [5] give parameters e_2 and d_2 so that $\mathsf{IDR}(G,\gamma)$ is (e_2,d_2)-depth robust if G is (e,d)-depth robust. We complete the reduction by giving parameters e_1 and d_1 so that $\mathsf{IDR}(G,\gamma)$ is (e_1,d_1)-reducible if G is (e,d)-reducible.

Transformation $\mathsf{IDR}(G,\gamma)$

Input: An DAG $G=(V,E)$ with indegree δ, parameter γ.

1. Let the vertices of G be $[|V|]$.
2. Initialize G' to be a graph with $(\delta+\gamma)|V|$ vertices and let these vertices be $[(\delta+\gamma)|V|]$
3. If $(\delta+\gamma)n+1 \leq u < (\delta+\gamma)n$ for some integer n, add edge $(u, u+1)$ to G'.
4. If (u,v) is the i^{th} incoming edge of v by some fixed predetermined ordering, then add $(u_{\delta+\gamma}, v_i)$ to G'.

Output: G'

We given an illustration of the IDR transformation in Figure 5.

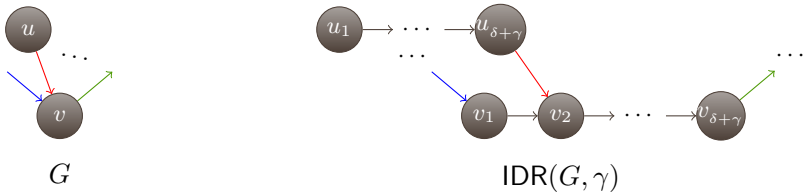


Figure 5 An example of the transformation $\mathsf{IDR}(G,\gamma)$. We remark that if the red edge (u,v) is the 2^{nd} incoming edge of v, then in $\mathsf{IDR}(G,\gamma)$ we should add $(u_{\delta+\gamma}, v_2)$ to G'.

C Useful Theorems

We rely on the following results in our constructions and proofs.

▶ **Theorem 18** ([2]). *Let G be a DAG with N vertices and indegree δ. If G is (e,d)-reducible, then* $\mathsf{cc}(G) \leq \min_{g \geq d} \left\{ eN + \delta g N + \frac{N^2 d}{g} \right\}$.

▶ **Theorem 19** ([5]). *Let G be a DAG with N vertices and indegree δ. If G is (e,d)-depth robust, then* $\mathsf{cc}(G) \geq ed$.

While Theorem 18 and Theorem 19 are nice results that relate the pebbling complexities of (e_1, d_1)-reducible and (e_2, d_2)-depth robust graphs, these statements are ultimately misleading in that $d \leq N$ and thus there will never be a gap between the pebbling complexities of the graphs.

▶ **Theorem 20** ([45]). *For any integer $k \geq 2$ and constant $\varepsilon > 0$, given a DAG G with N vertices, it is Unique Games hard to distinguish between the following cases:*

- *(Completeness): G is $\left(\left(\frac{1-\varepsilon}{k}\right) N, k\right)$-reducible.*
- *(Soundness): G is $\left((1-\varepsilon)N, N^{1-\varepsilon}\right)$-depth robust.*

▶ **Definition 21.** *Given a parameter $0 < \gamma < 1$, a DAG $G = (V, E)$ is* γ-extreme depth-robust *if G is (e,d)-depth robust for any e, d such that $e + d \leq (1-\gamma)N$.*

▶ **Theorem 22** ([6]). *For any fixed $0 < \gamma < 1$, there exists a constant $c_1 > 0$ such that for all integers $N > 0$, there exists an γ-extreme depth robust graph G with N vertices and* $\mathsf{indeg}(G), \mathsf{outdeg}(G) \leq c_1 \log N$.

▶ **Lemma 23** ([11]). *Let G be an (e,d)-depth robust graph with N vertices. Then*

$$\mathsf{cc}(\mathsf{superconc}(G)) \geq \min\left(\frac{eN}{8}, \frac{dN}{8}\right).$$

D Missing Proofs

▶ **Reminder of Lemma 7.** *There exists a polynomial time procedure $\mathsf{IDR}(G, \gamma)$ that takes as input a DAG G with N vertices and $\mathsf{indeg}(G) = \delta$ and outputs a graph $G' = \mathsf{IDR}(G, \gamma)$ with $(\delta + \gamma)N$ vertices and $\mathsf{indeg}(G') = 2$. Moreover, the following properties hold: (1) If G is (e,d)-reducible, then $\mathsf{IDR}(G, \gamma)$ is $(e, (\delta+\gamma)\cdot d)$-reducible, and (2) If G is (e,d)-depth robust, then $\mathsf{IDR}(G, \gamma)$ is $(e, \gamma \cdot d)$-depth robust.*

Proof of Lemma 7. Alwen et al. [5] show that $\mathsf{IDR}(G, \gamma)$ is $(e, \gamma \cdot d)$-depth robust if G is (e,d)-depth robust. It remains to show that $\mathsf{IDR}(G, \gamma)$ is $(e, (\delta+\gamma)\cdot d)$-reducible if G is (e,d)-reducible.

Given an (e,d)-reducible graph $G = (V, E)$ of N vertices, we use $[N]$ to represent the vertices of G and let $G' = \mathsf{IDR}(G, \gamma)$ so that the vertices of G' can be associated with $[(\delta+\gamma)N]$. Let S be a set of e vertices in G such that $\mathsf{depth}(V - S) < d$. Let S' be a set of e vertices in G' so that $(\delta+\gamma)v \in S'$ for each vertex $v \in S$.

Suppose, by way of contradiction, that there exists a path P' of length $(\delta+\gamma)\cdot d$ in $G' - S'$. Observe that if $y, z \in G'$ such that $(\delta+\gamma)a + 1 \leq y \leq (\delta+\gamma)(a+1)$ and $(\delta+\gamma)b + 1 \leq z \leq (\delta+\gamma)(b+1)$ for integers $a < b$, then y cannot be connected to z unless $y = (\delta+\gamma)(a+1)$. Hence that if P' contains vertex $u \in G'$ such that $(\delta+\gamma)c + 1 \leq u \leq (\delta+\gamma)(c+1)$ and u

is not one of the final $\delta+\gamma-1$ vertices of P', then $(\delta+\gamma)(c+1) \in P'$. Thus by a simple Pigeonhole argument, there exists at least d integers $j_1, j_2, \ldots, j_d$ such that $(\delta+\gamma)j_n \in P'$ for $1 \leq n \leq d$ and moreover there exists an edge in P' from each vertex $(\delta+\gamma)j_n$ to some vertex w such that $(\delta+\gamma)(j_{n+1}-1)+1 \leq w \leq (\delta+\gamma)j_{n+1}$ for $1 \leq n \leq d-1$. However, this implies that $j_1, \ldots, j_d$ is a path in G by construction of $\mathsf{IDR}(G,\gamma)$. Moreover, since $(\delta+\gamma)v \in S'$ for each vertex $v \in S$, this implies that $j_1, \ldots, j_d$ is a path of length d in $G-S$, which contradicts the assumption that $\mathsf{depth}(G-S) < d$. ◀

▶ **Reminder of Corollary 8.** *For any integer $k \geq 2$ and constant $\varepsilon > 0$, given a DAG G with N vertices and maximum indegree $\mathsf{indeg}(G) = 2$, it is Unique Games hard to decide whether G is (e_1, d_1)-reducible or (e_2, d_2)-depth robust for (Completeness): $e_1 = \frac{1}{k}N^{\frac{1}{1+2\varepsilon}}$ and $d_1 = kN^{\frac{2\varepsilon}{1+2\varepsilon}}$, and (Soundness): $e_2 = (1-\varepsilon)N^{\frac{1}{1+2\varepsilon}}$ and $d_2 = 0.9N^{\frac{1+\varepsilon}{1+2\varepsilon}}$.*

Proof of Corollary 8. Suppose G' is a graph with M vertices. By applying Lemma 7 to Theorem 6 and setting $k = M^{2\varepsilon}$ and $\gamma = M^{2\varepsilon} - \delta$, then we start from a graph with M vertices and end with a graph G with $N = M^{1+2\varepsilon}$ vertices or equivalently, $M = N^{1/(1+2\varepsilon)}$. Thus, $G = \mathsf{IDR}(G', \gamma)$ is (e,d)-reducible for $e = \frac{(1-\varepsilon)M}{k} = \frac{1-\varepsilon}{k}N^{1/(1+2\varepsilon)}$ and $d = kM^{2\varepsilon} = kN^{2\varepsilon/(1+2\varepsilon)}$. Since $e < \frac{M}{k}$, it is clearly the case that $G = \mathsf{IDR}(G', \gamma)$ is (e_1, d_1)-reducible for $e_1 = \frac{M}{k} > e$ and $d_1 = d = kN^{2\varepsilon/(1+2\varepsilon)}$ as we delete more nodes and the depth reducibility guarantees the same upper bound of the remaining depth. On the other hand, $\mathsf{IDR}(G', \gamma)$ is (e_2, d_2)-depth robust for $e_2 = (1-\varepsilon)M = (1-\varepsilon)N^{1/(1+2\varepsilon)}$, while $d_2 = \gamma M^{1-\varepsilon} = (M^{2\varepsilon} - \delta)M^{1-\varepsilon}$. By Theorem 6, $\delta = \mathcal{O}\left(M^{\varepsilon}\log^2 M\right)$ so that for sufficiently large M, $d_2 = 0.9M^{1+\varepsilon} = 0.9N^{(1+\varepsilon)/(1+2\varepsilon)}$. ◀

▶ **Lemma 24.** *If G is (e,d)-reducible, then $\mathsf{superconc}(G)$ is $\left(e + \frac{N}{d}, 2d + \log(42N)\right)$-reducible, where N is the number of vertices in $\mathsf{superconc}(G)$.*

Proof. Let $G = (V, E)$ be a (e,d)-reducible DAG with N vertices. Let $G' = \mathsf{superconc}(G)$ and suppose G' has M vertices, which we designate $[M]$. Thus, there exists a set $S \subseteq V$ such that $|S| \leq e$ and $\mathsf{depth}(G-S) < d$. Let T be the set of $\frac{N}{d}$ vertices $\{M, M-d, M-2d, \ldots, M-N+d\}$, so that $T \subseteq \mathsf{output}(G')$. We claim $\mathsf{depth}(G'-S-T) < 2d + \log(42N)$.

Suppose by way of contradiction that there exists a path P in $G'-S-T$ of length at least $2d+\log(42N)$. By Lemma 10, the depth of any path from an input node to an output vertex is at most $\log(42N)$. Moreover, all edges added in the superconcentrator overlay are either between input vertices or two output vertices. Hence, then at least $2d$ vertices of P have to lie in either the first N vertices or the last N vertices of G'. Because P does not contain vertices of T, there is no path of length of length d in the last N vertices of G', so there must be a path of length d in the first N vertices of G', which contradicts $\mathsf{depth}(G-S) < d$.

Therefore, G' is $\left(e + \frac{N}{d}, 2d + \log(42N)\right)$-reducible. ◀

From Lemma 24 we immediately obtain an upper bound on the pebbling complexity of $\mathsf{superconc}(G)$ by applying Theorem 18 to Lemma 24. However, the upper bound is not as strong as we would like.

▶ **Corollary 25.** *Let G be an (e,d)-reducible graph with N vertices. Then*

$$\mathsf{cc}(\mathsf{superconc}(G)) \leq \min_{g \geq d}\left(e + \frac{N}{d}\right)42N + 2g(42N) + \frac{42N}{g}\left(2d + \log(42N)\right)42N.$$

Scheduling with Predictions and the Price of Misprediction

Michael Mitzenmacher
School of Engineering and Applied Sciences, Harvard University, Cambridge, MA, USA
michaelm@eecs.harvard.edu

Abstract

In many traditional job scheduling settings, it is assumed that one knows the time it will take for a job to complete service. In such cases, strategies such as shortest job first can be used to improve performance in terms of measures such as the average time a job waits in the system. We consider the setting where the service time is not known, but is predicted by for example a machine learning algorithm. Our main result is the derivation, under natural assumptions, of formulae for the performance of several strategies for queueing systems that use predictions for service times in order to schedule jobs. As part of our analysis, we suggest the framework of the "price of misprediction," which offers a measure of the cost of using predicted information.

2012 ACM Subject Classification Mathematics of computing → Queueing theory; Theory of computation → Online algorithms; Theory of computation → Scheduling algorithms

Keywords and phrases Queues, shortest remaining processing time, algorithms with predictions, scheduling

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.14

Funding *Michael Mitzenmacher*: This work was supported in part by NSF grants CCF-1563710 and CCF-1535795.

1 Introduction

While machine learning research seems to be growing at an exponential rate, there seems to be surprisingly little overlap with "traditional" algorithms and data structures and their analysis. Here we attempt to bridge this gap for the area of job scheduling, providing a general framework that may prove useful for additional problems.

Although we begin with settings with a finite number of jobs in order to provide insight into our approach, our main results are in the area of queueing systems. In this setting, we assume there is some algorithm (such as a neural network or other machine learning algorithm) that predicts the job time[1] upon entry; we model this predictor via a density function $g(x, y)$, so that $g(x, y)$ is the probability density for a job having actual service time x and predicted service time y. We emphasize that only the predicted service time is known to the system on the job's arrival, and we need not assume that the joint distribution is known in order to perform the scheduling. Rather, we use the joint distribution to derive equations for queue performance. This probabilistic model is sensible under the assumption that the true distribution of jobs is not changing over time, and the prediction quality is not changing over time; such assumptions frequently appear in settings utilizing machine learning.

In standard queueing theory, under standard assumptions such as Poisson arrivals, and independent service times, one can derive formulae for the behavior of many natural scheduling strategies, including *shortest job first* (SJF) and *shortest remaining processing time* (SRPT), which both minimize the average time a job spends in the system. (Shortest job first assumes

[1] We use the terms job time, service time, and processing time interchangeably; historically these different terms have all been used.

no preemption; shortest remaining processing time allows preemption.) In the setting where job times are predicted, we refer to the corresponding natural strategies as *shortest predicted job first* (SPJF) and *shortest predicted remanining processing time* (SPRPT). Our main result is to derive formulae for the expected time a job spends in the system for such strategies; the formulae can be computed in terms of the density function $g(x, y)$. We further provide some empirical evidence from simulations that even weak predictions can yield very good performance.

More generally, we consider the cost of using predictions in place of accurate job service times, and introduce the concept of the *price of misprediction* to describe this cost. Our results provide the price of misprediction for these basic strategies.

We emphasize that our goal here is *not* to develop specific prediction methods, and we do not do so in this work. Rather, our goal is to show that given a real or hypothetical prediction system matching our assumptions, we can develop equations for its performance. This general framework may apply to a variety of machine learning methods. In this way, we aim to extend traditional queueing theoretic formulations to the general setting where machine learning prediction systems are available. Just as queueing theoretic models and results have historically guided many real-world systems (see, e.g., [12, 16, 17] for background), our motivation stems from the idea that extending such models and results to setting with predictions will enhance the use of machine learning prediction in real-world systems that use queues. Indeed, we further hope that our approach may prove useful for the analysis of other traditional algorithms and data structures.

2 Related Work

While traditional algorithmic analysis focuses on worst-case algorithm behavior, there is a growing movement to develop frameworks that go beyond worst-case analysis [27]. While such frameworks have existed in the past, most notably via probabilistic analysis (e.g., [24]), semi-random models (e.g., [7, 9]), and smoothed analysis [31], one natural approach that has received little attention is the use of machine-learning-based approaches to provide predictions to algorithms, with the goal of realizing provable performance guarantees. (The idea of using machine learning to give hints as to which *heuristic* algorithm to employ has been considered in meta-heuristics for several large-scale problems, most notably for satisfiability [33]; this is a distinct line of work.)

Notable recent work with this theme is that of Lykouris and Vassilvitskii [20], who show how to use prediction advice from machine learning algorithms to improve online algorithms for caching in a way that provides provable performance guarantees, using the framework of competitive analysis. A series of recent papers consider the setting of optimization with noise, such as in settings when sampling data in order to obtain values used in an optimization algorithm for submodular functions [2, 14, 3, 4, 5, 26]. Other recent works analyze the performance of learned Bloom filter structures [19, 23], a variation on Bloom filters [6] that make use of machine learning algorithms that predict whether an element is in a given fixed set as a subfilter structure, and heavy hitter algorithms that use predictions [15]. One prior work in this vein has specifically looked at scheduling with predictions in the setting of a fixed collection of jobs, and considered variants of shortest predicted processing time that yield good performance in terms of the competitive ratio, with the performance depending on the accuracy of the predictions [25].

In scheduling for queues, some works have looked at the effects of using imprecise information, usually for load balancing in multiple queue settings. For example, Mitzenmacher considers using old load information to place jobs (in the context of the power of two

choices) [21]. A strategy called TAGS studies an approach to utilizing multiple queues when no information exists about the service time; jobs that run more than some threshold in the first queue are cancelled and passed to the second queue, and so on [11].

For single queues, the setting examined in this paper, Wierman and Nuyens look at variations of SRPT and SJF with inexact job sizes, bounding the performance gap based on bounds on how inexact the estimates can be [32]. Dell'Amico, Carra, and Michardi note that such bounds may be impractical, as outliers in estimating job sizes occur frequently; they empirically study scheduling policies for queueing systems with estimated sizes [8]. We note [8] points out there are natural methods to estimate job size, such as by running a small portion of the code in a coding job; we expect this or other inputs would be features in a machine learning formulation. Recent work by Scully and Harchol-Balter have considered scheduling policies that are based on the amount of service received, where the scheduler only knows the service received approximately, subject to adversarial noise, and the goal is to develop robust policies [29]. Our work differs from these past works in providing a model specifically geared toward studying performance with machine-learning based predictions, along with corresponding analyses. Finally, we note that our policies appear to fit within the more general framework of SOAP policies presented by Scully et al. [30] This provides an alternative approach to analyzing the policies studied here. We present derivations here based on the original analyses of SJF and SRPT, as we feel they are more instructive and straightforward.

3 Price of Misprediction

3.1 A Simple Example

To demonstrate our framework, we start with a simple example. Consider a collection of n jobs $j_1, \ldots, j_n$, each of one of two types, short or long. Short jobs require time s to process and long jobs require time ℓ to process, with $s < \ell$. Jobs are to be ordered and then processed sequentially. When the job times are known, shortest job first is known to minimize the average waiting time over all jobs. (Here and throughout waiting time is the time spent in the system before starting being served.) If there are n_s short jobs and n_ℓ long jobs the average waiting time is

$$\frac{1}{n}\left(n_s\frac{n_s-1}{2}s + n_\ell\frac{n_\ell-1}{2}\ell + n_\ell n_s s\right).$$

That is, on average each of the n_s jobs waits for half of the remaining $n_s - 1$ short jobs, and similarly for the long jobs, and the long jobs further have to wait for all of the short jobs.

We note that asymptotically if we drop lower order terms this is approximately

$$\frac{n_s^2 s + n_\ell^2 \ell + 2n_s n_\ell s}{2n}.$$

For simplicity, we will generally work with asymptotic expressions throughout.

If one has no information about the type of all of the jobs, the optimal policy (under the assumption that an adversary can present the jobs in a worst-case order) is to randomize the order of the jobs. In this case, using linearity of expectations, we can find the overall expected waiting time by finding the expected waiting time of each job. A simple calculation shows that if there are n_s short jobs and n_ℓ long jobs the expected waiting time is

$$\frac{1}{n}\left(n_s\left(\frac{n_s-1}{2}s + \frac{n_\ell}{2}\ell\right) + n_\ell\left(\frac{n_s}{2}s + \frac{n_\ell-1}{2}\ell\right)\right).$$

Asymptotically, this is approximately

$$\frac{n_s^2 s + n_\ell^2 \ell + n_s n_\ell (s+\ell)}{2n}.$$

Finally, in the prediction setting, when a job's type is predicted we assume short jobs are misclassified as long jobs with some probability p and long jobs are misclassified as short jobs with some probability q. We consider the policy of *shortest-predicted-job-first*; that is, we apply shortest-job-first based on the predictions. Our analysis requires multiple cases, as we must consider the expected waiting time for a job conditioned on whether it was classified correctly or incorrectly. Considering all of these four cases leads to the following expression for the expected waiting time when mispredictions occur:

$$\begin{aligned}
&\frac{1}{n}\Bigg((1-p)n_s\left(\frac{(1-p)(n_s-1)}{2}s + \frac{qn_\ell}{2}\ell\right)\\
&+ pn_s\left((1-p)(n_s-1)s + \frac{p(n_s-1)}{2}s + \frac{(1-q)n_\ell}{2}\ell + qn_\ell\ell\right)\\
&+ (1-q)n_\ell\left(\frac{(1-q)(n_\ell-1)}{2}\ell + q(n_\ell-1)\ell + \frac{pn_s}{2}s + (1-p)n_s s\right)\\
&+ qn_\ell\left(\frac{q(n_\ell-1)}{2}\ell + \frac{(1-p)n_s}{2}s\right)\Bigg).
\end{aligned}$$

Asymptotically, this is approximately

$$\frac{n_s^2 s + n_\ell^2 \ell + n_s n_\ell ((2-(p+q))s + (p+q)\ell)}{2n}.$$

This differs from the optimal (asymptotic) expression additively by $n_s n_\ell \frac{(p+q)(\ell-s)}{2n}$; it depends specifically on the "total error" $p+q$.

Following the standard terminology[2], we might refer to the ratio between the expected waiting time with imperfect information and the expected waiting time with perfect information as the *price of misprediction*. We propose the following definition:

▶ **Definition 1.** *Let $M_A(Q;I)$ be the value of some measure (such as the expected waiting time) for a system Q given information I about the system using algorithm A, and let $M_A(Q;P)$ be the value of that metric using predicted information P in place of I when using algorithm A. Then the price of misprediction is defined as $M_A(Q;I)/M_A(Q;P)$.*

We note that often the "price of" terminology is used to compare to an optimal algorithm, while here we are suggesting comparing to a (perhaps non-optimal) algorithm with perfect information. Of course one could also compare against optimal algorithms, as one does for online algorithms; see [20], for example.

In the example of short and long jobs above, the asymptotic price of misprediction for the waiting time is the ratio R given by:

$$R = \frac{n_s^2 s + n_\ell^2 \ell + n_s n_\ell ((2-(p+q))s + (p+q)\ell)}{n_s^2 s + n_\ell^2 \ell + n_s n_\ell (s+\ell)}$$

[2] The terms "price of anarchy" [18] and "price of stability" [1] are commonly used in game theoretic situations, and in particular in job scheduling, when multiple players act in their own self interest instead of cooperating. One could also view this as a multiplicative form of regret, but we think this terminology is more general and potentially helpful.

We can find where the ratio is maximized by considering $n_s = \gamma n$ for a constant γ. Some algebraic work yields that:

$$R \leq 1 + \frac{(p+q)(\sqrt{l/s}-1)}{2},$$

giving a bound on the price of misprediction. Note that the setting with no information, or a random ordering, is equivalent to the case $p = q = 1/2$, in which case R is bounded by $1 + (\sqrt{l/s}-1)/2$.

3.2 General Predictions

More generally, we can consider a setting where each job can be described as an independent random variable, where the random variable for a job is given by a density distribution $g(x, y)$; that is, $g(x, y)$ is the density function for a job, where a job has service time x and predicted service time y. We assume that $g(x, y)$ is "well-behaved" throughout this work, so that it is continuous and all necessary derivatives exist. (The analysis can be readily modified to handle point masses or other discontinuities in the distribution.)

It is convenient to let $f_s(x) = \int_{y=0}^{\infty} g(x, y)\, dy$ be the density function for the service time, and $f_p(y) = \int_{x=0}^{\infty} g(x, y)\, dx$ be the density function for the predicted service time. If there are n total jobs, the expected waiting time for a job using shortest job first given full information is given by

$$(n-1)\int_{x=0}^{\infty} f_s(x)\left(\int_{z=0}^{x} z f_s(z)\, dz\right) dx,$$

while the expected waiting time for a job using predicted information using shortest predicted job first is given by

$$(n-1)\int_{y=0}^{\infty} f_p(y)\left(\int_{x=0}^{\infty}\int_{z=0}^{y} x g(x, z)\, dz\, dx\right) dy.$$

In words, in the full information case, to compute the expected waiting time for a job, given its service time, we can determine the probability each other job has a smaller service time and the conditional value of that smaller service time to compute the waiting time. In the predicted information case, to compute the expected waiting time for a job given its predicted service time, we must determine the probability each other job has a smaller predicted service time and the conditional value of the actual service time of a job given that it has a smaller predicted service time to compute the waiting time.

Note that the factors of $n-1$ are cancelled in the ratio of the expected waiting times, and the function g suffices to determine to price of misprediction. As an example motivated by the prevalence of exponential distributions in queueing theory, suppose that service times are exponentially distributed with mean 1, and a job with service time x has a prediction that is distributed according to an exponential distribution with mean x. Then $g(x, y) = e^{-x-y/x}/x$, $f_s(x) = e^{-x}$, and $f_p(y) = \int_{x=0}^{\infty} \frac{e^{-x-y/x}}{x}\, dx$. We note $f_p(y)$ does not appear to have a simple closed form, though it is expressible in terms of Bessel functions.[3] In fact the price of misprediction can be shown to be exactly 4/3 using the integration features of Mathematica, and we can directly and formally prove it is exactly equal to 4/3 through a subtle argument allowing us to evaluate the corresponding integrals; this argument is given in the appendix.

[3] This was determined using the integration features of Mathematica 11.3.

In comparison, using no information and just scheduling in sequential order the expected waiting time is a factor of 2 worse than when using full information. This example, while not meant to match a real-world example,provides the right high-level intuition, in that it shows that even a weak predictor can yield significant improvements. Indeed, this is natural; for a predictor to work well in this setting, it simply has to order most of the jobs correctly in the queue.

The key here is that the price of misprediction can be computed (at least numerically) given the density distribution g. In practice, one might use this framework to determine the benefit of using a predictor; for example, one might seek to trade off the reduction in total waiting time with the cost of developing or using better prediction methods. While g may not be known exactly, we expect in practice good approximations for g can be determined empirically, which in turn will allow a good approximation for the price of misprediction or related quantities.

We note that, for suitably good prediction schemes, ordering by predicted service time should naturally correspond to ordering by the expected service time. That is, denote a job by (X, Y), where X is a random variable representing the true service time and Y is a random variable representing the predicted service time. Then suppose the density distribution g satisfies for any y_1, y_2 with $y_1 < y_2$ the natural inequality

$$\mathbf{E}[X \mid Y = y_1] < \mathbf{E}[X \mid Y = y_2].$$

In this case ordering by predicted service times yields an ordering according to expected service times.

We now extend these ideas to queueing theoretical models.

4 Single Queue Models

In this section, we present results providing formulae for prediction-based variants of shortest job first and shortest remaining processing time for single queue systems, which yield expressions for the price of misprediction. We briefly review the appropriate analysis methods for standard queues, starting with jobs with priorities, and then extend them prediction setting.

4.1 Priority-based Systems

Consider a queueing system with k types of jobs, $t_1, \ldots, t_k$. We assume Poisson arrivals, and that the arrival rate for type t_i is λ_i, with $\sum_{i=1}^{k} \lambda_i = \lambda$. A natural setting is that the ith type of job has service time q_i, with $q_1 < q_2 < \ldots < q_k$. In this case, with complete information about the service times, the shortest job first (SJF) strategy (without preemption) corresponds to a priority-based strategy, with the types corresponding to priorities; t_1 has the highest priority, and so on. More generally, the ith type of job may have a service time distribution, rather than a fixed service time. We describe this more general case, where job t_i has service distribution S_i; here the natural setting is $\mathbf{E}[S_1] < \mathbf{E}[S_2] < \ldots < \mathbf{E}[S_k]$, and if the types are prioritized by expected service time, the strategy is expected shortest job first (ESJF).

We describe some standard formula for priority systems, following the framework of [12]. Let $\rho_i = \lambda_i \mathbf{E}[S_i]$; this represents the load on the system from jobs of type i. Further, let $\rho = \sum_{i=1}^{k} \rho_i$, and $W(i)$ be the distribution of the waiting time in the queue for jobs

of type i in equilibrium. Also, let S be service time distribution of an incoming job, so $\mathbf{E}[S] = \sum_{i=1}^{k} \lambda_i \mathbf{E}[S_i]/\lambda$, and $\rho = \lambda \mathbf{E}[S]$. Then the following is known (see Equation (31.1) of [12]):

$$\mathbf{E}[W(i)] = \frac{\rho \mathbf{E}[S^2]}{2\mathbf{E}[S]\left(1 - \sum_{j=1}^{i} \rho_j\right)\left(1 - \sum_{j=1}^{i-1} \rho_j\right)}.$$

We now consider a system where types are not known but are predicted, for example according to a machine learning algorithm. For convenience, going forward, we refer to the *true type* of a job for its type, and refer to the machine prediction for a job as the *predicted type* where appropriate. We may represent the machine learning algorithm by a matrix M where m_{ij} is the probability that a job of true type i has predicted type j; here we are assuming that each job labelling can be treated as independent. In this case, let λ'_i be the arrival rate of jobs with predicted type i. Then

$$\lambda'_i = \sum_{j=1}^{k} \lambda_j m_{ji}.$$

Correspondingly, the distribution of service times for jobs having predicted type i is that the job has service time given by S_ℓ with probability

$$\lambda_\ell m_{\ell i} / \sum_{j=1}^{k} \lambda_j m_{ji}.$$

If we use S'_i to represent the distribution of service times for jobs having predicted type i, then

$$\mathbf{E}[S'_i] = \frac{\sum_{j=1}^{k} \lambda_j \mathbf{E}[S_j] m_{ji}}{\sum_{j=1}^{k} \lambda_j m_{ji}}.$$

Of course the expected service time S' over all jobs is

$$\mathbf{E}[S'] = \sum_{i=1}^{k} \lambda_i \mathbf{E}[S_i] = \mathbf{E}[S].$$

Assuming we prioritize jobs now according to their predicted type, we may again use the standard formula for priority systems. We derive the corresponding result. First, let

$$\rho'_i = \lambda'_i \mathbf{E}[S_i] = \sum_{j=1}^{k} \lambda_j \mathbf{E}[S_j] m_{ji}.$$

Also let $W'(i)$ be the distribution of the waiting time in the queue for jobs of predicted type i in equilibrium. Then

$$\begin{aligned}\mathbf{E}[W'(i)] &= \frac{\rho' \mathbf{E}[(S')^2]}{2\mathbf{E}[(S')]\left(1 - \sum_{j=1}^{i} \rho'_j\right)\left(1 - \sum_{j=1}^{i-1} \rho'_j\right)} \\ &= \frac{\rho \mathbf{E}[S^2]}{2\mathbf{E}[S]\left(1 - \sum_{j=1}^{i} \rho'_j\right)\left(1 - \sum_{j=1}^{i-1} \rho'_j\right)}.\end{aligned}$$

Hence, by summing over all possible types, we can see that the price of misprediction for the expected waiting time corresponds to the following expression:

$$\frac{\sum_{i=1}^{k} \lambda'_i \left(\left(1-\sum_{j=1}^{i} \rho'_j\right)\left(1-\sum_{j=1}^{i-1} \rho'_j\right)\right)^{-1}}{\sum_{i=1}^{k} \lambda_i \left(\left(1-\sum_{j=1}^{i} \rho_j\right)\left(1-\sum_{j=1}^{i-1} \rho_j\right)\right)^{-1}}.$$

4.2 Shortest Job First

We now show that the performance of shortest predicted job first, which we denote as SPJF, can be readily expressed as a limiting case of the priority analysis, similarly to how shortest job first is the limiting case of a priority queue based on service time. (Here we roughly follow the methodology of Section 31.3 of [12].) To start, we recall the formula for shortest job first; this is easily obtained as the limit of the priority system setting, where there are an infinitely many possible "priorities", and the priority corresponds to the service time. Here again let S be the service distribution of an incoming job. Further, let $f_s(x)$ be the corresponding density function, $\rho_x = \lambda \int_{t=0}^{x} t f_s(t) dt$, and $\rho = \lambda \int_{t=0}^{\infty} t f_s(t) dt$. We consider $W(x)$, the time spent waiting in the queue (not being served) for jobs with service time x in equilibrium. Then for standard shortest job first without preemption, where we know the exact service times without prediction, it is known that

$$\mathbf{E}[W(x)] = \frac{\rho \mathbf{E}[S^2]}{2\mathbf{E}[S]\left(1-\rho_x\right)^2}.$$

The overall expected time waiting in a queue, which we denote by $\mathbf{E}[W]$ where W is the waiting time in queue of an incoming job, is then simply

$$\mathbf{E}[W] = \int_{x=0}^{\infty} f(x)\mathbf{E}[W(x)]\, dx.$$

We now generalize this to SPJF. For a non-preemptive queue that uses a service time estimate, if $g(x, y)$ is the joint distribution that a job has service time x and predicted service time y, we again let $f_s(x) = \int_{y=0}^{\infty} g(x, y)\, dy$ be the density function for the service time, and $f_p(y) = \int_{x=0}^{\infty} g(x, y)\, dx$ be the density function for the predicted service time. We also let $\rho'_y = \lambda \int_{t=0}^{y} \int_{x=0}^{\infty} x g(x, t)\, dx\, dt$ to be the load on the system associated with jobs of predicted service time up to y. With the assumption that each job's service time characteristics are independently determined according to $g(x, y)$, if we let $W'(y)$ be the distribution of time spent waiting in the queue for a job with predicted service time y in equilibrium, then

$$\mathbf{E}[W'(y)] = \frac{\rho \mathbf{E}[S^2]}{2\mathbf{E}[S]\left(1-\rho'_y\right)^2},$$

where $W'(y)$ is the distribution of time in the queue for jobs with predicted service time y. Integrating over service time or predicted services time gives us that the price of misprediction is given by:

$$\frac{\int_{y=0}^{\infty} \frac{f_p(y)}{(1-\rho'_y)^2}\, dy}{\int_{x=0}^{\infty} \frac{f_s(x)}{(1-\rho_x)^2}\, dx}.$$

Let us again consider the example of service times that are exponentially distributed with mean 1, where a job with service time x has a prediction that is distributed according to an exponential distribution with mean x. Then the price of misprediction can be expressed as

$$\frac{\int_{y=0}^{\infty} \frac{\int_{x=0}^{\infty} \frac{e^{-x-y/x}}{x}\,dx}{(1-\lambda\int_{t=0}^{y}\int_{x=0}^{\infty} e^{-x-y/x}\,dx\,dt)^2}\,dy}{\int_{x=0}^{\infty} \frac{e^{-x}}{(1-\lambda(1-(x+1)e^{-x}))^2}\,dx}.$$

While there does not appear to be a simple closed form for this expression, it can be readily evaluated numerically for a given λ.

We note that a similar analysis can be used for preemptive shortest predicted job first (PSPJF), where a job may be preempted by another job that has an originally shorter predicted time (note that the time a job has been serviced is not considered). This is because preemptive shortest job first (PSJF) can be represented as the limit of a preemptive priority-based system (as in Section 32.3 of [12]), leading to a similar analysis. (We provide the analysis in the appendix.)

4.3 Shortest Remaining Processing Time

A more challenging variation involves extending the shortest remaining processing time (SRPT) policy to predictions. With complete information, SRPT maintains the remaining processing time for each job, and the job being processed can be preempted by an incoming job with service time smaller than the remaining processing time. To generalize to the prediction setting, we follow the framework of Schrage and Miller [28], who presented an analysis of SRPT. (See also [10] for a similar derivation, or [12] for an alternative.) Because the system is preemptive, it makes sense to consider the total time in the system, rather than the waiting time (as jobs may have further waits after they start being served). Because of the complexity of the expressions, we do not have a clean form for the price of misprediction, but they can be found from the derived formulae.

Before starting, we note a key issue in using SPRPT and how we describe the predicted remaining service time. Suppose that the original predicted service time for a job is y, but the actual service time is $x > y$. If the amount of service received by the job has been t, then the remaining service time is $x - t$, and it is natural to use $y - t$ as the predicted remaining service time. Of course, at some point we will have $t > y$, and the predicted remaining service time will be negative, which seems unsuitable.

Here we simply use $(y - t)^+ = \max(y - t, 0)$ as the predicted remaining service time. We recognize that this remains problematic; clearly the predicted remaining service time should be positive, and ideally would be a function $f(y, t)$. However, determining the appropriate function would appear to require some knowledge of the joint distribution $g(x, y)$; our aim here is to explore simple, general approaches that are agnostic to the underlying distribution g, such as SPRPT. We therefore leave the question of how to optimize the estimate of the predicted remaining time to achieve the best performance in this context as future work. It is worth noting, though, that [8] finds empirically that with sufficiently heavy-tailed service distributions and inaccurate predictions, large jobs whose service times are significantly underestimated can delay many short jobs, leading to very poor performance. They correspondingly suggest policies to deal with these situations, either sharing the processor among jobs in some cases, or suggesting PSJF over SPRPT.

We again use $g(x, y)$ for the joint distribution that a job has service time x and predicted service time y, and let $f_s(x) = \int_{y=0}^{\infty} g(x, y)\,dy$, $f_p(y) = \int_{x=0}^{\infty} g(x, y)\,dx$, $\rho_x = \lambda \int_{t=0}^{x} t f_s(t)\,dt$, and $\rho'_y = \lambda \int_{t=0}^{y}\int_{x=0}^{\infty} g(x, t)x\,dx\,dt$. The expected time in the system in equilibrium for a job

can be expressed as the sum of its residence time (time in the system once it has started receiving service) and it waiting time (time spent waiting before being served). For SRPT, a job with service time x has mean residence time

$$\int_{t=0}^{x} \frac{dt}{1-\rho_t};$$

one can think of this as the remaining service times drop from t to 0, at any instant there is a possible addition (given by the $1/(1-\rho_t)$ factor) due to preemptions.

The corresponding mean residence time for a job of service time x and predicted service time y under SPRPT is

$$\int_{t=0}^{x} \frac{dt}{1-\rho'_{(y-t)^+}}.$$

That is, here the predicted remaining processing time drops from y to $(y-t)^+ = \max(y-t, 0)$; it is possible the predicted remaining processing time is 0, but the job continues to require service, in which case we leave its predicted remaining processing time at 0, and it cannot be preempted. It follows that the mean residence time $\mathbf{E}[R(y)]$ for a job of predicted service time y is

$$\mathbf{E}[R(y)] = \int_{x=0}^{\infty} \frac{g(x,y)}{f_p(y)} \int_{t=0}^{x} \frac{dt}{1-\rho'_{(y-t)^+}}\, dx.$$

We now compute the waiting time, which is more difficult. The steady-state probability that an arriving job finds the server working on a job whose remaining predicted processing time is less than q is given by

$$b(q) = \rho'_q + \lambda \int_{t=q}^{\infty} \int_{x=0}^{\infty} g(x,t)(x-(t-q))^+\, dx\, dt.$$

The first term comes from arrivals with predicted service time less than q; the second term comes from jobs that start with predicted service time greater than q, but later their remaining predicted service time falls below q.

If $Y(q)$ is the length of a busy period where all jobs processed have predicted remaining processing times less than q, then the waiting time $W(q)$ for a job of predicted service time q is given by:

$$\mathbf{E}[W(q)] = b(q)\frac{\mathbf{E}[Y(q)^2]}{2\mathbf{E}[Y(q)]}.$$

To find the first two moments of $Y(q)$, we use the fact that the length of the busy period $Y(q)$ has the same distribution as the busy period for a first-come, first-served server where the job that initiates the busy period has processing time according to some distribution $Z(q)$, where additional jobs have a processing time distribution $X(q)$, and the arrivals are Poisson with rate $\lambda F_p(q)$, for $F_p(q) = \int_{x=0}^{q} f_p(x)dx$. We require the first two moments of $Z(q)$ and $X(q)$.

The moments for $X(q)$ are fairly straightforward, as $X(q)$ corresponds to the processing time of a job with predicted processing time at most q:

$$\mathbf{E}[X(q)] = \frac{1}{F_p(q)} \int_{t=0}^{q} \int_{x=0}^{\infty} g(x,t)x\, dx\, dt,$$

and

$$\mathbf{E}[X(q)^2] = \frac{1}{F_p(q)} \int_{t=0}^{q} \int_{x=0}^{\infty} g(x,t)x^2\, dx\, dt.$$

To determine the first two moments of $Z(q)$, we note that there are two ways a job can start the corresponding busy period. It either arrives when a busy period is not in progress and has predicted processing time at most q, or it is a job with predicted processing time greater than q (which starts a busy period when the predicted processing time reaches q). Note that in the second case, if the predicted processing time t is greater than q, but the actual processing time x is such that $x < t - q$, then the job cannot start a busy period, as the job will finish before the remaining predicted processing time reaches q. (Ideally, such situations should not occur with a suitably good predictor, but it must be taken into account.) Hence the probability a job initiates a corresponding busy period is

$$d(q) = (1 - b(q))F_p(q) + \int_{t=q}^{\infty}\int_{x=t-q}^{\infty} g(x,t)\,dx\,dt.$$

If we let (for typesetting reasons)

$$a_1(q) = (1 - b(q))\int_{t=0}^{q}\int_{x=0}^{\infty} g(x,t)x\,dx\,dt + \int_{t=q}^{\infty}\int_{x=t-q}^{\infty} g(x,t)(x-(t-q))\,dx\,dt$$

and

$$a_2(q) = (1 - b(q))\int_{t=0}^{q}\int_{x=0}^{\infty} g(x,t)x^2\,dx\,dt + \int_{t=q}^{\infty}\int_{x=t-q}^{\infty} g(x,t)(x-(t-q))^2\,dx\,dt$$

then

$$\mathbf{E}[Z(q)] = a_1(q)/d(q),$$

and

$$\mathbf{E}[Z(q)^2] = a_2(q)/d(q).$$

We now use the facts (see, e.g., Problem 49 of [10])

$$\mathbf{E}[Y(q)] = \frac{\mathbf{E}[Z(q)]}{1-\rho'_q}$$

and

$$\mathbf{E}[Y(q)^2] = \frac{\mathbf{E}[Z(q)^2]}{(1-\rho'_q)^2} + \lambda\mathbf{E}[Z(q)]F_p(q)\frac{\mathbf{E}[X(q)^2]}{(1-\rho'_q)^3}.$$

This yields

$$\mathbf{E}[W(q)] = b(q)\left(\frac{a_2(q)}{2a_1(q)(1-\rho'_q)} + \lambda F_p(q)\frac{\mathbf{E}[X(q)^2]}{2(1-\rho'_q)^2}\right).$$

The expected time in the system for a job is simply $\int_{y=0}^{\infty} f_p(y)\mathbf{E}[W(y)+R(y)]dy$. From this value (and the corresponding equations for standard SRPT) one can compute the price of misprediction for the total expected time in the system. (Of course, the expected service time can be subtracted if desired.)

Table 1 Results from simulations and equations for Shortest Job First (SJF) and Shortest Predicted Job First (SPJF).

λ	SJF Eqns	SJF Sims	SPJF Eqns	SPJF Sims	FIFO Eqns
0.5	1.7127	1.7128	1.7948	1.7949	2.00
0.6	1.9625	1.9625	2.1086	2.1087	2.50
0.7	2.3122	2.3121	2.5726	2.5730	3.33
0.8	2.8822	2.8828	3.3758	3.3760	5.00
0.9	4.1969	4.1987	5.3610	5.3609	10.00
0.95	6.2640	6.2701	8.6537	8.6541	20.00
0.98	11.2849	11.2734	16.9502	16.9782	50.00
0.99	18.4507	18.4237	29.0536	29.1162	100.00

5 Simulation Results

We present a small number of simulation results to demonstrate that our equations are accurate and, at least in the cases we have examined, the price of misprediction is generally reasonably small. Correspondingly, this implies that even a small amount of predictive power yields significantly better performance than standard First-In First-Out (FIFO) queueing. We focus on high load settings, as under low load all systems perform well. We also note that additional simulations we have performed further substantiate our high-level conclusions. (Our simulation was written in C and runs on a standard laptop.)

We first compare simulation results against the results from our equations; we also provide results for schemes with full information for comparison. Our results are for the setting with Poisson arrivals, service times are exponential with mean 1, and predicted service times are exponential with mean x when the actual service time is x. For consistency, we provide the total expected time in the system (waiting and service). The results of the equations were computed using Mathematica 11.3 and numerical integration. The calculations for SPRPT and PSPJF are somewhat lengthy and can lead to numerical stability issues; we found integrating up to predicted times of at most 50 gives accurate answers while being computable in reasonable time, approximately half an hour on a modern laptop. (Predicted service times greater than 50 are very rare; they occur with probability less than $5 \cdot 10^{-7}$.) We did not optimize the calculations and expect this could be improved. The results of simulations were from our own implementation of a queue simulator. The simulations are the results of averaging the average time over 1000 trials, where in each trial we recorded the time in system of each completed job. The trials were each run for 1 000 000 time units, with jobs completing in the first 100 000 time units discarded from the calculations of the averages to remove bias from starting with an empty system.

Table 1 shows both that the results from equations for SPJF match very closely to the simulation results, and that the performance is not too much worse than when the service times are known. With regards to accuracy, the difference is less than 1%, which appears due to simulation variance. With regard to performance, using predicted times naturally becomes increasingly worse as load grows, but the difference still shows the benefits of using imperfect information. Recall that, with no information, for standard queueing schemes such as FIFO the expected time in the system is $1/(1-\lambda)$; for example, $\lambda = 0.99$ leads to an expected time in the system of 100. We see that under high loads, the gains from prediction remain substantial.

Table 2 Results from simulations and equations for Preemptive Shortest Job First (PSJF) and Preemptive Shortest Predicted Job First (PSPJF).

λ	PSJF Eqns	PSJF Sim	PSPJF Eqns	PSPJF Sim	FIFO Eqns
0.5	1.5314	1.5312	1.6636	1.6634	2.00
0.6	1.7526	1.7524	1.9527	1.9521	2.50
0.7	2.0839	2.0841	2.3970	2.3963	3.33
0.8	2.6589	2.6594	3.1943	3.1940	5.00
0.9	4.0518	4.0521	5.2232	5.2235	10.00
0.95	6.2648	6.2688	8.6166	8.6118	20.00
0.98	11.5513	11.5212	17.1090	17.1211	50.00
0.99	18.9556	18.8717	29.3783	29.2907	100.00

Table 3 Results from simulations and equations for Shortest Remaining Processing Time (SRPT) and Shortest Predicted Remaining Processing Time (SPRPT).

λ	SRPT Eqns	SRPT Sim	SPRPT Eqns	SPRPT Sim	FIFO Eqns
0.5	1.4254	1.4251	1.6531	1.6588	2.00
0.6	1.6041	1.6039	1.9305	1.9397	2.50
0.7	1.8746	1.8757	2.3539	2.3684	3.33
0.8	2.3528	2.3519	3.1168	3.1376	5.00
0.9	3.5521	3.5486	5.04808	5.0973	10.00
0.95	5.5410	5.5466	8.3221	8.4075	20.00
0.98	10.4947	10.5003	16.6239	16.7852	50.00
0.99	17.6269	17.6130	28.7302	28.7847	100.00

Table 2 shows results for the same simulation setting using PSJF and PSPJF, with similar conclusions; indeed, in this case, the numerical results are very similar.

Table 3 shows similar results for the same simulation setting using SRPT and SPRPT. For SPRPT, the results from equations align a little less closely to the simulation results, but the difference remains than 1%. Given the complexity of the equations, and the higher variability in the time in system for SPRPT, this is unsurprising.

Figure 1 provides another example of prediction performance. Here we fix $\lambda = 0.95$, and consider two types of service distributions: exponential with mean 1, and a Weibull distribution with cumulative distribution $1 - e^{-\sqrt{2x}}$. (The Weibull distribution is more heavy-tailed, but also has mean 1.) The simulations are again the average of the measured time in system, averaged from results of 1000 trials, in the same manner as previously. Here the predictions depend on a scale parameter α; a job with service time x has a predicted service time that is uniform over $[(1-\alpha)x, (1+\alpha)x]$. By varying α, we can see the impact on performance as prediction accuracy diminishes. Note that when $\alpha = 0$ the predicted service time equals the true service time. In these examples, we observe that performance degrades gracefully with α, a feature we see across values of λ in other experiments not presented. The main point here is that even weak predictors may perform well under SPJF, PSPJF, and SPRPT; as long as they generally lead jobs to be processed in the right order, they can yield substantial benefits. (We note the standard deviation over trials ranges from 2-4%, with higher variance for simulations with the Weibull distribution.)

We do point out some specific features of note. First, SPJF and PSPJF obtain essentially the same performance with exponential service times, but are notably different with Weibull distributed service times. Second, for the Weibull distribution, PSPJF is much better than

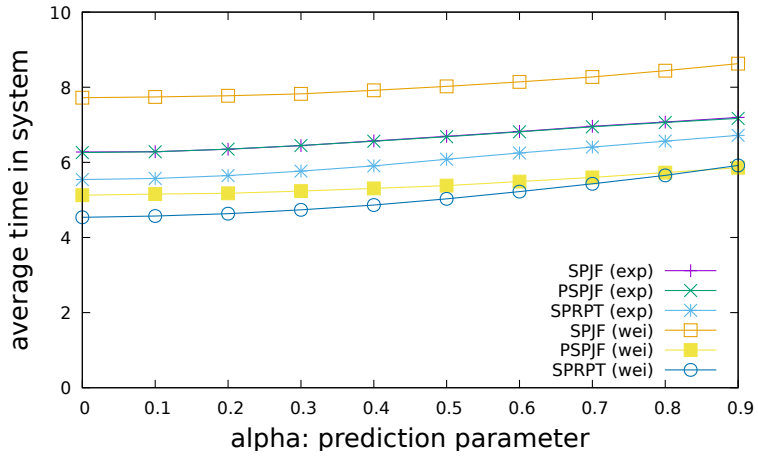


Figure 1 Results from simulations at $\lambda = 0.95$ for exponential and Weibull distributions. A job with service time x has predicted service time uniform over $[(1-\alpha)x, (1+\alpha)x]$. Performance degrades gracefully with α. Note $\alpha = 0$ corresponds to the full information case, as then the predicted service time equals the true service time.

SPJF, and even becomes slightly better than SPRPT with large prediction errors. Both of these feature are explained in large part by noting that the preemption of PSPJF is particularly helpful when large jobs are significantly underestimated, as in those settings without the preemption a large job can hold service for a long time, blocking shorter jobs; this has also been noted previously in [8]. Under the Weibull distribution for service, there are many much larger jobs than when the service times are exponentailly distributed. We expect our equations for the expected time in system for SPJF, PSPJF, and PSRPT may allow us to gain more insight into these kinds of behaviors, with provable results instead of simulation-based results.

6 Conclusion

We have demonstrated that the analyses of various single-queue job scheduling approaches can be generalized to the setting where predicted service times are used in place of true values, under the assumption that the predictions can be modeled as joint distribution with a corresponding density function. Such analyses can be used to determine the price of misprediction, or the potential benefits of better prediction, for such systems.

In future work, we plan to provide analyses of multiple queue systems using predicted service times. Multiple queue systems are quite common in practice, but can have more highly variable performance depending on how the workload is divided among queues. Natural strategies to consider include the power of two choices [22] and size interval task assignment (SITA) [13]. We expect analysis of such systems may require additional techniques, but will show that in this setting also even mildly accurate predictions can provide significant value.

In the problems considered here, we were able to determine *exact* formulae for performance, based on our probabilistic assumptions. It would be interesting to consider more general job scheduling scenarios with fewer assumptions, perhaps using methods more akin to online analysis, as in [20].

We believe this work suggests there is great potential in analyzing the large variety of job scheduling problems, as well as other similar traditional algorithmic problems, in the context of prediction.

References

1 Elliot Anshelevich, Anirban Dasgupta, Jon Kleinberg, Eva Tardos, Tom Wexler, and Tim Roughgarden. The price of stability for network design with fair cost allocation. *SIAM Journal on Computing*, 38(4):1602–1623, 2008.

2 Eric Balkanski, Aviad Rubinstein, and Yaron Singer. The power of optimization from samples. In *Advances in Neural Information Processing Systems*, pages 4017–4025, 2016.

3 Eric Balkanski, Aviad Rubinstein, and Yaron Singer. The limitations of optimization from samples. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1016–1027, 2017.

4 Eric Balkanski and Yaron Singer. The sample complexity of optimizing a convex function. In *Proceedings of the 30th Conference on Learning Theory*, pages 275–301, 2017.

5 Eric Balkanski and Yaron Singer. Approximation guarantees for adaptive sampling. In *Proceedings of the 35th International Conference on Machine Learning*, pages 393–402, 2018.

6 Burton H Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970.

7 Avrim Blum and Joel Spencer. Coloring random and semi-random k-colorable graphs. *Journal of Algorithms*, 19(2):204–234, 1995.

8 Matteo Dell'Amico, Damiano Carra, and Pietro Michiardi. PSBS: Practical size-based scheduling. *IEEE Transactions on Computers*, 65(7):2199–2212, 2015.

9 Uriel Feige and Robert Krauthgamer. Finding and certifying a large hidden clique in a semirandom graph. *Random Structures & Algorithms*, 16(2):195–208, 2000.

10 Natarajan Gautam. *Analysis of queues: methods and applications*. CRC Press, 2012.

11 Mor Harchol-Balter. Task assignment with unknown duration. *J. ACM*, 49(2):260–288, 2002.

12 Mor Harchol-Balter. *Performance modeling and design of computer systems: queueing theory in action*. Cambridge University Press, 2013.

13 Mor Harchol-Balter, Mark E Crovella, and Cristina D Murta. On choosing a task assignment policy for a distributed server system. *Journal of Parallel and Distributed Computing*, 59(2):204–228, 1999.

14 Avinatan Hassidim and Yaron Singer. Submodular optimization under noise. In *Proceedings of the 30th Conference on Learning Theory*, pages 1069–1122, 2017.

15 Chen-Yu Hsu, Piotr Indyk, Dina Katabi, and Ali Vakilian. Learning-based frequency estimation algorithms. In *International Conference on Learning Representations*, 2019.

16 Leonard Kleinrock. *Queueing systems, volume 1*. Wiley, New York, 1975.

17 Leonard Kleinrock. *Queueing systems, volume 2: Computer applications*. Wiley, New York, 1976.

18 Elias Koutsoupias and Christos Papadimitriou. Worst-case equilibria. In *Annual Symposium on Theoretical Aspects of Computer Science*, pages 404–413, 1999.

19 Tim Kraska, Alex Beutel, Ed H Chi, Jeffrey Dean, and Neoklis Polyzotis. The case for learned index structures. In *Proceedings of the 2018 International Conference on Management of Data*, pages 489–504, 2018.

20 Thodoris Lykouris and Sergei Vassilvitskii. Competitive caching with machine learned advice. In *Proceedings of the 35th International Conference on Machine Learning*, pages 3302–3311, 2018.

21 Michael Mitzenmacher. How useful is old information? *IEEE Trans. Parallel Distrib. Syst.*, 11(1):6–20, 2000.

22 Michael Mitzenmacher. The power of two choices in randomized load balancing. *IEEE Trans. Parallel Distrib. Syst.*, 12(10):1094–1104, 2001.

23 Michael Mitzenmacher. A model for learned bloom filters and optimizing by sandwiching. In *Advances in Neural Information Processing Systems*, pages 462–471, 2018.

24 Michael Mitzenmacher and Eli Upfal. *Probability and computing - randomized algorithms and probabilistic analysis*. Cambridge University Press, 2005.

25 Manish Purohit, Zoya Svitkina, and Ravi Kumar. Improving online algorithms via ML predictions. In *Advances in Neural Information Processing Systems*, pages 9684–9693, 2018.

26 Nir Rosenfeld, Eric Balkanski, Amir Globerson, and Yaron Singer. Learning to optimize combinatorial functions. In *Proceedings of the 35th International Conference on Machine Learning*, pages 4371–4380, 2018.

27 Tim Roughgarden. Beyond worst-case analysis. *arXiv preprint*, 2018. `arXiv:1806.09817`.

28 Linus E Schrage and Louis W Miller. The queue M/G/1 with the shortest remaining processing time discipline. *Operations Research*, 14(4):670–684, 1966.

29 Ziv Scully and Mor Harchol-Balter. SOAP bubbles: Robust scheduling under adversarial noise. In *Proceedings of the 56th Annual Allerton Conference on Communication, Control, and Computing*, 2018.

30 Ziv Scully, Mor Harchol-Balter, and Alan Scheller-Wolf. SOAP: One clean analysis of all age-based scheduling policies. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 2(1):16, 2018.

31 Daniel A Spielman and Shang-Hua Teng. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. *Journal of the ACM (JACM)*, 51(3):385–463, 2004.

32 Adam Wierman and Misja Nuyens. Scheduling despite inexact job-size information. In *ACM SIGMETRICS Performance Evaluation Review*, volume 36 (1), pages 25–36, 2008.

33 Lin Xu, Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. SATzilla: portfolio-based algorithm selection for SAT. *Journal of artificial intelligence research*, 32:565–606, 2008.

A Proof of $4/3$ price of misprediction for the finite case

We recall the setting where there are n jobs, service times are exponential with mean 1, the predicted service times are exponential with mean x when the actual service time is x, and we seek to determine the expected waiting time. Since we care only about the expectation, we may consider the expected waiting time with just a pair of jobs; linearity yields the price of misinformation is the same.

When using the correct service times, the expected waiting time of a job with the shortest job first is

$$\int_{x=0}^{\infty} e^{-x} \left(\int_{z=0}^{x} z e^{-z} \, dz \right) dx,$$

which is easily found to evaluate to $1/4$. When using predicted service times, the expected waiting time is

$$\int_{y=0}^{\infty} f_p(y) \left(\int_{x=0}^{\infty} \int_{z=0}^{y} e^{-x-z/x} \, dz \, dx \right) dy,$$

where $f_p(y) = \int_{x=0}^{\infty} \frac{e^{-x-y/x}}{x} \, dx$. This does not appear to evaluate to a closed form of simple functions. However, suppose we use as our prediction an exponentially distributed service time with mean $1/x$ instead of x. This effectively reverses the predicted order, but leads to an easier integral calculation. Since the expected waiting time for a job over both orderings is trivially 1, finding the expected waiting time for the reverse order suffices.

For the reversed order,

$$\begin{aligned} f_p(y) &= \int_{x=0}^{\infty} x e^{-x-yx} \, dx \\ &= \frac{1}{(y+1)^2}, \end{aligned}$$

and the integral becomes

$$\begin{aligned}
&\int_{y=0}^{\infty} \frac{1}{(y+1)^2} \left(\int_{x=0}^{\infty} \int_{z=0}^{y} x^2 e^{-x-xz}\, dz\, dx \right) dy \\
&= \int_{y=0}^{\infty} \frac{1}{(y+1)^2} \left(\int_{x=0}^{\infty} (xe^{-x} - xe^{-x-yx})\, dx \right) dy \\
&= \int_{y=0}^{\infty} \frac{1}{(y+1)^2} \left(\int_{x=0}^{\infty} (xe^{-x} - xe^{-x-yx})\, dx \right) dy \\
&= \int_{y=0}^{\infty} \left(\frac{1}{(y+1)^2} - \frac{1}{(y+1)^4} \right) dy \\
&= 2/3.
\end{aligned}$$

The expected waiting time where predictions are exponential with mean x is therefore $1/3$, and the price of misprediction is $4/3$ as claimed.

B Derivation for PSPJF

We consider the expected time a job spends in the system in equilibrium for preemptive shortest predicted job first (PSPJF), where a job may be preempted by another job that has an originally shorter predicted time (note that the time a job has been serviced is not considered). The analysis is similar to both SPJF and SPRPT.

Here we consider the expected waiting time and the expected residence time in steady-state. We again use $g(x, y)$ for the joint distribution that a job has service time x and predicted service time y, and let $f_s(x) = \int_{y=0}^{\infty} g(x, y)\, dy$, $f_p(y) = \int_{x=0}^{\infty} g(x, y)\, dx$, $F_p(y) = \int_{t=0}^{y} f_p(y)\, dt$, and $\rho'_y = \lambda \int_{t=0}^{y} \int_{x=0}^{\infty} g(x, t) x\, dx\, dt$. As a job will be preempted by another job with smaller predicted service time, the mean residence time for a job of service time x and predicted service time y is

$$\frac{x}{1 - \rho'_y}.$$

This is because the residence time with preemptions is the same as the busy period started by a job of length x and predicted length y, where the only jobs that need to be considered in the busy period have predicted length at most y. This leads to the additional $1/(1 - \rho'_y)$ factor.

It follows that the mean residence time $\mathbf{E}[R(y)]$ for a job of predicted service time y is

$$\mathbf{E}[R(y)] = \int_{x=0}^{\infty} \frac{x g(x, y)}{f_p(y)(1 - \rho'_y)}\, dx.$$

The expected waiting time for a job with predicted service time y is the same as for a shortest job first system, except that the job only waits for jobs of predicted service times as most y. It follows that

$$\mathbf{E}[W(y)] = \frac{\lambda \int_{t=0}^{y} \int_{x=0}^{\infty} x^2 g(x, t)\, dx\, dt}{2\left(1 - \rho'_y\right)^2}.$$

Note that here we have simplified the expression, which would originally have had a factor

$$\rho'_y \frac{\left(\int_{t=0}^{y} \int_{x=0}^{\infty} x^2 g(x, t)\, dx\, dt\right) / F_p(y)}{\left(\int_{t=0}^{y} \int_{x=0}^{\infty} x g(x, t)\, dx\, dt\right) / F_p(y)}.$$

The integral expressions are the second and first moments of the expected service time for a job with predicted service time at most y. As $\rho'_y = \lambda \int_{t=0}^{y} \int_{x=0}^{\infty} g(x,t)x \, dx \, dt$, the expression for $\mathbf{E}[W(y)]$ follows.

The expected time in the system for a job is then again simply $\int_{y=0}^{\infty} f_p(y)\mathbf{E}[W(y) + R(y)] \, dy$.

Reducing Inefficiency in Carbon Auctions with Imperfect Competition

Kira Goldner
Department of Computer Science, Columbia University, New York, NY, USA
http://www.kiragoldner.com/
kgoldner@cs.columbia.edu

Nicole Immorlica
Microsoft Research, New York, NY, USA
http://www.immorlica.com/
nicimm@microsoft.com

Brendan Lucier
Microsoft Research, Cambridge, MA, USA
https://www.microsoft.com/en-us/research/people/brlucier/
brlucier@microsoft.com

Abstract

We study auctions for carbon licenses, a policy tool used to control the social cost of pollution. Each identical license grants the right to produce a unit of pollution. Each buyer (i.e., firm that pollutes during the manufacturing process) enjoys a decreasing marginal value for licenses, but society suffers an increasing marginal cost for each license distributed. The seller (i.e., the government) can choose a number of licenses to put up for auction, and wishes to maximize the societal welfare: the total economic value of the buyers minus the social cost. Motivated by emission license markets deployed in practice, we focus on uniform price auctions with a price floor and/or price ceiling. The seller has distributional information about the market, and their goal is to tune the auction parameters to maximize expected welfare. The target benchmark is the maximum expected welfare achievable by any such auction under truth-telling behavior. Unfortunately, the uniform price auction is not truthful, and strategic behavior can significantly reduce (even below zero) the welfare of a given auction configuration.

We describe a subclass of "safe-price" auctions for which the welfare at any Bayes-Nash equilibrium will approximate the welfare under truth-telling behavior. We then show that the better of a safe-price auction, or a truthful auction that allocates licenses to only a single buyer, will approximate the target benchmark. In particular, we show how to choose a number of licenses and a price floor so that the worst-case welfare, at any equilibrium, is a constant approximation to the best achievable welfare under truth-telling after excluding the welfare contribution of a single buyer.

2012 ACM Subject Classification Theory of computation → Algorithmic game theory and mechanism design; Theory of computation → Computational pricing and auctions

Keywords and phrases welfare, price of anarchy, mechanism design, equilibrium, costs

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.15

Related Version Full version: http://www.kiragoldner.com/papers/carbonauctions.pdf

Funding *Kira Goldner*: Supported in part by NSF CCF-1420381 and by a Microsoft Research PhD Fellowship. Supported in part by NSF award DMS-1903037 and a Columbia Data Science Institute postdoctoral fellowship.
Nicole Immorlica: Supported in part by NSF Award 1841550, "EAGER: Algorithmic Approaches for Developing Markets."

1 Introduction

Licenses for carbon and other emissions are a market-based policy tool for reducing pollution and mitigating the effects of climate change. Roughly speaking, a government agency distributes pollution licenses to firms according to some mechanism. At the end of a period of time (e.g., a year), firms must submit licenses to cover their pollution or else face severe penalties. Different ways of distributing licenses are possible. For example, if licenses are simply sold at a fixed price to anyone who wishes to pay, then this is equivalent to a *carbon tax* where polluters must pay a linear fee to offset their emissions. Alternatively, in a *cap-and-trade* mechanism, the agency releases a fixed number C of pollution licenses, either via a pre-determined allocation at no price (e.g., based on prior pollution or industry averages) or via auction, and then polluters can trade these licenses on the open market. Many emission trading systems take features of both schemes, combining a cap-and-trade market with a *price floor* $\underline{p}$, where each license must be sold above the reserve price of $\underline{p}$, and/or a *price ceiling* $\overline{p}$, where an extra license beyond the cap can always be purchased at a price of $\overline{p}$. In general, such a pollution license market is referred to as an Emission Trading System (ETS).

There are many important ETS in effect today. The EU ETS has thousands of participating firms and has raised billions of dollars in auction revenue over the past 10 years [2]. The Western Climate Initiative runs a license auction that is linked between California and Quebec [4, 1], and the Regional Greenhouse Gas Initiative (RGGI, pronounced "Reggie") serves New England and the New York region [3]. These markets differ in the details of their implementations, but in each case, licenses are distributed on a regular schedule, with a significant quantity of those licenses sold at auction.

One can model such a license auction as a multi-unit auction; that is, an auction for multiple identical goods. There is no bound on the supply of goods, but there is a cost of production corresponding to the social cost of more pollution. The social cost is typically assumed to be increasing and convex, and the value for licenses for each buyer is commonly assumed to be increasing and concave [27]. Each of the ETS described above uses a uniform-price auction rule to resolve this multi-unit auction. Such auctions proceed roughly as follows. Participating firms declare bids, which take the form of a non-decreasing concave function that describes their willingness-to-pay for varying quantities of licenses. This can alternatively be viewed as a list of non-increasing marginal bids for each successive license. Any marginal bids below the reserve price $\underline{p}$ are removed, and licenses are then distributed to the highest remaining marginal bids while supplies last. Each firm then pays a fixed price p per license, where p is set to some value between the lowest marginal winning bid and the highest marginal losing bid. Such auctions are not truthful, but are common in practice due to their many advantageous properties; see, e.g., Chapter 7 of [22].

Implementing a uniform-price auction for carbon licenses presents an optimization challenge: good outcomes require that the system designer correctly sets the quantity of licenses to distribute and/or the price (either direct price or auction reserve) at which they will be sold. The goal is to maximize *social welfare*: the aggregate value that firms receive for their licenses (i.e., by producing goods) minus the externality on society caused by polluting the corresponding amount during production. This optimization problem is complicated by uncertainty. Even if the social cost of pollution is fully known, the designer may not know what the demand for licenses will be, which makes it hard to predict the optimal level of pollution to allow. The goal of the designer, then, is to set the parameters of the uniform price auction to maximize the expected outcome over uncertainty in the market. Notably, the presence of social costs significantly increases the complexity of multi-unit auctions,

since efficiency becomes a mixed-sign objective. Even a slight misallocation of the licenses, resulting in a small reduction of received value, might have a disproportionately large effect on net welfare. In practice, inefficient allocations may be partially resolved in the trade phase of cap-and-trade systems. However, there are significant trading frictions, so it is imperative to choose an initial allocation, via the auction, that is as efficient as possible.

Since uniform price auctions are not truthful, one must account for strategic behavior of participating firms. A notable feature of ETS markets in practice is that, despite their size, they commonly have a small number of participants with significant market power.[1] The presence of dominant players suggests imperfect competition and raises the issue of strategic manipulation, in which large individual firms try to influence the market in their favor. One particular concern is demand reduction, where firms reduce their bids to suppress the price determined by the auction. We ask: how should one set auction parameters to (approximately) maximize welfare, in the face of strategic bidding?

1.1 Our Results

In this paper we apply ideas from theoretical computer science to approach the design of carbon license auctions with strategic firms. In this new domain, we use facts regarding uniform price auctions to understand what is happening in these markets and probabilistic analysis to handle the uncertainty of valuation realizations. We then apply a Price of Anarchy analysis to give strategic guarantees. As a result, we provide a concrete recommendation for how to set the parameters of the mechanism used in practice – a recommendation with provable approximation guarantees.

A market instance is described by a convex social cost curve and, for each participating firm, an independent distribution over concave valuations. To capture the relevant space of auctions, we formalize a class of allocation rules that we call *cap-and-price auctions*.[2] These auctions are parameterized by a maximum number of licenses to allocate, a price floor, and a price ceiling. They proceed by running a uniform-price auction for the given number of licenses, subject to the price floor and ceiling. Given a market instance, there is some choice of cap-and-price auction that maximizes the expected welfare generated under non-strategic (truth-telling) behavior. This optimal non-strategic solution will serve as a benchmark, which we denote OPT. Our main result is an auction that approximates OPT at any equilibrium of strategic behavior.

We show that, in general, the welfare obtained at an equilibrium of a cap-and-price auction can be negative, even for the auction that optimizes welfare under non-strategic bidding. This motivates our main question: given a cap-and-price mechanism that achieves a certain expected welfare W under truth-telling behavior, can one modify the auction parameters so that the expected welfare is an $O(1)$ approximation to W at any Bayes-Nash equilibrium? To answer this question, we describe a subclass of cap-and-price auctions that use *safe prices*. A *safe-price auction* is a cap-and-price auction in which the price floor is at least the average social cost, per license, of selling all of that auction's licenses. We show that for any such auction the worst-case welfare at any Bayes-Nash equilibrium is within a constant factor of the welfare under truth-telling. This result proceeds by transforming the market instance into a corresponding instance without social costs, and makes use of bounds on the Bayes-Nash price of anarchy for uniform-price auctions [20, 13].

[1] For example, in the EU ETS, the top 10 firms together control approximately 30 percent of all licenses allocated and traded; see [9], page 127.

[2] Note, we are focused on the auction phase in this paper and do not model the trade phase of these systems.

To construct an auction that approximates OPT at equilibrium, it therefore suffices to find a safe-price auction whose non-strategic welfare approximates that of the best unconstrained cap-and-price auction. However, as it turns out, there are market instances for which no constant approximation by a safe-price auction is possible. Such a situation can occur if there is a single firm that drives most of the demand for licenses, and this demand has very high variance. We show that this is essentially the only barrier to our desired price of anarchy result: one can always construct an auction that achieves a constant approximation to OPT minus the welfare obtainable by allocating licenses to any one single auction participant. In other words, as long as no single firm accounts for most of the expected welfare of the carbon license market, one can find a cap and price floor so that the welfare at any equilibrium is within a constant of the optimal outcome achievable without strategic behavior. We argue that this assumption is reasonable, since presumably the motivation for running a carbon license market in the first place is that the demand for emission licenses is distributed across multiple firms. An alternative way to view this result is that one can always achieve a constant approximation to our benchmark by either running a safe-price auction, or by selling to only a single firm.

Along the way, we also show that for any cap-and-price auction that sells all of its licenses with probability $q > 0$, there is a safe-price auction whose welfare (under truth-telling) is within a factor $O(1/q)$ of the original auction's welfare. In particular, if our benchmark is implemented by a cap-and-price auction that distributes all of its licenses with constant probability, then we can construct a safe-price auction whose welfare at any Bayes-Nash equilibrium is a constant approximation to the benchmark.

Note that our results are applicable for any setting with imperfect competition where a uniform price auction is used to allocate goods with a production or social cost. For instance, they may apply to allocating queries to a database, where the social cost is privacy. Another example might be allocating medallions to taxi and rideshare drivers in New York City, where the social cost is congestion.

1.2 Related Work

Our work is related to rich line of literature in economics comparing emission licence auction formats and flat "carbon tax" pricing methods to control the externalities of pollution. Weitzman [27] proposed a model of demand uncertainty and initiated a study comparing price-based vs quantity-based screening in the context of pollution externalities. Kerr and Cramton [10] noted that auctions tend to generate more efficient outcomes than grandfathered contracts (i.e., pre-determined allocations based on prior usage), which distort incentives to reduce pollution and efficiently redistribute licenses. Cramton, McKay, Ockenfels and Stoft [11] subsequently lay out arguments in favor of tax-based approaches. Murray, Newell, and Pizer [23] analyze the use of price ceilings in emission license auctions, and argue that they provide benefits of both auction-based and tax-based systems, improving efficiency in dynamic markets with intertemporal arbitrage. For a recent overview of auction-based systems used in practice, from both the economic and legal perspectives, we recommend [9].

A similar line of literature studied alternative approaches to the related electricity markets, where individual providers sell electricity into a central grid. Whereas the main issue in emission license sales is the social externality of production, the main focus in electricity markets was incentivizing participation of small firms. The primary discussion focused on using uniform pricing versus discriminatory pricing in the resulting procurement auction [12, 24].

Within the theoretical computer science community, there have been numerous studies of the efficiency of auction formats at equilibrium. For multi-unit auctions without production costs (or, equivalently, no social cost of pollution), the price of anarchy of the uniform-price auction was shown to be constant for full-information settings, and this was subsequently extended to Bayes-Nash equilibria [20, 13]. Our work can be seen as an extension of that work to the setting with a convex cost of allocation. Auctions with production costs have also been studied [6, 15], but primarily from the perspective of mechanism design, where the goal is to develop allocation and payment rules to achieve efficient outcomes. In the present work we do not take a mechanism design approach; we instead restrict our attention to (non-truthful) uniform price auctions, as these are used in practice, and study bounds on the worst-case welfare at equilibrium under different choices of the auction parameters.

Uniform-price auctions with costs can be viewed as games for which the designer has a mixed-sign objective (i.e., total value generated minus the social cost of production). Prior work on the price of anarchy under mixed-sign objectives has focused primarily on routing games [25, 7, 8]. Results in this space tend to be negative, motivating alternative measures of performance (such as minimizing a transformed measure of total cost) that avoid the pitfalls of mixed-sign optimization. Our work shows that in an auction setting, use of an appropriately-chosen thresholding rule (in the form of a reserve price) can enable a constant approximation to our mixed-sign objective of total value minus a convex social cost.

There is a vast amount of work on auctions with externalities, such as the seminal work of [16], and other work including externalities in advertising auctions [14], characterizations of equilibria in auctions with externalities [19], and more. However, these externalities are private and held by the buyers, as opposed to public and suffered by the seller as in this paper. In this case, the externality functions more like a production cost, as described above.

Kesselheim, Kleinberg and Tardos [17] study the price of anarchy of an energy market auction, which is a similar application to the carbon license auction that we study. Their focus is on the uncertainty of the supply and temporal nature of the auction, and not on externalities of production, and hence their technical model is quite different.

The theory of bidding in uniform-price auctions with imperfect competition is well-developed in the economic theory literature. Much of this work focuses specifically on the efficiency and revenue impact of demand reduction, and how modifications to the auction format or context might impact it. Demand reduction occurs at equilibrium even in very simple settings of full information, can dramatically reduce welfare, and is a concern in practice [28, 26, 5]. One way to reduce the inefficiency of demand reduction is to perturb the supply, either by allowing the seller to adjust the supply after bids are received [21], or by randomizing the total quantity of goods for sale (or otherwise smoothing out the allocation function) [18]. Such results are typically restricted to full-information settings. Moreover, these approaches are not necessarily appropriate in the sale of government-issued licenses, where one typically expects commitment and certainty about the quantity being sold. In contrast, we forego a precise equilibrium analysis and instead argue that setting a sufficiently high price floor can likewise mitigate the impact of demand reduction.

1.3 Roadmap

Our main result is a cap-and-price auction whose welfare in equilibrium approximates the maximum welfare of a cap-and-price auction with non-strategic reporting in markets without a single dominant bidder. In Section 2, we introduce our model and formally define cap-and-price auctions. In Section 3, we first show that we can restrict attention to cap-and-price auctions with an infinite price ceiling (i.e., ones who never sell more licenses than the cap).

We then derive a class of such cap-and-price auctions, which we call safe-price auctions, whose performance in equilibrium approximates the performance in a non-strategic setting. Thus it suffices to show that the best safe-price auction approximates the best cap-and-price auction in a non-strategic setting. However, this is not true: we give an example where it can be unboundedly worse. In Section 4, we demonstrate that the only barrier to this is the existence of a dominant bidder, yielding our main result.

2 Preliminaries

There are n firms seeking to purchase carbon licenses. Licenses are identical, and each permits one pollution unit. Each firm i has a monotone non-decreasing concave *valuation curve* $V_i(\cdot)$ that maps a number of pollution units $x \in \mathbb{Z}_{\geq 0}$ to a value $\mathbb{R}_{\geq 0}$; this is the value they enjoy for polluting this amount. The profile of valuation curves $\mathbf{V} = (V_1, \ldots, V_n)$ is drawn from a publicly-known distribution F over profiles, where the draw of $V_i(\cdot)$ is the private information of firm i. Valuations are drawn independently across firms, so $F = F_1 \times \ldots \times F_n$ is a product distribution and $V_i \sim F_i$.

There is a publicly known monotone non-decreasing convex *cost function* $Q(\cdot)$ that maps a number of pollution units $x \in \mathbb{Z}_{\geq 0}$ to the value of externality that society faces for having this much pollution. Given a valuation profile $(V_1, \ldots, V_n)$, the *welfare* of a given allocation rule $\mathbf{x} = (x_1, \ldots, x_n)$ is $\sum_i V_i(x_i) - Q(\sum_{i=1}^n x_i)$. Our objective is to maximize expected welfare.

Given some integer $x \geq 0$, we write $V(x)$ for the maximum aggregate value that could be obtained by optimally dividing x licenses among the firms. That is,

$$V(x) = \max_{\vec{y} \in \mathbb{Z}_+^n : ||y||_1 = x} \sum_i V_i(y_i).$$

We refer to V as the *combined valuation curve*. As each $V_i(\cdot)$ is weakly concave, so is $V(\cdot)$. We will sometimes abuse notation and write $V \sim F$ to mean that V is distributed as the aggregate value when $(V_1, \ldots, V_n) \sim F$. We'll write $W(V, x) = V(x) - Q(x)$ for the welfare generated by allocating x licenses optimally among the firms.

We'll write $v_i(j) = V_i(j) - V_i(j-1)$ for firm i's marginal value for aquiring license j, for each $j \geq 1$. By the monotonicity and concavity of $V_i(\cdot)$, $v_i(\cdot)$ is non-increasing. We'll also write $V_i(j|k) = V_i(j+k) - V_i(k)$ for the marginal value of j additional items given k already allocated. We'll write $d_i(p) = \max\{j : v_i(j) \geq p\}$ for the number of units demanded by bidder i at price p.

We will study equilibria and outcomes of license allocation auctions. An auction takes as input a reported valuation V_i from each firm i. The auction then determines an allocation $\mathbf{x} = (x_1, \ldots, x_n)$ and a price $p_i \geq 0$ that each firm must pay. The auctions we consider will be uniform price auctions, where the auction determines a per-license price p and each firm i pays $p_i = p \cdot x_i$. Given an implicit uniform-price auction, we will tend to write $x_i(\tilde{\mathbf{V}})$ (resp., $p(\tilde{\mathbf{V}})$) for the allocation to agent i (resp., per-unit price) when agents report according to $\tilde{\mathbf{V}}$. For a given valuation profile $\mathbf{V}$, we'll also write $U_i(\tilde{\mathbf{V}})$ for the utility enjoyed by firm i when agents bid according to $\tilde{\mathbf{V}}$:

$$U_i(\tilde{\mathbf{V}}) = V_i(x_i(\tilde{\mathbf{V}})) - p(\tilde{\mathbf{V}}) \cdot x_i(\tilde{\mathbf{V}}).$$

Finally, given an auction M and a distribution F over input profiles, we will write $W(M, F)$ for the expected welfare of M on input distribution F. We will sometimes drop the dependence of F and simply write $W(M)$ when F is clear from context. We emphasize that $W(M)$ is a

non-strategic notion of welfare; it is the expected welfare of M with respect to *inputs* drawn from F, or equivalently the welfare of M under truthful reporting when agent valuations are distributed according to F. Let $\mathbf{x}^M(V) = (x_1^M(V), \ldots, x_n^M(V))$ be the allocation rule of mechanism M for realization V. Then

$$W(M) = \mathbb{E}_{V \sim F}\left[\sum_i V_i\left(x_i^M(V)\right) - Q\left(\sum_i x_i^M(V)\right)\right].$$

A *Bayes-Nash equilibrium* of a uniform-price auction is a choice of bidding strategies $\sigma = (\sigma_1, \ldots, \sigma_n)$ for each agent, mapping each realized valuation curve V_i to a (possibly randomized) reported valuation $\sigma_i(V_i)$, so that each agent maximizes their expected utility by bidding according to σ_i given that other agents are bidding according to σ_{-i}. That is, for all V_i and all $\tilde{V}_i$, we have

$$\mathbb{E}_{V_{-i} \sim F_{-i}}[U_i(\sigma_i(V_i), \sigma_{-i}(V_{-i}))] \geq \mathbb{E}_{V_{-i} \sim F_{-i}}[U_i(\tilde{V}_i, \sigma_{-i}(V_{-i}))].$$

We make a standard assumption of *no overbidding* on behalf of the firms uniform-price auction, which is motivated by the fact that overbidding is a weakly-dominated strategy.

Motivated by license auctions used in practice, we study a particular form of uniform price auction that we call a *cap-and-price auction* which forces prices to be within some fixed interval.[3]

▶ **Definition 1.** *A* cap-and-price auction $M(C, \underline{p}, \overline{p})$ *is parameterized by a quantity* $C \geq 1$*, a price floor* $\underline{p}$*, and a price ceiling* $\overline{p} > \underline{p}$*. The allocation and price are determined as follows:*

1. *If* $\sum_i d_i(\overline{p}) \geq C$*, then* $x_i = d_i(\overline{p})$ *for all* i *and* $p = \overline{p}$*.*
2. *If* $\sum_i d_i(\underline{p}) < C$*, then* $x_i = d_i(\underline{p})$ *for all* i *and* $p = \underline{p}$*.*
3. *Otherwise, choose* $p = V(C) - V(C-1)$*, the* C^{th} *highest bid, and choose* $\vec{x}$ *to be any optimal allocation of* C *licenses among the firms:* $x \in \arg\max_{\vec{y} \in \mathbb{Z}_+^n : ||y||_1 = C} \sum_i V_i(y_i)$*.*

We can think of this as a uniform price auction of up to C licenses (case 3 above), where the price is set to the lowest winning marginal bid, with two modifications. First, there is a reserve price $\underline{p}$, so that possibly fewer than C licenses are sold if there is not enough demand at price $\underline{p}$; this is case 2. Second, if the lowest winning marginal bid would be larger than $\overline{p}$, then the price is lowered to $\overline{p}$ and firms can purchase as many licenses as they like at this price; this is case 1.

Given social cost function Q and valuation distribution F, we will write OPT for the optimal expected welfare obtained by any cap-and-price auction under truthful reporting. That is, $\text{OPT} = \max_{C, \underline{p}, \overline{p}}\{W(M(C, \underline{p}, \overline{p}))\}$. We will also tend to write C^*, $\underline{p}^*$, and $\overline{p}^*$ for the parameters that achieve this maximum. We emphasize that this is a non-strategic notion: OPT is the maximum expected welfare attainable when bidders report truthfully. We view OPT as a benchmark against which we will compare performance at Bayes-Nash equilibrium. Note also that, in general, OPT may be strictly less than the expected welfare of the unconstrained welfare-optimal allocation; an example is provided in Appendix C.

3 Safe-Price Auctions

We will derive a class of cap-and-price auctions, which we will call safe-price auctions, whose performance in equilibrium approximates their non-strategic welfare. The hope is that the non-strategic welfare of this class then approximates the non-strategic welfare of the larger class of cap-and-price auctions.

[3] Note the cap is not a hard constraint, but rather governs which prices bind.

▶ **Definition 2.** *A* safe-price auction *is a cap-and-price auction $M(C, \underline{p}, \overline{p})$ parameterized by a license quantity C, a price floor $\underline{p} = Q(C)/C$, and price ceiling $\overline{p} = \infty$. We will write $M(C)$ for the safe-price auction with license quantity C, and $P(C) = Q(C)/C$ for the associated safe price.*

This definition restricts cap-and-price auctions in two ways. The first is that the price ceiling is now infinite. The second is that we have imposed a lower bound on the price floor.[4] The first restriction is for convenience; the following lemma shows that it does not cause much loss in welfare. Motivated by this lemma, we will restrict our attention from now on to auctions with no price ceiling, that is, $\overline{p} = \infty$. We notate such mechanisms $M = (C, \underline{p})$. Note that, in the language of definition 1, case (1) can no longer occur. That is, when is there is no price ceiling, only exactly C or less than C licenses will be allocated.

▶ **Lemma 3.** *For any distribution F and any cap-and-price auction $M(C, \underline{p}, \overline{p})$, there exist a cap C' and price floor $\underline{p}'$ such that $W(M(C', \underline{p}', \infty)) \geq \frac{1}{2} W(M(C, \underline{p}, \overline{p}))$.*

The proof of Lemma 3 appears in Appendix A.1. The main idea is to decompose the expected welfare of a cap-and-price auction $M(C, \underline{p}, \overline{p})$ into the welfare attained from the first (at most) C licenses sold plus the incremental welfare attained from any licenses sold after the first C. The first can be approximated by $M(C, \underline{p}, \infty)$; the latter, if non-negative, can be approximated by $M(\infty, \overline{p}, \infty)$.

The second way safe-price auctions restrict price-and-cap auctions is with a lower bound on the price floor. As the following example shows, the equilibrium welfare of cap-and-price auctions, even those with an infinite price ceiling, suffer from a problem known as demand reduction in which a bidder improves her price, and hence utility, by asking for fewer units. This can have drastic consequences on welfare, causing it to become negative, due to the existence of the social cost. In particular, this means the approximation of such auctions, relative to the best welfare attainable in non-strategic settings, is unbounded.

▶ **Example 4.** Consider two agents, 1 and 2. Their distributions over valuations will be point-masses, so that the valuations are actually deterministic. These valuation functions are given by the following marginals: $v_1(1) = 10$, $v_1(2) = 10$, $v_2(1) = 6$, $v_2(2) = 1$, and all other marginals are 0. We will have $C = 2$, $\underline{p} = 0$, $\overline{p} = \infty$, and the social cost function is given by $Q(x) = 9x$. Under truthful reporting, the auction $M(C, \underline{p})$ allocates 2 licenses to agent 1 at a price of 6 each, resulting in a welfare of $V_1(2) - Q(2) = 20 - 18 = 2$. However, we note that firm 1 could improve their utility from 8 to 9 by instead reporting a modified valuation $\tilde{V}$ given by $\tilde{v}_1(1) = 10$ and $\tilde{v}_1(2) = 1$. If agent 2 continues to report truthfully, $M(C, \underline{p})$ will allocate 1 license to each agent at a price of 1 each, resulting in a welfare of $V_1(1) + V_2(1) - Q(2) = 16 - 18 = -2$. One can verify that this is indeed a pure Nash equilibrium of the auction, and hence a Bayes-Nash equilibrium as well.

The issue illustrated in Example 4 is that strategic behavior can cause licenses to be allocated to agents whose marginal values for these licenses are below $Q(C)/C$, causing the aggregate value derived from allocating C licenses to fall below $Q(C)$. Safe-price auctions seek to prevent this by imposing a sufficiently high price floor.

Indeed, the following lemma shows that price floors do indeed circumvent the issue in the above example. Namely, we show safe-price auctions have good equilibria, compared to their own non-strategic welfare. This is the main result of this section, and motivates us to focus on analyzing the non-strategic welfare of safe-price auctions.

[4] In fact we set $\underline{p}$ to be *equal* to $P(C)$, but our results still hold if this is relaxed to $\underline{p} \geq P(C)$.

▶ **Lemma 5.** *For any safe-price auction $M(C)$, the expected welfare at any Bayes-Nash equilibrium is at least $(\frac{1}{3.15}) \cdot W(M(C))$.*

Proof. Fix a license cap C and write $\underline{p} = Q(C)/C$ for the corresponding safe price, and recall that $M(C) = M(C, \underline{p})$. We claim that this auction is strategically equivalent to a modified auction, as follows. First, since each agent pays at least $\underline{p}$ per license, we can define $\hat{v}_i(j) = v_i(j) - \underline{p}$ for the residual marginal utility of agent i for their j^{th} license, after taking into account that they must pay at least $\underline{p}$. Then from each agent's perspective, playing in $M(C, \underline{p})$ with marginal values $v_i(j)$ gives the same utility as playing in $M(C, 0)$ with marginal values $\hat{v}_i(j)$. The welfare generated by the original auction is $V(x) - Q(x)$, which is equal to $\hat{V}(x) - (Q(x) - \underline{p}x)$. So the welfare generated by M is equivalent to the welfare generated with modified valuations $\hat{V}$, where the marginal social cost of each unit of pollution is reduced by $\underline{p}$. But now note that from the definition of $\underline{p}$, $Q(x) - \underline{p}x \leq 0$ for all $x \leq C$. So in an auction with a cap of C, the welfare is at least the welfare obtained with a social cost of 0; it is always non-negative.

We conclude that the welfare generated by $M(C, \underline{p})$, at any equilibrium, is equal to the welfare generated at an equilibrium of the standard uniform price auction with modified valuations $\hat{V}$. Theorem 3 of [13] (which bounds the welfare in Bayes-Nash equilibria of uniform auctions without costs) therefore applies[5], and the welfare at any equilibrium is at least $\frac{1}{3.15}$ times the optimum achievable under the modified valuations, which is at most $W(M(C))$. ◀

4 Welfare Approximation

Motivated by Lemma 5, we would like to show that the non-strategic welfare of safe-price auctions approximates that of cap-and-price auctions. Unfortunately, there are cases where this does not hold, as demonstrated by Example 6. The example consists of just a single firm that dominates the market and whose demand for licenses has high variance. The firm's value is always large enough that even at a low price and high license cap, the resulting allocation has high net welfare. However, safe-price auctions will generate much lower welfare. Roughly speaking, for any number of licenses C that the auction designer selects as the cap for a safe-price auction, significant welfare will be lost from realizations where the firm's demand is much higher than C, and the corresponding safe price will exclude welfare gains from realizations where the firm's demand is much lower than C. If the variance is high enough, this will cause overall welfare to be low regardless of the choice of C.

▶ **Example 6.** We will present an example for which the expected welfare of any safe-price auction $M(C)$ is at most an $O(1/n)$-approximation to OPT. Let $Q(x) = x^2$. There is a single firm participating in the auction. That firm's valuation curve is drawn according to a distribution F over n different valuation curves, which we'll denote $V^{(1)}, \ldots, V^{(n)}$. For all i, valuation $V^{(i)}$ is defined by $V^{(i)}(x) = \max\{2^{i+1} \cdot x, 2^{2i+1}\}$. The probability that $V^{(i)}(\cdot)$ is drawn from F is proportional to $\frac{1}{2^{2i}}$. That is, the firm has valuation $V^{(i)}$ with probability $\frac{1}{\beta 2^{2i}}$, where $\beta = \sum_{i=1}^{n} 2^{-2i} = \frac{1}{3}(1 - 2^{-2n})$ is the normalization constant. Note that a cap-and-price auction with cap $C = \infty$ and price floor $\underline{p} = 1$ achieves welfare $\sum_{i=1}^{n} 2^{2i} \frac{1}{\beta 2^{2i}} = \frac{1}{\beta} \cdot n$, so OPT is at least this large.[6]

[5] The proof of the 3.15 bound in [13] is actually for a uniform-second-price auction that charges the highest losing bid, or $V(C+1) - V(C)$; however, the proof is also correct for charging a price of the lowest winning bid $V(C) - V(C-1)$.

[6] In fact, this is the "first-best" welfare obtainable at every possible realization, so this is actually the best possible cap-and-price auction and hence the exact value of OPT.

Consider a safe-price auction with cap C'. Suppose that $C' \in (2^k, 2^{k+1}]$ for some $k \geq 1$. Then the safe price is $P(C') = Q(C')/C' = (C')^2/C' = C' > 2^k$. If the firm has valuation $V^{(i)}$ with $i \leq k-1$, then all marginal values are strictly below $P(C')$, then no licenses are allocated and the welfare generated is 0. On the other hand, if the firm has valuation $V^{(i)}$ for any $i \geq k$, then since at most $C' \leq 2^{k+1}$ licenses can be purchased, the auction generates welfare at most $2^{i+1}(2^{k+1}) - (2^{k+1})^2 = 2^{2k+2}(2^{i-k} - 1)$.

The total expected welfare of $M(C')$ is therefore at most

$$\sum_{i=k}^{n} 2^{2k+2}(2^{i-k} - 1) \cdot \frac{1}{\beta 2^{2i}} = \frac{4}{\beta} \sum_{i=k}^{n} 2^{-2(i-k)}(2^{i-k} - 1) \leq \frac{4}{\beta} = O(1).$$

Since $\textsc{opt} \geq \frac{1}{\beta} \cdot n$ but $W(M(C')) = O(1)$ for all C', we conclude that no auction with a safe price achieves an $o(n)$-approximation to $\textsc{opt}$. This concludes the example.

Example 6 is driven by a single firm that dominates the market and has high variance in their demand. We next show that this is the *only* barrier to a good approximation: either a safe-price auction is constant-approximate, or else one can approximate the optimal welfare by selling to just a single firm. To this end, we will be interested in the expected maximum welfare attainable by allocating licenses to just one firm, which we will denote $W^{(1)}$. That is,

$$W^{(1)} = \mathbb{E}_{V \sim F}\left[\max_{i, x \geq 0}\{V_i(x) - Q(x)\}\right].$$

The following theorem is our main result.

▶ **Theorem 7.** *There exists a constant c such that, for any cap C and price floor $\underline{p}$, there exists C' such that $c \cdot W(M(C')) + W^{(1)} \geq W(M(C, \underline{p}))$.*

Note that a corollary of Theorem 7 (combined with Lemmas 3 and 5) is that for any cap-and-price auction $M(C, \underline{p}, \overline{p})$, there exists some C' such that the expected welfare of any Bayes-Nash equilibrium of $M(C')$ is at least a constant factor of $W(M(C, \underline{p}, \overline{p})) - W^{(1)}$. That is, the worst-case equilibrium welfare is a constant-factor approximation to the best welfare achievable by any cap-and-price auction under truth-telling, excluding the welfare contribution of a single firm.

Also, $W^{(1)}$ is the expected welfare of a truthful mechanism that we will call $M^{(1)}$. In $M^{(1)}$, the firms first participate in a second-price auction for the right to buy licenses. The firm with the highest bid wins, and they pay the second-highest bid. The winning firm can then purchase any number of licenses $x \geq 0$, for which they pay $Q(x)$ (in addition to their payment from the initial second-price auction). Since the utility obtained by firm i if they win the initial auction is precisely $\max_x\{V_i(x) - Q(x)\}$, and since a second-price auction is truthful and maximizes welfare, we can conclude that $W(M^{(1)}) = W^{(1)}$. Then another corollary of Theorem 7 is that one can approximate the non-strategic welfare of any cap-and-price auction using either a safe-price auction (in which case the approximation holds at any Bayes-Nash equliibrium), or using the (truthful) mechanism $M^{(1)}$ that allocates licenses to at most one firm.

We are now ready to prove Theorem 7. Fix a cost function Q and distribution F, and choose the optimal cap C and price floor p such that $W(M(C, p))$ is maximized. Let $d(\cdot)$ and $x(\cdot)$ be the demand and allocation functions for $M(C, p)$. For any given realization of preferences V, there is a total demand $d(V) = \sum_i d_i(p)$, where again, $d_i(p) = \arg\max_x V_i(x) - px$. Recall also that $W(x, V) = V(x) - Q(x)$ is the welfare generated by allocation x given aggregate valuation V.

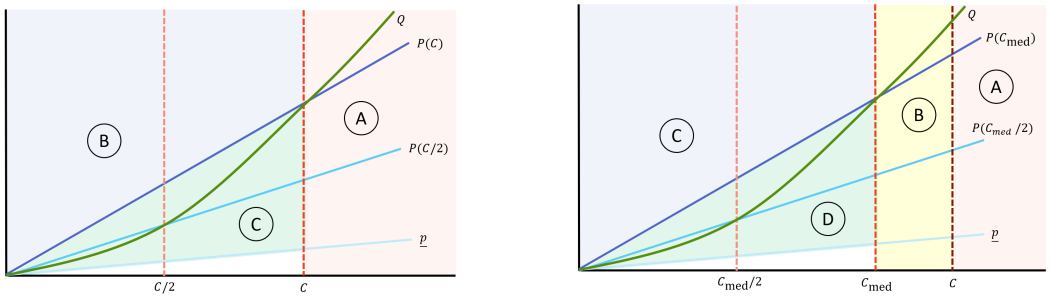


Figure 1 A visualization of the proofs of Theorem 8 (a) and Theorem 7 (b). The x-axis represents quantity of licenses; the y-axis represents value. The expected welfare of benchmark $W(M(C, \underline{p}))$ is divided into separate contributions, based on the location of allocation outcome $(x, V(x))$; these are depicted as shaded regions. Each of these contributions is bounded by either the welfare of a safe-price auction or the welfare obtained from a single buyer. Figure (a) is the (simpler) partition used in Theorem 8, and (b) is the partition used in Theorem 7.

We will begin by proving a simpler version of Theorem 7, which is parameterized by $q := \Pr[d(V) \geq C]$, the probability that the auction sells C licenses. When q is bounded away from 0, we show that there is a safe-price auction that achieves an $O(1/q)$-approximation to $W(M(C, p))$. We note that this result does not require firm valuations to be independent, and holds even if the valuations are drawn from an arbitrarily correlated distribution F. We will explain in Section 4.1 how to extend the argument to the general case of Theorem 7, with details deferred to the appendix.

▶ **Theorem 8.** *For the welfare-optimal cap C and price floor p, there exists a cap C' such that $(1 + 2/q) \cdot W(M(C')) \geq W(M(C, p))$.*

Proof. The welfare generated by $M(C, p)$ can be broken down as follows:

$$W(M(C,p)) = \Pr_{V\sim F}[d(V) \geq C] \cdot \mathbb{E}[W(x(V), V) \mid d(V) \geq C] + \Pr_{V\sim F}[d(V) < C] \cdot \mathbb{E}[W(x(V), V) \mid d(V) < C].$$

The first term is the contribution to the welfare from events where C licenses are sold (or, equivalently, at least C licenses are demanded). The second term is the contribution from events where the total demand for licenses is less than C.

Consider price $P(C)$, the safe price for quantity C. We wish to decompose the second term in the expression above into marginal values above $P(C)$ and those below $P(C)$. For this, we need to introduce some notation. Write θ_i for the largest $j \geq 1$ such that $v_i(j) \geq P(C)$, or $\theta_i = 0$ if $V_i(1) < P(C)$. Write $x_i^{>} = \min\{x_i, \theta_i\}$, and write $x_i^{<} = x_i - x_i^{>}$. Then $x_i^{>}$ is the part of allocation x_i for which firm i has marginal value at least $P(C)$ per unit, and $x_i^{<}$ is the part of x_i for which firm i has a marginal value less than $P(C)$ per unit. We'll also define $V_i^{<}$ as $V_i^{<}(x) = V_i^{<}(x|\theta_i)$; this is the marginal value of i for receiving x additional licenses after already having received θ_i licenses. Note then that $V_i(x_i) = V_i(x_i^{>}) + V_i^{<}(x_i^{<})$, for all i. We then have

$$\begin{aligned}
&\Pr_{V\sim F}[d(V) < C] \cdot \mathbb{E}[W(x(V), V) \mid d(V) < C] \\
&\quad = \int_{V\,:\,d(V)<C} \left(\sum_i V_i(x_i(V)) - Q(x(V)) \right) dFV \\
&\quad \le \int_{V\,:\,d(V)<C} \left(\sum_i V_i(x_i^{>}(V)) - Q(x^{>}(V)) \right) dFV \\
&\qquad + \int_{V\,:\,d(V)<C} \left(\sum_i V_i^{<}(x_i^{<}(V)) - Q(x^{<}(V)) \right) dFV
\end{aligned}$$

where the final inequality used the fact that $Q(x) \ge Q(x^{<}) + Q(x^{>})$, due to the convexity of Q. We conclude that

$$W(M(C,p)) \le \Pr_{V\sim F}[d(V) \ge C] \cdot \mathbb{E}[W(x(V), V) \mid d(V) \ge C] \tag{1A}$$

$$+ \int_{V\,:\,d(V)<C} \left(\sum_i V_i(x_i^{>}(V)) - Q(x^{>}(V)) \right) dFV \tag{1B}$$

$$+ \int_{V\,:\,d(V)<C} \left(\sum_i V_i^{<}(x_i^{<}(V)) - Q(x^{<}(V)) \right) dFV \tag{1C}$$

so that $W(M(C,p)) \le (1\text{A}) + (1\text{B}) + (1\text{C})$. See Figure 1(a) for an illustration.

We claim that the welfare obtained from the first two terms, (1A) and (1B), are covered by the safe-price auction with cap C. The intuition is that a price floor of $P(C)$ does not interfere with the welfare contribution due to licenses with marginal values greater than $P(C)$. One subtlety is that some of the licenses in the summation (1A) might have marginal values less than $P(C)$, but it turns out that it can only improve welfare to exclude such licenses from the allocation.

▷ Claim 9. $W(M(C)) \ge (1\text{A}) + (1\text{B})$.

Proof. Write d' and x' for the demand and allocation under auction $M(C)$, respectively. We have

$$\begin{aligned}
W(M(C)) = &\Pr_{V\sim F}[d(V) \ge C] \cdot \mathbb{E}[W(x'(V), V) \mid d(V) \ge C] \\
&+ \Pr_{V\sim F}[d(V) < C] \cdot \mathbb{E}[W(x'(V), V) \mid d(V) < C].
\end{aligned}$$

Note that we intentionally use $d(V)$, the demand for $M(C,p)$, rather than $d'(V)$ in the expressions above. The second term is precisely (1B), since $x'_i = x_i^{>}$ from the definition of $x^{>}$. We claim that the first term is at least (1A). To see why, fix any V with $d(V) \ge C$ (and corresponding allocation $x = x(V)$ where necessarily $|x| = C$), and note that $x'_i \le x_i$ for all i. Furthermore, $V_i(x'_i) \ge V_i(x_i) - (x_i - x'_i)P(C)$, since x'_i is simply x_i after possibly excluding some items with marginal value less than $P(C)$. Thus, since $|x| = C$,

$$V(x') - |x'| \cdot P(C) \ge V(x) - |x| \cdot P(C) = V(x) - Q(|x|).$$

Also, by convexity, we have $Q(y) \le y \cdot P(C)$ for all $y \in [0, |x|]$. In particular, we have $V(x') - Q(|x'|) \ge V(x') - |x'| \cdot P(C) \ge V(x) - Q(|x|)$, and hence

$$\begin{aligned}
&\Pr_{V\sim F}[d(V) \ge C] \cdot \mathbb{E}[W(x'(V), V) \mid d(V) \ge C] \\
&\ge \Pr_{V\sim F}[d(V) \ge C] \cdot \mathbb{E}[W(x(V), V) \mid d(V) \ge C]
\end{aligned}$$

as claimed. ◁

The more difficult part of the proof of Theorem 8 is to account for term (1C). This represents the contribution of licenses whose marginal values lie below $P(C)$, and are therefore excluded when the price floor is set to $\underline{p} = P(C)$. What we show is that the welfare contribution from all such licenses is approximated by $W(M(C/2))$, the welfare obtained by the safe auction with cap $C/2$.[7] We do this in three steps. First, we show that any optimal cap-and-price auction $M(C, \underline{p})$ must generate non-negative expected welfare conditional on selling C licenses. The intuition is that if the expected welfare from selling C licenses is negative, then it must be strictly preferable to reduce the quantity of licenses.

▶ **Lemma 10.** *If cap C and price floor p are chosen to maximize $W(M(C,p))$, then*

$$\mathbb{E}_{V\sim F}[W(M(C,p)) \mid d(V) \geq C] \geq 0,$$

where $d(V)$ is the total quantity of licenses demanded at price p.

The proof of Lemma 10 appears in Appendix B.1. Next, for any cap-and-price auction M, we provide an upper bound on the total welfare contribution that comes from events where $x < C$ licenses are sold, and the average marginal value of the sold licenses, $V(x)/x$, is less than $P(C)$. In fact, it will be useful to state the lemma more generally: we provide a more general bound that applies to any $\tilde{C} \leq C$ and any valuation that is at most $P(\tilde{C}) \cdot x$ (not necessarily $V(x)$). The lemma bounds the total welfare contribution, $P(\tilde{C}) \cdot x - Q(x)$, by the difference in price between (a) $\tilde{C}$ licenses sold at the safe price for $\tilde{C}$, and (b) the same $\tilde{C}$ licenses sold at the safe price for $\tilde{C}/2$.

▶ **Lemma 11.** *Choose some quantity $\tilde{C}$ and a number of licenses $x \leq \tilde{C}$. Then $P(\tilde{C}) \cdot x - Q(x) \leq \tilde{C} \cdot (P(\tilde{C}) - P(\tilde{C}/2))$.*

The proof of Lemma 11 appears in Appendix B.2. We are now ready to bound the contribution of (1C). The intuition is as follows. By Lemma 11, the contribution from (1C) is at most (a constant times) the gap between the line $y = P(C) \cdot x$ and the curve $Q(x)$ at point $x = C/2$. But recall that $\Pr[d(V) \geq C] = q$, and Lemma 10 implies that, on average, the expected marginal value of licenses allocated subject to this event is at least $P(C)$. Therefore, if we set a license cap of $C/2$, then with probability at least q all $C/2$ licenses will be sold at an average expected marginal value of at least $P(C)$. The welfare generated in this case covers the "gap" at $C/2$, and hence covers the loss due to excluding licenses with marginal values at most $P(C)$.

▷ Claim 12. $W(M(C/2)) \geq \frac{1}{2}q \cdot (1\text{C})$.

Proof. Write d' and x' for the demand and allocation under mechanism $M(C/2)$, respectively. Choose any V such that $d(V) \geq C$. For any such V, $d'(V) \leq d(V)$, and is formed by removing items with marginal value at most $P(C/2)$. In particular, since $x'(V) \leq d'(V)$, we have $V(x'(V)) \geq V(C) - (C - x'(V)) \cdot P(C/2)$. Noting that $Q(|x'(V)|) \leq |x'(V)| \cdot P(C)$ since $x'(V) \leq x(V)$, we have

$$\begin{aligned} V(x'(V)) - Q(|x'(V)|) &\geq V(x'(V)) - |x'(V)| \cdot P(C) \\ &\geq V(C) - (C - |x'(V)|) \cdot P(C/2) - |x'(V)| \cdot P(C). \end{aligned}$$

[7] For convenience we will assume C is even for the remainder of this section. When C is odd, the result holds for at least one of the floor or the ceiling of $C/2$. Details appear in the full version.

Taking an expectation over all V such that $d(V) \geq C$, we note that Lemma 10 implies $\mathbb{E}[V(C)] \geq C \cdot P(C)$. We therefore have

$$\begin{aligned}\mathbb{E}[V(x'(V)) - Q(|x'(V)|)] &\geq C \cdot P(C) - (C - \mathbb{E}[|x'(V)|]) \cdot P(C/2) - \mathbb{E}[|x'(V)|] \cdot P(C) \\ &= (C - \mathbb{E}[|x'(V)|])(P(C) - P(C/2)) \\ &\geq (C/2) \cdot (P(C) - P(C/2))\end{aligned}$$

where in the last inequality we used that $|x'(V)| \leq C/2$ by definition. Since $\Pr[d(V) \geq C] = q$, we therefore have that

$$\begin{aligned}W(M(C/2)) \geq \Pr[d(V) \geq C] \cdot \mathbb{E}[V(x'(V)) - Q(|x'(V)|) \mid d(V) \\ \geq C] \geq q(C/2) \cdot (P(C) - P(C/2).\end{aligned}$$

We now claim that $(C/2) \cdot (P(C) - P(C/2) \geq (1C)/2$, completing the proof. To see why, note that for any $x^* < C$ and any V, $V^<(x^*) - Q(|x^*|) \leq |x^*| \cdot P(C) - Q(|x^*|)$. Thus $(1C) \leq \max_{x^*}\{|x^*| \cdot P(C) - Q(|x^*|)\}$. But note that for any $x^* > C/2$, we have $Q(x^*) > |x^*|P(C/2)$, and hence $|x^*| \cdot P(C) - Q(|x^*|) \leq |x^*| \cdot (P(C) - P(C/2)) \leq (C) \cdot (P(C) - P(C/2))$. Also, for any $x^* < C/2$, $Q(|x^*|)$ lies above the line joining $(C/2, Q(C/2))$ and $(C, Q(C))$, and hence $|x^*| \cdot P(C) - Q(|x^*|)$ is at most the distance between $|x^*| \cdot P(C)$ and this line, which is at most $(C) \cdot (P(C) - P(C/2))$. So in either case we have $\max_{x^*}\{|x^*| \cdot P(C) - Q(|x^*|)\} \leq C \cdot (P(C) - P(C/2))$ as required. ◁

Combining Claim 9 and Claim 12 we have that $W(M(C)) + (2/q)W(M(C/2)) \geq W(M(C, \underline{p}))$, which completes the proof of Theorem 8. ◀

As a corollary, Theorem 8 combined with Lemma 5 and Lemma 3 implies that for any cap and price auction $M(C, \underline{p}, \overline{p})$, there exists a safe-price auction $M(C')$ such that, at any Bayes-Nash equilibrium of $M(C')$, the expected welfare generated is at least $\frac{1}{3.15} \cdot \frac{1}{2} \cdot \frac{1}{1+2/q} \cdot W(M(C, \underline{p}, \overline{p}))$. In particular, if q is a constant bounded away from 0, then $M(C')$ obtains a constant fraction of $W(M(C, \underline{p}, \overline{p}))$ at any Bayes-Nash equilibrium.

4.1 The General Case

We complete the proof of Theorem 7 in Appendix B.3. Here we describe at a high level what steps are needed to complete the argument. We will focus on the case where $\Pr[d(V) \geq C] < 1 - 1/e$, since if $\Pr[d(V) \geq C] \geq 1 - 1/e$ then Theorem 7 follows from Theorem 8.

Recall that in Claim 12, we used the assumption that $\Pr[d(V) \geq C] = q$ to argue that the welfare gained in $M(C/2)$ in the event that $d(V) \geq C$ covers the welfare lost from marginals lying below $P(C)$, up to a constant factor. If the probability that $d(V) \geq C$ is small, then this may no longer be true. To handle this, we consider a reduced cap $C_{\text{med}} < C$ set to be the largest integer such that $\Pr[d(V) \geq C_{\text{med}}] \geq 1 - 1/e$. Our hope is to reproduce the argument from Claim 12, but substituting C_{med} for C. To this end, we divide the welfare of $M(C, \underline{p})$ into *four* parts: all welfare under the event that $d(V) > C$; all welfare from individual agents whose demand is at least C_{med}; the contribution of marginal values greater than $P(C_{\text{med}})$ (but with individual firms demanding at most C_{med}) when $d(V) < C$, and the contribution of marginal values less than $P(C_{\text{med}})$ when $d(V) < C$. See Figure 1(b) for an illustration.

As in the proof of Theorem 8, the contribution due to events where $d(V) > C$ can be covered by the welfare of $M(C, P(C))$.

The contribution from agents who individually demand at least C_{med} licenses is a new case that we didn't have to handle in Theorem 8. It is here that we use $M^{(1)}$, allocating to any single agent. Because the *total* demand is at most than C_{med} with probability at least

$1/e$, independence implies that the expected number of agents who demand more than C_{med} licenses is at most 1, given that the probability that none have demand more C_{med} must be at least $1/e$. So the total contribution to welfare of all such events is at most $W(M^{(1)})$.

If $d(V) \leq C_{\text{med}}$, then the contribution from marginal values that are at least $P(C_{\text{med}})$ can be covered by the welfare of $M(C_{\text{med}}, P(C_{\text{med}}))$, precisely as in Claim 9. We must also handle the case that $d(V) \in [C_{\text{med}}, C]$, and consider the welfare contribution of agents that do not (individually) demand more than C_{med} licenses. Here we use independence: the total quantity demanded by such "small" agents is likely to concentrate, so it is unlikely that the total demand will be larger than $2C_{\text{med}}$. Thus, by imposing a cap of C_{med}, we lose at most a constant factor of the welfare from marginal values greater than $P(C_{\text{med}})$.

The final step is to show that $W(M(C_{\text{med}}/2))$ obtains at least a constant fraction of the welfare generated by marginal values less than $P(C_{\text{med}})$, similarly to Claim 12. When proving Claim 12, we used Lemma 10 to argue about the welfare generated by events where $d(V) > C$. Unfortunately, Lemma 10 does not extend to C_{med}: it could be that the expected welfare generated by $M(C_{\text{med}}, P(C_{\text{med}}))$, conditional on selling C_{med} licenses, is negative. However, we *can* prove an upper bound on how negative this expected welfare can be. After all, if the expected welfare is sufficiently negative sufficiently often, it would be welfare-improving to increase the price floor of $M(C, \underline{p})$ from $\underline{p}$ to $P(C_{\text{med}})$, contradicting the supposed optimality of $M(C, \underline{p})$. This turns out to be enough to prove a bound similar to Claim 12.

Combining these bounds, we can conclude that each of the four parts of the welfare of $M(C, \underline{p})$ can be covered by either a safe-price auction or by $M^{(1)}$, which completes the proof of the theorem.

References

1. California Air Resources Board | Quarterly Auction Information. `https://ww3.arb.ca.gov/cc/capandtrade/auction/auction.htm`. Accessed: 2019-09-06.
2. EU Emissions Trading System (EU ETS). `https://ec.europa.eu/clima/policies/ets_en`. Accessed: 2019-09-06.
3. RGGI, Inc. `https://www.rggi.org/`. Accessed: 2019-09-06.
4. Western Climate Initiative, Inc. `http://www.wci-inc.org/`. Accessed: 2019-09-06.
5. Lawrence M. Ausubel, Peter Cramton, Marek Pycia, Marzena Rostek, and Marek Weretka. Demand Reduction and Inefficiency in Multi-Unit Auctions. *Review of Economic Studies*, 81(4):1366–1400, 2014. URL: `https://EconPapers.repec.org/RePEc:oup:restud:v:81:y:2014:i:4:p:1366-1400`.
6. A. Blum, A. Gupta, Y. Mansour, and A. Sharma. Welfare and Profit Maximization with Production Costs. In *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, pages 77–86, October 2011. `doi:10.1109/FOCS.2011.68`.
7. Chi Kin Chau and Kwang Mong Sim. The price of anarchy for non-atomic congestion games with symmetric cost maps and elastic demands. *Operations Research Letters*, 31(5):327–334, 2003. `doi:10.1016/S0167-6377(03)00030-0`.
8. Richard Cole, Yevgeniy Dodis, and Tim Roughgarden. Bottleneck Links, Variable Demand, and the Tragedy of the Commons. *Netw.*, 60(3):194–203, October 2012. `doi:10.1002/net.21458`.
9. Deborah Cotton. Emissions Trading Design – A Critical Overview, edited by stefan e. weishaar. published by edward elgar, uk, 2014, pp. 249, isbn: 978 1 78195 221 4, aud$114.00 (hardcover). *Australian Journal of Agricultural and Resource Economics*, 59(1):156–158, 2015. `doi:10.1111/1467-8489.12094`.
10. Peter Cramton and Suzi Kerr. Tradeable carbon permit auctions: How and why to auction not grandfather. *Energy Policy*, 30(4):333–345, 2002. URL: `https://EconPapers.repec.org/RePEc:eee:enepol:v:30:y:2002:i:4:p:333-345`.

11 Peter Cramton, David JC MacKay, Axel Ockenfels, and Steven Stoft. *Global Carbon Pricing: The Path to Climate Cooperation*. The MIT Press, June 2017. doi:10.7551/mitpress/10914.001.0001.

12 Peter Cramton and Steven Stoft. Why We Need to Stick with Uniform-Price Auctions in Electricity Markets. *The Electricity Journal*, 20(1):26–37, 2007. URL: https://ideas.repec.org/a/eee/jelect/v20y2007i1p26-37.html.

13 Bart de Keijzer, Evangelos Markakis, Guido Schäfer, and Orestis Telelis. Inefficiency of Standard Multi-unit Auctions. In Hans L. Bodlaender and Giuseppe F. Italiano, editors, *Algorithms – ESA 2013*, pages 385–396, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

14 Arpita Ghosh and Mohammad Mahdian. Externalities in Online Advertising. In *Proceedings of the 17th International Conference on World Wide Web*, WWW '08, pages 161–168, New York, NY, USA, 2008. ACM. doi:10.1145/1367497.1367520.

15 Zhiyi Huang and Anthony Kim. Welfare Maximization with Production Costs: A Primal Dual Approach. In *Proceedings of the Twenty-sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '15, pages 59–72, Philadelphia, PA, USA, 2015. Society for Industrial and Applied Mathematics. URL: http://dl.acm.org/citation.cfm?id=2722129.2722135.

16 Philippe Jehiel, Benny Moldovanu, and Ennio Stacchetti. How (not) to sell nuclear weapons. *The American Economic Review*, pages 814–829, 1996.

17 Thomas Kesselheim, Robert Kleinberg, and Eva Tardos. Smooth Online Mechanisms: A Game-Theoretic Problem in Renewable Energy Markets. In *Proceedings of the Sixteenth ACM Conference on Economics and Computation*, EC '15, pages 203–220, New York, NY, USA, 2015. ACM. doi:10.1145/2764468.2764487.

18 Ilan Kremer and Kjell Nyborg. Underpricing and Market Power in Uniform Price Auctions. *Review of Financial Studies*, 17(3):849–877, 2004. URL: https://EconPapers.repec.org/RePEc:oup:rfinst:v:17:y:2004:i:3:p:849-877.

19 Renato Paes Leme, Vasilis Syrgkanis, and Éva Tardos. Sequential auctions and externalities. In *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms*, pages 869–886. SIAM, 2012.

20 Evangelos Markakis and Orestis Telelis. Uniform Price Auctions: Equilibria and Efficiency. *Theory of Computing Systems*, 57(3):549–575, October 2015. doi:10.1007/s00224-014-9537-9.

21 David McAdams. Adjustable supply in uniform price auctions: Non-commitment as a strategic tool. *Economics Letters*, 95(1):48–53, April 2007. URL: https://ideas.repec.org/a/eee/ecolet/v95y2007i1p48-53.html.

22 Paul Milgrom. *Putting Auction Theory to Work*. Churchill Lectures in Economics. Cambridge University Press, 2004. doi:10.1017/CBO9780511813825.

23 Brian Murray, Richard Newell, and William Pizer. Balancing Cost and Emissions Certainty: An Allowance Reserve for Cap-and-Trade. *Review of Environmental Economics and Policy*, 3(1):84–103, 2009. URL: https://EconPapers.repec.org/RePEc:oup:renvpo:v:3:y:2009:i:1:p:84-103.

24 Stephen J Rassenti, Vernon L Smith, and Bart J Wilson. Discriminatory Price Auctions in Electricity Markets: Low Volatility at the Expense of High Price Levels. *Journal of Regulatory Economics*, 23(2):109–123, March 2003. URL: https://ideas.repec.org/a/kap/regeco/v23y2003i2p109-23.html.

25 Adrian Vetta. Nash Equilibria in Competitive Societies, with Applications to Facility Location, Traffic Routing and Auctions. In *43rd Symposium on Foundations of Computer Science (FOCS 2002), 16-19 November 2002, Vancouver, BC, Canada, Proceedings*, page 416, 2002. doi:10.1109/SFCS.2002.1181966.

26 Robert J. Weber. Making More from Less: Strategic Demand Reduction in the FCC Spectrum Auctions. *Journal of Economics & Management Strategy*, 6(3):529–548, 1997. doi:10.1111/j.1430-9134.1997.00529.x.

27 Martin L Weitzman. Prices vs. quantities. *The review of economic studies*, 41(4):477–491, 1974.

28 Robert Wilson. Auctions of Shares. *The Quarterly Journal of Economics*, 93(4):675–689, 1979. URL: http://www.jstor.org/stable/1884475.

A Omitted Proofs from Section 3

A.1 Proof of Lemma 3

Recall the statement of Lemma 3: for any distribution F and any cap-and-price auction $M(C, \underline{p}, \overline{p})$, there exist a cap C' and price floor $\underline{p}'$ such that $W(M(C', \underline{p}', \infty)) \geq \frac{1}{2}W(M(C, \underline{p}, \overline{p}))$.

Proof. We show that given any mechanism $M = (C_1, \underline{p}, \overline{p})$, we can construct a mechanism M' with price ceiling ∞ such that $W(M') \geq \frac{1}{2}W(M)$.

We can decompose the expected welfare of M into (a) the welfare attained from the first (at most) C_1 licenses sold, and (b) the incremental welfare attained from any licenses sold after the first C_1.

Note that the auction $M^1 = (C_1, \underline{p}, \infty)$, which is M but with price ceiling set to ∞, achieves welfare precisely equal to the former of these two parts. This is because M^1 always allocates at most C_1 licenses, and will allocate them efficiently subject to all marginal values being at least $\underline{p}$.

Next consider the second of these two parts of the welfare of M. If the expected welfare in the second part is negative, then we are already done, so suppose not. Whenever more than C_1 licenses are sold, all licenses are sold at price $\overline{p}$, and therefore have marginal value at least $\overline{p}$. Auction $M^2 = (\infty, \overline{p}, \infty)$, with no license cap and with a price floor of $\overline{p}$, will also sell all such licenses that have marginal value at least $\overline{p}$. We note, however, that M^2 additionally also includes the marginal contribution of the first C_1 licenses. But we claim that this contribution is non-negative: when the event occurs that more than C_1 licenses are sold, the marginal contribution of the first C_1 licenses to the welfare can only be greater than that of those beyond the first C_1. Thus, since the expected welfare in the second part is non-negative, the welfare is only higher if we also include the contribution of the first C_1 licenses whenever more than C_1 licenses are sold. This is precisely the welfare of auction $M^2 = (\infty, \overline{p}, \infty)$, so the welfare of M^2 is therefore at least that of the second of the two parts of the welfare of M.

We conclude that $W(M^1) + W(M^2) \geq W(M)$, and hence either $W(M^1)$ or $W(M^2)$ is at least $\frac{1}{2}W(M)$. ◀

B Omitted Proofs from Section 4

B.1 Proof of Lemma 10

First recall the statement of the lemma. If C and p are chosen to maximize $W(M(C, p))$, then $\mathbb{E}[W(V, x(V)) \mid d(V) \geq C] \geq 0$.

Proof. Suppose not. Then it must be that $\mathbb{E}[W(V, x(V)) \mid d(V) \geq C] < 0$. We will show this implies $W(M(C-1, p)) > W(M(C, p))$, contradicting the optimality of C. To see why, note that when $d(p) < C$, the welfare of the two auctions is identical. Write x and x' for the allocation functions from $M(C, p)$ and $M(C-1, p)$, respectively. When $x \geq C$, we have $x' = C - 1$, and $V(x') \geq \frac{C-1}{x} \cdot V(x)$ by concavity. Similarly, $Q(x') \leq \frac{C-1}{x} \cdot Q(x)$ by convexity. Thus, for any V such that $d(V) \geq C$ and hence $x(V) = C$, we have

$$V(x'(V)) - Q(x'(V)) \geq \frac{C-1}{x(V)}(V(x(V)) - Q(x(V))) = \frac{C-1}{C}(V(x(V)) - Q(x(V))).$$

Taking expectations, we therefore have

$$\begin{aligned}\mathbb{E}[V(x'(V)) - Q(x'(V)) \mid d(V) \geq C] &\geq \frac{C-1}{C}(\mathbb{E}[V(x(V)) - Q(x(V)) \mid d(V) \geq C]) \\ &> \mathbb{E}[V(x(V)) - Q(x(V)) \mid d(V) \geq C]\end{aligned}$$

where the last inequality follows because $\mathbb{E}[V(x(V)) - Q(x(V)) \mid d(V) \geq C] < 0$ by assumption. This implies $W(M(C-1,p)) > W(M(C,p))$, which is the desired contradiction. ◀

B.2 Proof of Lemma 11

First recall the statement of the lemma. Choose some quantity $\tilde{C}$ and a number of licenses $x \leq \tilde{C}$. Then $P(\tilde{C}) \cdot x - Q(x) \leq \tilde{C} \cdot (P(\tilde{C}) - P(\tilde{C}/2))$.

Proof. Suppose $x \geq \tilde{C}/2$. Then $Q(x) \geq \frac{x}{\tilde{C}/2} \cdot Q(\tilde{C}/2)$ by convexity. So $P(\tilde{C}) \cdot x - Q(x) \leq x \cdot \big(P(\tilde{C}) - Q(\tilde{C}/2)/(\tilde{C}/2)\big) \leq \tilde{C} \cdot (P(\tilde{C}) - P(\tilde{C}/2))$ as claimed.

Next suppose $x < \tilde{C}/2$. Then $(x, P(\tilde{C}) \cdot x)$ and $(x, Q(x))$ both lie between the line through the origin with slope $P(\tilde{C})$, and the line between $(\tilde{C}, Q(\tilde{C}))$ and $(\tilde{C}/2, Q(\tilde{C}/2))$. Their difference is therefore at most twice the difference between those two lines at x-coordinate $\tilde{C}/2$, which is $(\tilde{C}/2) \cdot (P(\tilde{C}) - P(\tilde{C}/2))$. ◀

B.3 Omitted Details from the Proof of Theorem 7

Recall the statement of Theorem 7: for any cap C and price floor $\underline{p}$, there exists C' and a constant c such that $c \cdot W(M(C')) + W(M^{(1)}) \geq W(M(C, \underline{p}))$.

As in the proof of Theorem 8, the welfare generated by $M(C,p)$ can be broken down as follows:

$$\begin{aligned}W(M(C,p)) = &\Pr_{V\sim F}[d(V) \geq C] \cdot \mathbb{E}[W(x(V), V) \mid d(V) \geq C] \\ &+ \Pr_{V\sim F}[d(V) < C] \cdot \mathbb{E}[W(x(V), V) \mid d(V) < C]\end{aligned}$$

We will break down the second term into three sub-terms. Recall that we can assume $\Pr[d(V) \geq C] \leq 1 - 1/e$. Choose $C_{\text{med}} < C$ so that $\Pr[d(V) \geq C_{\text{med}}] = 1 - 1/e$. We will write $x_i = x_i^* + x_i^> + x_i^<$. If $x_i \geq C_{\text{med}}$ then we set $x_i^* = x_i$ and $x_i^> = x_i^< = 0$, otherwise we set $x_i^* = 0$. In this latter case, we set $x_i^>$ and $x_i^<$ similarly as in case 1, except that we consider marginal values above and below $P(C_{\text{med}})$ rather than $P(C)$. That is, if θ_i is the largest $j \geq 1$ such that $v_i(j) \geq P(C_{\text{med}})$ (or $\theta_i = 0$ if $v_i(1) < P(C_{\text{med}})$), we have $x_i^> = \min\{x_i, \theta_i\}$ and $x_i^< = x_i - x_i^>$. And as before, we will define $V_i^<$ as $V_i^<(x) = V_i^<(x|\theta_i)$, the valuation of i counting only those marginal values less than $P(C_{\text{med}})$.

Using convexity of Q as in case 1, we then have

$$W(M(C,p)) \leq \Pr_{V\sim F}[d(V) \geq C] \cdot \mathbb{E}[W(x(V), V) \mid d(V) \geq C] \tag{2A}$$

$$+ \int_{V\,:\,d(V)<C} (\sum_i V_i(x_i^*(V)) - Q(x^*(V)))dFV \tag{2B}$$

$$+ \int_{V\,:\,d(V)<C} (\sum_i V_i(x_i^>(V)) - Q(x^>(V)))dFV \tag{2C}$$

$$+ \int_{V\,:\,d(V)<C} (\sum_i V_i^<(x_i^<(V)) - Q(x^<(V)))dFV \tag{2D}$$

so that $W(M(C,p)) \leq (2A) + (2B) + (2C) + (2D)$.

As in Theorem 8, we have that $W(M(C)) \geq (2A)$. For $(2B)$, note that the probability that every bidder i has $x_i < C_{\text{med}}$ is at least $1/e$, from the definition of C_{med}. If we write p_i for the probability that $x_i < C_{\text{med}}$, then we have $\prod_i p_i \geq 1/e$. Subject to this condition, the value of $\sum_i (1-p_i)$ is maximized by setting all p_i equal, in which case we have $\sum_i (1-p_i) \leq n(1-e^{-1/n}) \leq 1$. But this means that the expected number of agents with $x_i \geq C_{\text{med}}$ is at most 1. Thus, $(2B)$ is at most the value of allocating to just a single bidder with $x_i > C_{\text{med}}$, which is at most the maximum possible welfare attainable from allocating to any one bidder. We therefore have that $W(M^{(1)}) \geq (2B)$.

We next claim that $W(M(C_{\text{med}})) \geq (2C)/4$. To see why, note that

$$\begin{aligned}
&\int_{V\,:\,d(V)<C} \left(\sum_i V_i(x_i^{>}(V)) - Q(x^{>}(V))\right) dFV \\
&\leq 2\int_{V\,:\,d(V)<C} 1[x(V) < 2C_{\text{med}}] \cdot \left(\sum_i \left(V_i(x_i^{>}(V)) - x_i^{>} \cdot \tfrac{Q(x^{>}(V))}{x^{>}(V)}\right)\right) dFV \\
&\leq 4\int_{V\,:\,d(V)<C} 1[x(V) < 2C_{\text{med}}] \cdot \left(\sum_i \left(\sum_{j \leq x_i^{>}} v_i(j)/2\right) - x_i^{>} \cdot \tfrac{Q(\min\{C_{\text{med}}, x^{>}(V)\})}{\min\{C_{\text{med}}, x^{>}(V)\}}\right) dFV \\
&\leq 4W(M(C_{\text{med}}))
\end{aligned}$$

where the last inequality follows because, when imposing a cap of C_{med} on an allocation of total size at most $2C_{\text{med}}$, the value obtained is at least the top half of all marginal values, which is more than half of all marginal values.

Finally, we claim that $W(M(C_{\text{med}}/2)) \geq (2D)/21$. Define $\Psi := (C_{\text{med}}/2) \cdot (P(C_{\text{med}}) - P(C_{\text{med}}/2))$. Then Lemma 11, combined with the fact that the total demand is at most C_{med} with probability $1/e$, implies that $(2D) \leq (2/e) \cdot \Psi$.

We claim that for $\beta = \frac{8}{3(e-1)} \approx 1.55$, we must have $\Pr[V(C_{\text{med}}) < Q(C_{\text{med}}) - \beta \cdot \Psi \mid d(V) > C_{\text{med}}] \leq 1/4$. That is, conditional on having total demand at least C_{med}, the probability that the welfare generated at C_{med} is more negative than $-\beta \cdot \Psi$ is at most $1/4$. Suppose not: then consider the difference in welfare between $M(C,p)$ and $M(C, P(C_{\text{med}}))$. The difference is that (2D) would be removed, as would any (negative) contribution with $V(C_{\text{med}}) < Q(C_{\text{med}})$ and $d(V) > C_{\text{med}}$. As we noted above, the total contribution of (2D) to the welfare is at most $2\Psi/e$. But the contribution due to the event $V(C_{\text{med}}) < Q(C_{\text{med}}) - \Psi$ and $d(V) > C_{\text{med}}$ is at most $-(\beta\Psi)(1-1/e)(3/4)$. So as long as $\beta \geq (8/3(e-1)) \approx 1.55$, we would have $W(M(C, P(C_{\text{med}}))) > W(M(C,p))$. This contradicts the optimality of $M(C,p)$.

We can therefore assume $\Pr[V(C_{\text{med}}) \geq Q(C_{\text{med}}) - \beta\Psi \mid d(V) > C_{\text{med}}] \geq 1/4$. By concavity of V and convexity of Q, this means $\Pr[V(C_{\text{med}}/2) \geq Q(C_{\text{med}}/2) + (2-\beta)\Psi/2 \mid d(V) > C_{\text{med}}] \geq 1/4$. Since $d(V) \geq C_{\text{med}}$ with probability at least $1-1/e$, the total welfare generated by $M(C_{\text{med}}/2)$ from the events $d(V) \geq C_{\text{med}}$ is at least $(2-\beta)(1-1/e)\Psi/8$, and hence $W(M(C_{\text{med}}/2)) \geq (2-\beta)(1-1/e)\Psi/8$. Since $(2-\beta)(1-1/e)e/16 \geq 1/21$, the result follows.

We conclude that

$$W(M(C)) + W(M^{(1)}) + 4W(M(C_{\text{med}})) + 21W(M(C_{\text{med}}/2)) \geq W(M(C, \underline{p})).$$

This implies Theorem 7, with $c = 26$.

C Cap-and-Price Auction Outcomes are not Fully Efficient

We note that cap-and-price auctions cannot always implement the fully optimal allocation rule for every distribution F. For an allocation rule to be implementable by some $M(C, \underline{p}, \overline{p})$, it must be that on every input $\mathbf{V}$, either the total allocation is C, or the total allocation is

at most C and each agent wins precisely their marginal bids above $\underline{p}$, or the total allocation is at least C and each agent wins precisely their marginal bids above $\overline{p}$. When there is uncertainty about $\mathbf{V}$, such a restricted allocation rule might return suboptimal allocations on some realizations.

In fact, we note that there are cases where, even under truthful reporting, no cap-and-price auction can achieve a non-vanishing approximation of the unrestricted welfare-optimal allocation.

▶ **Example 13.** We present an example for which the expected welfare of any cap-and-price auction $M(C, \underline{p})$ is at most an $O(1/n)$-approximation to the unrestricted allocation that can optimize individually for each realization of the valuation curves, the welfare of which we call the "first-best welfare" in line with economics terminology.

Let $Q(x) = x^2$. There is a single firm participating in the auction. That firm's valuation curve is drawn according to a distribution F over n different valuation curves, which we'll denote $V^{(1)}, \ldots, V^{(n)}$. For all i, valuation $V^{(i)}$ is defined by $V^{(i)}(x) = 2^{i+1} \cdot x$. These curves are depicted in Figure 2. The probability that $V^{(i)}(\cdot)$ is drawn from F is proportional to $\frac{1}{2^{2i}}$. That is, the firm has valuation $V^{(i)}$ with probability $\frac{1}{\beta 2^{2i}}$, where $\beta = \sum_{i=1}^{n} 2^{-2i} = \frac{1}{3}(1 - 2^{-2n})$ is the normalization constant.

Then in expectation over the realizations of $V(\cdot)$, the *first-best* welfare is

$$\sum_{i=1}^{n} 2^{2i} \frac{1}{\beta 2^{2i}} = \frac{1}{\beta} \cdot n.$$

Consider a cap-and-price auction $M(C, \underline{p})$. For any cap C, the optimal price floor is to set $\underline{p} = 2C$. This eliminates all possible negative welfare contributions at C. As a sanity check, any smaller price floor allows negative contributions, yet any larger price floor excludes positive contributions. For $C' \in (2^k, 2^{k+1}]$, , if the firm has valuation $V^{(i)}$ with $i \leq k$, then all marginal values are strictly below $\underline{p}$, so no licenses are allocated and the welfare generate is 0. On the other hand, if the firm has valuation $V^{(i)}$ for any $i \geq k+1$, then since at most $C < 2^{k+1}$ licenses can be purchased, the auction generates welfare at most $2^{i+1}(2^{k+1}) - (2^{k+1})^2 = 2^{2k+2}(2^{i-k} - 1)$. This yields expected welfare

$$\begin{aligned}
\sum_{i=k+1}^{n} 2^{2k+2}(2^{i-k} - 1) \cdot \frac{1}{\beta 2^{2i}} &= \frac{1}{\beta} \sum_{i=k+1}^{n} 2^{-2(i-k-1)}(2^{i-k} - 1) \\
&= \frac{1}{\beta} \sum_{j=1}^{n-k} 2^{-2(j-1)}(2^{j} - 1) \\
&\leq \frac{1}{\beta} \sum_{j=1}^{n-k} 2^{-(j-1)} \\
&= \frac{1}{\beta}(2 - 2^{-(n-k-1)}) \\
&\leq \frac{2}{\beta} = o(n).
\end{aligned}$$

In comparison to OPT, any cap-and-price policy is off by an order of n, so no $o(1/n)$-approximation to the first-best welfare is possible. This concludes the example.

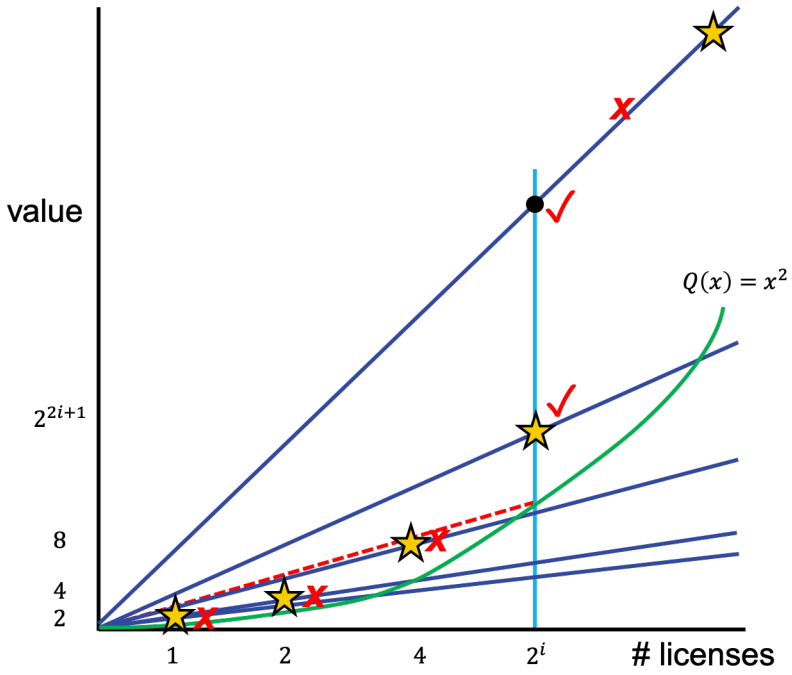


Figure 2 A depiction of Example 13, with the welfare achieve by the vertical cap and dashed price floor denoted by the checks and x's.

Preference-Informed Fairness

Michael P. Kim
Stanford University, CA, USA
mpk@cs.stanford.edu

Aleksandra Korolova
University of Southern California, CA, USA
korolova@usc.edu

Guy N. Rothblum
Weizmann Institute of Science, Rehovot, Israel
rothblum@alum.mit.edu

Gal Yona
Weizmann Institute of Science, Rehovot, Israel
gal.yona@weizmann.ac.il

Abstract

In this work, we study notions of fairness in decision-making systems when individuals have diverse preferences over the possible outcomes of the decisions. Our starting point is the seminal work of Dwork et al. [ITCS 2012] which introduced a notion of *individual fairness* (IF): given a task-specific similarity metric, every pair of individuals who are similarly qualified according to the metric should receive similar outcomes. We show that when individuals have diverse preferences over outcomes, requiring IF may unintentionally lead to less-preferred outcomes for the very individuals that IF aims to protect (e.g. a protected minority group). A natural alternative to IF is the classic notion of fair division, *envy-freeness* (EF): no individual should prefer another individual's outcome over their own. Although EF allows for solutions where all individuals receive a highly-preferred outcome, EF may also be overly-restrictive for the decision-maker. For instance, if many individuals agree on the best outcome, then if any individual receives this outcome, they all must receive it, regardless of each individual's underlying qualifications for the outcome.

We introduce and study a new notion of *preference-informed individual fairness* (PIIF) that is a relaxation of both individual fairness and envy-freeness. At a high-level, PIIF requires that outcomes satisfy IF-style constraints, but allows for deviations provided they are in line with individuals' preferences. We show that PIIF can permit outcomes that are more favorable to individuals than any IF solution, while providing considerably more flexibility to the decision-maker than EF. In addition, we show how to efficiently optimize any convex objective over the outcomes subject to PIIF for a rich class of individual preferences. Finally, we demonstrate the broad applicability of the PIIF framework by extending our definitions and algorithms to the multiple-task targeted advertising setting introduced by Dwork and Ilvento [ITCS 2019].

2012 ACM Subject Classification Theory of computation → Theory and algorithms for application domains

Keywords and phrases algorithmic fairness

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.16

Funding *Michael P. Kim*: Part of this work completed while visiting the Weizmann Institute of Science. Supported, in part, by a Google Faculty Research Award, CISPA Center for Information Security, and the Stanford Data Science Initiative.
Aleksandra Korolova: Part of this work completed while visiting the Weizmann Institute of Science. Another part completed while visiting the Simons Institute for the Theory of Computing.
Guy N. Rothblum: Research supported by the ISRAEL SCIENCE FOUNDATION (grant number 5219/17).
Gal Yona: Research supported by the ISRAEL SCIENCE FOUNDATION (grant number 5219/17).

Acknowledgements This work grew out of conversations during the semester on Societal Concerns in Algorithms and Data Analysis (SCADA) hosted at the Weizmann Institute of Science. The authors thank Omer Reingold for helpful conversations, which influenced our understanding and the presentation of the work.

1 Introduction

Increasingly, algorithms are used to make consequential decisions about individuals. Examples range from determining which content users see online to deciding which applicants are considered in lending and hiring decisions. Automated decision-making comes with benefits, but it also raises substantial societal concerns (cf. [26] for a recent perspective). One prominent concern is that these algorithms might discriminate against individuals or groups in a way that violates laws or social and ethical norms [1, 29, 10, 7]. Thus there is an urgent need for frameworks and tools to mitigate the risks of algorithmic discrimination. A growing literature attempts to tackle these challenges by exploring different fairness criteria and ways to achieve them.

One prominent framework for establishing fairness in algorithmic decision-making systems comes from the seminal work of Dwork et al. [12], which introduced the notion of *individual fairness* (IF). IF relies on a task-specific similarity metric that specifies, for every pair of individuals, how similar they are with respect to the task at hand. Given such a metric, individual fairness requires that similar individuals (according to the metric) be treated similarly, i.e., assigned similar outcome distributions. This is formalized via a Lipschitz condition, requiring that for any two individuals i and j, the distance between their outcome distributions is bounded by their distance according to the metric. Although coming up with a good metric can be challenging, metrics arise naturally in prominent existing examples (e.g. credit or insurance risk scores), and in natural scenarios (e.g. a metric specified by an external regulator). Given an appropriate metric, individual fairness provides powerful protections from discrimination.

Accounting for individuals' preferences

Our work is motivated by settings in which individuals may hold diverse preferences over the possible outcomes. Natural examples of such settings include recommendation systems on professional employment websites where job-searchers have diverse considerations (geography, work-life balance, company culture, etc.) that affect their interest in potential employers, and targeted advertising systems where different users have a wide variety of preferences over the subset of ads they'd like to see out of an enormous set of possibilities. While the metric-based IF constraints prevent myriad forms of discrimination that can arise in automated decision-making systems, we argue that when individuals have different preferences over outcomes, IF can be too restrictive. Specifically, we show that in such settings, ignoring individuals' preferences (as IF does) can come at a high cost to *the very individuals that IF aims to protect*.

We illustrate this observation using a simple example. Consider a university organizing a career expo focused on software developer positions. The university would like to assign each graduating student to (at most) a single interview slot with a prospective employer. To prevent discrimination, the university would like to enforce individual fairness. For simplicity, we assume that there is an unbiased metric for judging qualifications for software development roles across employers based on GPA in the CS major. Consider candidates i, j, and k, who are all similarly qualified, and suppose there are three employers X, Y, and

Z. When the candidates are polled for their preferences, i prefers $X \succ Y \succ Z$, j prefers $Y \succ Z \succ X$, and k prefers $Z \succ X \succ Y$ (possibly due to geographic and work-life balance considerations). Despite the diversity of their preferences, since i, j, and k are all similarly qualified, IF requires that the candidates receive similar distributions over interviews with the employers X, Y, and Z. Thus, IF *rules out the allocation where each candidate gets their most-preferred interview.*

This toy example demonstrates that IF can be overly-restrictive, preventing some solutions where every individual is very happy with their outcome. Moreover, under IF, even the most socially-conscious decision-maker may be forced to disregard the preferences of some groups of individuals in order to satisfy the constraints. For example, if a decision-maker is required by IF to give similar members of majority and minority populations similar outcomes, then the decision-maker may choose the IF solution that gives everyone the outcome preferred by the majority, running the risk of ignoring the preferences of historically-marginalized groups of individuals.

Faced with this shortcoming of IF, we consider alternative notions of fairness that may be better suited to handle settings where individuals hold rich preferences over outcomes. The most natural alternative notion is *envy-freeness* (EF) [32, 14], a classic game-theoretic concept of fair division. A set of outcomes is said to be envy-free if no individual prefers the outcome given to any other individual over their own. At first glance, EF seems like a promising solution concept that addresses the concerns raised about IF: the decision where every individual receives their most-preferred outcome is EF. Indeed, Balcan et al. [4] recently presented EF as an alternative to IF in the context of fair classification.

However, we argue that EF may also be overly-restrictive, constraining the decision-maker in unreasonable ways. Returning to the example of the career expo, suppose another individual ℓ has similar preferences to i, ($X \succ Y \succ Z$), but ℓ has a significantly lower GPA than i. Consider an allocation where i receives their most-preferred interview X, but ℓ does not receive any interview. In this case, ℓ envies i so this solution does not satisfy EF; nevertheless, the solution is reasonable from a fairness perspective. Since i has a much better GPA than ℓ, it doesn't seem unfair to give i the interview with X over ℓ, especially if the interview spots are limited.

This expanded example highlights the need for distinguishing between outcome distributions that might make some (or even all) individuals *unhappy*, from distributions that are *unfairly discriminatory*; articulating this distinction was an important conceptual contribution of the definition of individual fairness [12]. Indeed, the unqualified individual ℓ might be unhappy that they do not receive an interview; further, they might be even less happy when they see that the qualified individual i received an interview with their top choice X. In the eyes of the task-specific similarity metric, however, these two individuals are *different* – according to their GPAs, one is qualified, the other – unqualified. Thus, IF does not consider such an outcome discriminatory. Furthermore, deciding to assign no one to interviews (qualified and unqualified alike) might make no one happy, but it is not unfairly discriminatory, since all individuals are treated similarly.

In this work, we adopt the perspective that given a suitable metric, solutions that are individually fair provide strong protections from discrimination, even though they might not be envy-free. Armed with this perspective, we seek to relax the IF requirements to allow for a richer set of solutions, while still providing meaningful protections against discrimination.

1.1 This Work: Preference-Informed Fairness

Building on the perspective from [12], we propose and study the notion of *preference-informed individual fairness* (PIIF). Our guiding principle is:

Allocations that deviate from individual fairness may be considered fair, provided the deviations are in line with individuals' preferences.

Before describing PIIF, we establish some notation. We model a decision-maker's policy π as a mapping from individuals to allocations, i.e., distributions over outcomes. We assume that each individual i has preferences over the possible allocations, where $p \succeq_i q$ denotes that i (weakly) prefers allocation p to allocation q. To discuss notions of individual fairness, we assume that D is a divergence where $D(p, q)$ measures some distance between two allocations p, q (e.g. the total-variation distance), and d is the task-specific metric where $d(i, j)$ specifies the similarity between individuals i and j.

Using this notation, we can restate the notions of IF and EF as follows. A policy π is *individually-fair* (IF) if for all pairs of individuals i, j, the Lipschitz condition $D(\pi(i), \pi(j)) \leq d(i, j)$ is satisfied.[1] A policy π is *envy-free* (EF) if for all individuals i, for all other individuals j, $\pi(i) \succeq_i \pi(j)$.

Preference-informed individual fairness

As in both IF and EF, PIIF establishes fairness by comparing the allocation of each individual i to the allocation of every other individual j. For each such comparison, PIIF requires that either $\pi(i)$ satisfies individual fairness with respect to $\pi(j)$ or i prefers their allocation $\pi(i)$ over some alternative allocation that would have satisfied individual fairness with respect to $\pi(j)$. More technically, for π to be considered PIIF for each individual i, we require that for every other individual j there exists some alternative allocation $p^{i;j}$ that i could have received that satisfies the IF Lipschitz condition with respect to $\pi(j)$ and where i (weakly) prefers their actual allocation $\pi(i)$ to the IF alternative $p^{i;j}$.

▶ **Definition 1** (PIIF). *A policy π that maps individuals to allocations satisfies Preference-Informed Individual Fairness with respect to a divergence D, a similarity metric d, and individual preferences $\{\succeq_i\}$, if for every individual i, for every other individual j, there exists an alternative allocation $p^{i;j}$ such that:*

- *$p^{i;j}$ is individually fair w.r.t $\pi(j)$:* $D\left(p^{i;j}, \pi(j)\right) \leq d(i, j)$.
- *i (weakly) prefers $\pi(i)$ over $p^{i;j}$:* $\pi(i) \succeq_i p^{i;j}$.

We emphasize that, in general, $p^{i;j} \neq p^{j;i}$; that is, the alternative chosen for i with respect to j's allocation need not be the same as that chosen for j with respect to i. Figure 1 provides a succinct summary of the definitions of IF, EF, and PIIF.

PIIF preserves the spirit of the core interpersonal fairness guarantee of IF: for each individual i, for every individual j who is similar to i, either i's outcome distribution is similar to j's, or i receives an even better (more-preferred) outcome distribution. The main advantage of PIIF over IF is that it allows for a much richer solution space, which can lead to preferable outcomes for individuals. Further, PIIF does not restrict the allocations unnecessarily; as in IF, the constraints only bind when a pairs of individuals are sufficiently similar according to the metric. In other words, PIIF – unlike EF – permits solutions that may be disappointing to some individuals (i.e. where i envies j) but should not be considered discriminatory (because i and j are substantially different according to the task at hand).

[1] Throughout, we assume that d and D are scaled appropriately to be in the same "units." That is, without loss of generality, we assume the relevant Lipschitz constant in the IF-style constraints is 1.

Individual Fairness (IF)
for every i, for every j:
$D(\pi(i), \pi(j)) \leq d(i, j)$

Envy-Freeness (EF)
for every i, for every j:
$\pi(i) \succeq_i \pi(j)$

Preference-Informed Individual Fairness (PIIF)
for every i, for every j, there exists $p^{i;j}$ s.t.
$D(p^{i;j}, \pi(j)) \leq d(i, j)$
$\pi(i) \succeq_i p^{i;j}$

Figure 1 Summary of individual fairness notions.

Referring back to the career expo example, we note that the allocation where the three qualified candidates i, j, and k (deterministically) interview with their preferred employer is PIIF. To see this, consider i comparing their outcome to those of j and k under such an interview assignment. Comparing with j, i prefers outcome X to receiving outcome Y, which would satisfy the IF constraint with respect to j. Similarly, she prefers X to Z, which would satisfy the IF constraint with respect to k. Indeed, since i receives her preferred outcome, one can argue that there is no discrimination against i in the allocation. Similar reasoning applies to j and to k. In fact, the allocation where each individual deterministically receives their preferred outcome is always PIIF, a property we find desirable for a fairness definition. Further, consider the allocation of ℓ, who we assumed was significantly less qualified than i (and thus, j and k). If ℓ is sufficiently dissimilar to all other candidates, then the scheduler can assign ℓ to any interview and still satisfy PIIF. To see this, note that if $d(\ell, i)$ is sufficiently large, we can always take $p^{\ell;i} = \pi(\ell)$, and the constraints for individual ℓ with respect to i will be satisfied (with identical arguments when comparing ℓ to j and k).

1.2 Our Contributions

Our running example illustrates that in many reasonable situations (involving rich and diverse individual preferences over outcomes), the existing notions of individual fairness and envy-freeness may not capture an appropriate notion of fairness or may unnecessarily constrain the decision-maker. In high-stakes domains, such as employment and personalized content selection, both limitations are significant and may hinder adoption of fairness-conscious decision-making. We propose PIIF as a relaxation of IF that addresses the identified shortcomings of existing notions while still providing meaningful protections against discrimination. We view this as an important conceptual contribution in its own right.

With the motivation and definition for PIIF in place, we provide a comprehensive characterization of the relationship between PIIF and other individual notions of fairness. In Section 2, we show formally that PIIF can be viewed as a relaxation of both IF and EF; that is, any solution that satisfies either IF or EF also satisfies PIIF. Further, we demonstrate that PIIF is a non-trivial relaxation of both notions, by proving that there exist settings in which PIIF solutions cannot be captured by IF or EF constraints alone, for *any* choice of metrics d, D and preferences.

To introduce PIIF, we have argued qualitatively that relaxing IF to PIIF allows for more preferable outcomes for individuals. We quantify these claims by comparing the *social welfare* of a decision-maker's policy achievable under PIIF and under IF. In Section 3, we show optimal bounds on the ratio of the best social welfare under PIIF to that under IF; the ratio can grow *linearly* in the number of individuals classified or in the number of possible outcomes grows.

With the definition and properties of PIIF in place, we turn our attention to the algorithmic question of how to achieve PIIF. In Section 4, we show that for a rich family of individual preferences, there is an efficient algorithm to minimize a convex objective subject to PIIF. The result follows by observing that for structured classes of preferences, the set of PIIF constraints is convex. In particular, to optimize over PIIF, we can augment the convex program defined for IF in [12] to capture the additional preference constraints. As such, optimization subject to PIIF is only slightly more complex than optimization subject to IF.

Finally, we demonstrate the versatility of the PIIF framework, by applying preference-informed fairness in the context of targeted advertising (as studied by [13]). Recent empirical findings demonstrate that the ad allocation algorithms run by online advertising platforms may result in discrimination [11, 24, 1] and are thus facing legal scrutiny [31, 29, 6]. As such, developing formal frameworks for understanding fairness in such advertising systems is of great importance. In Section 5, we extend our definition of PIIF and our results to the multiple-task setting defined [13] to model fairness desiderata for the domain of large-scale targeted advertising. We show that in this practically-motivated setting, IF still may restrict the social welfare considerably compared to PIIF *even when the individuals' similarity and preferences are perfectly aligned!* The ratio of the best social welfare under PIIF to that of IF grows *linearly* in the number of tasks.

Organization

Sections 2-5 contain the technical details and proofs of our major contributions with some results deferred to the appendix. We conclude in Section 6 with comparisons to other related works and a discussion of the strengths and limitations of the current approach of preference-informed fairness as well as directions for future investigations.

2 Preference-Informed Individual Fairness

Preliminaries

Given a set of individuals $\mathcal{X}$, we consider policies that assign every individual to an outcome in the set C. We allow randomized allocation rules $\pi : \mathcal{X} \to \Delta(C)$, where for each individual $i \in \mathcal{X}$, their allocation $\pi(i) \in \Delta(C)$ represents a distribution over outcomes $c \in C$. We model individuals' preferences by assuming that every individual $i \in \mathcal{X}$ has a reflexive and transitive binary relation $\succeq_i$ that encodes their preferences over allocations in $\Delta(C)$; for $p, q \in \Delta(C)$, we use $p \succeq_i q$ to denote that i (weakly) prefers p to q.[2] We use $\succeq$ to denote the set of individuals' preference relations, $\succeq = \{\succeq_i\}_{i \in \mathcal{X}}$.

One important structured class of preference relations are those that admit a *utility function*. Here, we assume each individual $i \in \mathcal{X}$ has a real-valued function over allocations $u_i : \Delta(C) \to \mathbb{R}$, where $u_i(\pi(i))$ represents the utility to individual i from the allocation given by π. Given such a utility function, $p \succeq_i q$ if and only if $u_i(p) \geq u_i(q)$.

With this technical notation in place, for completeness, we restate the three definitions of fairness.

▶ **Definition 2** (Individual Fairness). *Given a divergence $D : \Delta(C) \times \Delta(C) \to [0, 1]$ and a similarity metric $d : \mathcal{X} \times \mathcal{X} \to [0, 1]$, a policy $\pi : \mathcal{X} \to \Delta(C)$ is (D, d)-individually fair if for every two individuals $i, j \in \mathcal{X} \times \mathcal{X}$, the following Lipschitz condition holds.*

$$D(\pi(i), \pi(j)) \leq d(i, j) \tag{1}$$

[2] $\succeq_i$ need not be *total* nor *antisymmetric* over $\Delta(C)$.

▶ **Definition 3** (Envy Freeness). *Given a set of preferences $\succeq$, a policy $\pi : \mathcal{X} \to \Delta(C)$ is $\succeq$-envy-free if for all individuals $i \in \mathcal{X}$, and for all other individuals $j \in \mathcal{X}$,*

$$\pi(i) \succeq_i \pi(i) \tag{2}$$

▶ **Definition 4** (Preference-Informed Individual Fairness). *Given a divergence $D : \Delta(C) \times \Delta(C) \to [0,1]$, a similarity metric $d : \mathcal{X} \times \mathcal{X} \to [0,1]$, and a set of preferences $\succeq$, a policy $\pi : \mathcal{X} \to \Delta(C)$ is $(D, d, \preceq)$-PIIF if for all individuals $i \in \mathcal{X}$, for all other individuals $j \in \mathcal{X}$, there exists an allocation $p^{i;j} \in \Delta(C)$ such that:*

$$D\left(p^{i;j}, \pi(j)\right) \leq d(i,j) \tag{3}$$

$$\pi(i) \succeq_i p^{i;j} \tag{4}$$

Often, the divergence D, metric d, and preferences $\succeq$ will be fixed. In these contexts, we use $\Pi^{\text{IF}}, \Pi^{\text{EF}}, \Pi^{\text{PIIF}}$ to denote the set of IF, EF, and PIIF solutions, respectively.

2.1 PIIF relaxes IF and EF

We have argued informally that PIIF captures the appealing aspects of both IF (strong discrimination protections) and EF (respecting the preferences of individuals) without being overly prescriptive in a way that might hurt individuals or the decision-maker. Our first result formalizes these claims, by characterizing PIIF as a relaxation of both IF and EF. We show that any policy that is either IF or EF is also PIIF.

▶ **Proposition 5.** *Fixing a divergence, the metric, and preferences, $\Pi^{\text{IF}} \subseteq \Pi^{\text{PIIF}}$ and $\Pi^{\text{EF}} \subseteq \Pi^{\text{PIIF}}$.*

As solution concepts, both IF and EF are always feasible, but for very different reasons: for IF, any allocation that treats all individuals identically is feasible; for EF, the allocation that gives everyone their most-preferred outcome is envy-free. Thus, both of these extreme solutions will also be feasible for PIIF. In general, PIIF will be a strict relaxation of these concepts that allows for interpolation between the notions. Intuitively, more diverse preferences of individuals tend to give rise to richer sets of PIIF solutions compared to IF, and nontrivial metrics d (i.e., further from the all-zeros "metric") give rise to richer sets of PIIF solutions compared to EF. Given the right framing, the proof of this result is almost immediate.

Proof. To see that an IF policy π satisfies PIIF, for each i, we take $p^{i;j} = \pi(i)$ for all j. Consider an allocation $\pi \in \Pi^{\text{IF}}$. From the perspective of any individual $i \in \mathcal{X}$, when comparing to individual $j \in \mathcal{X}$, if $p^{i;j} = \pi(i)$, then, by the fact that π satisfies IF, condition (3) is satisfied. By reflexivity of $\succeq_i$, (4) is also satisfied, so $\pi \in \Pi^{\text{PIIF}}$.

To see that an EF policy π satisfies PIIF, for each i, we take $p^{i;j} = \pi(j)$ for all j. Consider an allocation $\pi \in \Pi^{\text{EF}}$. From the perspective of any $i \in \mathcal{X}$, when comparing to $j \in \mathcal{X}$, if $p^{i;j} = \pi(j)$, then, condition (3) is satisfied trivially because $D(\pi(j), \pi(j)) = 0$. Since π satisfies EF, we know that $\pi(j) \preceq_i \pi(i)$, so condition (4) also holds; thus $\pi \in \Pi^{\text{PIIF}}$. ◀

PIIF generalizes IF and EF

We remark that this intuition also shows that PIIF is a *generalization* of both IF and EF; that is, both notions can be "implemented" as special cases of PIIF. To implement IF, we can set all individual's preference relation $\succeq_i$ to be the trivial reflexive relation, where for all allocations p, $p \succeq_i p$, and for all nontrivial pairs $p \neq q$, p and q are incomparable. To

implement EF, we simply take $d(i,j) = 0$ for all i, j pairs. In other words, we can think of the set of IF solutions as those where we require the alternative allocation for i compared to j to be i's actual allocation $p^{i;j} = \pi(i)$, and we can think of the set of EF solutions as those where we require the alternative allocation for i compared to j to be j's allocation $p^{i;j} = \pi(j)$.

PIIF is a meaningful relaxation of IF and EF

A natural question to ask is whether we need to introduce a new definition of individual fairness. In particular, we might hope that we could "implement" PIIF using IF with a metric that incorporates preferences or with EF with preferences that incorporate distances. We argue that when there is a rich set of possible outcomes and a correspondingly-rich set of possible preferences, such an approach is infeasible. In particular, PIIF captures constraints that could not be cast within the language of IF or EF alone.

To build intuition, we revisit the career expo example: suppose that two similarly qualified individuals i and j have a similar top choice (say, X), but disagree on their second choice (i prefers Y, whereas j prefers Z). Do these individuals have similar preferences or divergent ones? Intuitively, a fair assignment could give them similar probabilities of seeing X, but different probabilities of seeing Y and Z. Individual fairness treats all outcomes symmetrically for all individuals, and does not let us make such distinctions. The following proposition strengthens this intuition, demonstrating that there are in fact settings in which EF preferences cannot be encoded using any IF metric, and vice versa. Note that this implies that PIIF – a relaxation of both notions – captures constraints that cannot be cast within the language of IF or EF alone.

▶ **Proposition 6.** *There exists a set of preferences $\succeq$ such that for any choice of divergence D and metric d*

$$\Pi^{\succeq\text{-EF}} \neq \Pi^{(D,d)\text{-IF}}.$$

There exists a divergence-metric pair D, d such that for any choice of preferences $\succeq$,

$$\Pi^{(D,d)\text{-IF}} \neq \Pi^{\succeq\text{-EF}}.$$

Proof. In both constructions, we will assume there are two disjoint groups of individuals $S, T \subseteq \mathcal{X}$. Consider two outcomes $p, q \in C$. Suppose $\succeq$ is such that for some $i \in S$ and $j \in T$, $p \succ_i q$ and $q \succ_j p$. Consider any D and d: if $D(p,q) \leq d(j,i)$, then assigning p to j and q to i will be (D,d)-IF, but it is not $\succeq$-EF; otherwise, if $D(p,q) > d(i,j)$, then assigning p to S and q to T will not be (D,d)-IF, even though it is $\succeq$-EF. Thus, no D, d can capture $\succeq$-EF.

Now take D to be total variation distance and consider a metric d where $d(i,j) = 0$ for $i, j \in S \times S$ and $T \times T$, and $d(i,j) = 1$ for $i, j \in S \times T$. Under this metric, assigning any fixed allocation to everyone in S and any (potentially-different) fixed allocation to everyone in T is (D,d)-IF. Consider some $\succeq$. If there is some $i \in S$ such that $p \succ_i q$ or if there is some $j \in T$ such that $q \succ_j p$, then the (D,d)-IF allocation that assigns q to every $i \in S$ and p to every $j \in T$ is not $\succeq$-EF.

Thus, for all individuals in $i \in S \cup T$, $\succeq_i$ must be either the relation, where $p \equiv q$ or the trivial reflexive relation where p and q are incomparable. Suppose $i, j \in S \times S$ both have $\succeq_i = \succeq_j = \equiv$. Then, the solution that assigns p to i and q to j is $\succeq$-EF, but violates the Lipschitz condition of (D,d)-IF. On the other hand, if there is some $i \in S$ that holds the trivial reflexive relation, then the (D,d)-IF solution that assigns p to all of S and q to all of T will not satisfy $\succeq$-EF, because $p \not\succeq_i q$. ◀

2.2 Metric Envy-Freeness

We arrived at PIIF by starting with the metric-based IF as a strong notion of nondiscrimination and relaxing the notion to incorporate individuals' preferences and allow for a richer set of solutions while providing a meaningful protections against discrimination. A conceptually-different approach towards these goals would start with preference-based EF, but allow the decision-maker some freedom by incorporating distances between individuals. In particular, consider the following relaxation of EF, which we call *Metric Envy-Freeness* (MEF), that intuitively captures the idea that no individual should envy the allocation of any other *similar* individual.

▶ **Definition 7.** *Suppose each individual $i \in \mathcal{X}$ has a utility function $u_i : \Delta(C) \to \mathbb{R}$; let $\mathcal{U} = \{u_i\}_{i\in\mathcal{X}}$. Given a similarity metric $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^+$, a policy $\pi : X \to \Delta(C)$ satisfies $(d,\mathcal{U})$-metric-envy-freeness if for every individual $i \in \mathcal{X}$, for every other individual $j \in \mathcal{X}$,*

$$u_i(\pi(i)) \geq u_i(\pi(j)) - d(i,j)$$

This definition starts with the envy-freeness constraint for utility-based preferences, but then relaxes the constraint between i and j by their distance according to the metric. For the metric-utility comparison of MEF to be meaningful, we assume that utilities and metric distances are normalized to one another; without loss of generality, assume that each utility and metric distance is bounded in $[0, 1]$. For each pair i, j, the notion interpolates between two extremes based on the value of $d(i, j)$: if $d(i, j) = 0$, then envy-freeness binds; when $d(i, j) = 1$, the allocation i receives is not constrained by the allocation j receives.

As $d(i, j) \geq 0$ for all pairs of individuals, MEF is clearly a relaxation of EF. That said, it's not immediately obvious how MEF relates to IF or PIIF. While conceptually different, we show that MEF captures a closely-related notion of fairness to PIIF, in the special case where preferences are given by structured utility functions. To relate MEF to PIIF, we need to assume the following Lipschitz conditions.

▶ **Definition 8** (Lipschitz utility)**.** *A utility function $u : \Delta(C) \to \mathbb{R}$ is ℓ-Lipschitz with respect to $D : \Delta(C) \times \Delta(C) \to \mathbb{R}^+$ if $|u(p) - u(q)| \leq \ell \cdot D(p, q)$.*

Lipschitz utility functions are quite natural. For instance, taking D to be the total variation distance, if individuals' preferences admit an expected utility function, where each outcome has utility in $[0, 1]$, then individuals' utilities will be 1-Lipschitz. In other words, individuals' utilities are not highly sensitive to very small changes in the allocation they receive.

▶ **Definition 9** (Reverse-Lipschitz utility)**.** *A utility function $u : \Delta(C) \to \mathbb{R}$ is ℓ-reverse-Lipschitz with respect to $D : \Delta(C) \times \Delta(C) \to \mathbb{R}^+$ if $\frac{1}{\ell} \cdot D(p, q) \leq |u(p) - u(q)|$.*

Reverse-Lipschitz utility functions are less natural. This assumption implies that no pair of outcomes is valued very similarly. One natural setting where the reverse-Lipschitz condition holds nontrivially is in the case of binary outcomes, where each individual prefers one outcome over the other. Under these assumptions, we can show the following relationship between MEF and PIIF.

▶ **Theorem 10.** *Suppose $D : \Delta(C) \times \Delta(C) \to \mathbb{R}^+$ is a divergence, $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^+$ is a similarity metric, and $\mathcal{U} = \{u_i\}$ is a family of utility functions. Let $\succeq^{\mathcal{U}}$ denote the family of preferences induced by $\mathcal{U}$. Consider a policy $\pi : \mathcal{X} \to \Delta(C)$. For some constant $\ell \geq 1$:*

- *Suppose for all $i \in \mathcal{X}$, u_i is ℓ-Lipschitz with respect to D. Then,*

$$\Pi^{(D,d,\succeq^{\mathcal{U}})\text{-PIIF}} \subseteq \Pi^{(\ell\cdot d,\mathcal{U})\text{-MEF}}.$$

- *Suppose for all $i \in \mathcal{X}$, u_i is ℓ-reverse-Lipschitz with respect to D. Then,*

$$\Pi^{(d,\mathcal{U})\text{-MEF}} \subseteq \Pi^{(D,\ell\cdot d,\succeq^{\mathcal{U}})\text{-PIIF}}.$$

Proof. To see that PIIF implies MEF, we start with a policy π that satisfies $(D, d, \succeq^{\mathcal{U}})$-PIIF. To establish MEF, we compare the utility of individual i on their allocation $\pi(i)$ to that of another individual $\pi(j)$; we denote by $p^{i;j}$ the alternative allocation for i that satisfies the PIIF constraints.

$$\begin{aligned} u_i(\pi(i)) &\geq u_i(p^{i;j}) && (5)\\ &\geq u_i(\pi(j)) - \left[u_i(\pi(j)) - u_i(p^{i;j})\right] \\ &\geq u_i(\pi(j)) - \ell \cdot D\left(\pi(j), p^{i;j}\right) && (6)\\ &\geq u_i(\pi(j)) - \ell \cdot d(i,j) && (7)\end{aligned}$$

where (5) follows by the fact that π satisfies PIIF; (6) follows by the Lipschitz condition; and (7) follows again from the fact that π satisfies PIIF. Thus, $u_i(\pi(i)) \geq u_i(\pi(j)) - \ell \cdot d(i,j)$, so π is $(\ell \cdot d, \mathcal{U})$-MEF.

To see that MEF implies PIIF, we start with a policy π that satisfies $(d, \mathcal{U})$-MEF. To establish PIIF, we consider an arbitrary pair of individuals i and j, and exhibit an allocation $p^{i;j}$ that satisfies the PIIF conditions with respect to $\pi(i)$ and $\pi(j)$. Comparing individual i to individual j, we consider two cases.

First, suppose $u_i(\pi(i)) \geq u_i(\pi(j))$ and take $p^{i;j} = \pi(j)$. In this case, the Lipshitz constraint is trivially satisfied,

$$D(p^{i,j}, \pi(j)) = D(\pi(j), \pi(j)) = 0 \leq \ell \cdot d(i,j)$$

and the assumption that $u_i(\pi(i)) \geq u_i(\pi(j))$ implies that

$$\pi(i) \succeq_i \pi(j) = p^{i;j},$$

so the PIIF constraints are satisfied.

Next, suppose $u_i(\pi(i)) < u_i(\pi(j))$ and take $p^{i;j} = \pi(i)$. In this case, the preference condition is trivially satisfied,

$$\pi(i) \succeq_i \pi(i) = p^{i;j}$$

and the Lipschitz condition follows as

$$\begin{aligned} D(p^{i;j}, \pi(j)) &= D(\pi(i), \pi(j)) \\ &\leq \ell \cdot (u_i(\pi(j)) - u_i(\pi(i))) && (8)\\ &\leq \ell \cdot d(i,j) && (9)\end{aligned}$$

where (8) follows by the reverse-Lipschitz condition and (9) follows by the fact that π satisfies MEF. Thus, π is $(D, \ell \cdot d, \succeq^{\mathcal{U}})$-PIIF. ◀

Relating PIIF and MEF

Theorem 10 shows that under appropriate assumptions, we can relate the notions of PIIF and MEF. We interpret the theorem to say that, in most settings we would apply preference-informed fairness, MEF is a strictly more relaxed notion than PIIF; that is, every PIIF solution will be MEF, but there will be MEF solutions that do not satisfy PIIF.

Specifically, as we remarked earlier, it is reasonable to assume that individuals' utility functions are 1-Lipschitz with respect to D. In this case, small changes in the allocation according to D cannot result in dramatically different utilities, and Theorem 10 says that $(D, d, \succeq^{\mathcal{U}})$-PIIF implies $(d,\mathcal{U})$-MEF.

In general, with a rich set of outcomes, we do not expect that individuals' utility functions will be reverse-Lipschitz with respect to D. As such, there are cases when $(d,\mathcal{U})$-MEF will be considerably more relaxed than $(D, d, \succeq^{\mathcal{U}})$-PIIF, allowing for deviations from (D, d)-IF that do not give utility improvements for all individuals. To see this point, consider the following example with two individuals i, j, where $d(i, j) = 0.5$ and two outcomes p, q, with the utility functions of i and j defined as follows.

	p	q
u_i	1.0	0.5
u_j	0.5	1.0

We take D to be the total variation distance between allocations. It's easy to verify that all allocations that treat the individuals identically and the welfare-maximizing solution where i receives p and j receives q will be MEF; this can be seen by noting that each of these solutions satisfies EF, thus, also MEF. In fact, in this example, the utility functions are sufficiently Lipschitz to guarantee that all PIIF solutions will be MEF.

On the other hand, consider the allocation where i receives q and j receives p, deterministically. This solution is not IF, because $1 = D(q, p) > d(i, j) = 0.5$, nor EF, because both individuals envy the others' solution. Further, the solution is not PIIF. To see this, note that because each individual receives their least favorite outcome, the only surrogate $p^{i;j}$ that can be chosen for individual i such that $q \succeq_i p^{i;j}$ is $p^{i;j} = q$ (similarly, $p^{j;i} = p$ for individual j). As the actual allocation is not IF, one of the PIIF constraints will be violated.

Still, this allocation does satisfy MEF. Specifically, $0.5 = u_i(q) \geq u_i(p) - d(i, j) = 1.0 - 0.5$ and $0.5 = u_j(p) \geq u_j(q) - d(j, i) = 1.0 - 0.5$. In other words, in this instance, MEF allows for a solution that is not permitted by any of the other notions of individual fairness we consider. In particular, the allocation deviates from individual fairness in a way that does not help either individual. This example suggests that in reasonable settings, MEF is a strictly weaker concept than PIIF and may be too permissive.

3 Welfare under IF, EF, and PIIF

In order to motivate PIIF, we argued that IF may be overly-restrictive and limit the "quality" of solutions from the perspective of individuals. In this section, we formalize this claim, showing that PIIF admits solutions that can be significantly more preferable to individuals. We quantify the idea of individual quality using the game-theoretic notion of social welfare.

We restrict our attention to the common setting where individuals hold preference relations that admit a utility function. In this setting, given a policy π, we can track the *social welfare* of the policy, defined to be the overall utility experienced by individuals.

▶ **Definition 11.** *Given a policy $\pi : \mathcal{X} \to \Delta(C)$, social welfare $\mathcal{W}(\pi)$ is the sum of the individuals' utilities under π.*

$$\mathcal{W}(\pi) = \sum_{i \in \mathcal{X}} u_i(\pi(i)) \tag{10}$$

For a collection of allocations Π, we let $\mathcal{W}^*(\Pi) = \max_{\pi \in \Pi} \mathcal{W}(\pi)$ denote the optimal social welfare achievable by any allocation in Π, and let $\mathcal{W}^* = \mathcal{W}^*(\Delta(C)^{\mathcal{X}})$ denote the optimal (unconstrained) social welfare.

An important special case of preferences that admit a utility function are those that admit an *expected utility function*. Using such preferences is a standard approach in economics for modeling decision-making in the presence of uncertainty.[3]

▶ **Definition 12** (Expected utility representation)**.** *A preference relation $\succeq$ admits an expected utility representation if and only if there exists a function $u : C \to \mathbb{R}^+$, such that for any two allocations $p, q \in \Delta(C)$,*

$$p \succeq q \iff \sum_{c \in C} p_c \cdot u(c) \geq \sum_{c \in C} q_c \cdot u(c) \tag{11}$$

Individual fairness may restrict social welfare

Using the notation introduced above, note that $\mathcal{W}^*(\Pi^{\text{PIIF}}) = \mathcal{W}^*$ because the welfare-maximizing allocation is feasible for PIIF. Here, we aim to understand how much IF may restrict the best social welfare compared to PIIF, by relating $\mathcal{W}^*(\Pi^{\text{IF}})$ to $\mathcal{W}^*$.

Intuitively, social welfare can be hurt significantly by requiring IF compared to PIIF when many individuals are considered similar, but there is a diversity of preferences over outcomes. Formalizing this intuition, we argue that without any assumptions about the class of utility functions of individuals, the ratio between the best social welfare under IF and PIIF can grow with the number of individuals. Further, even under the stronger assumption that individuals' preferences admit an expected utility representation, the ratio can grow with the number of outcomes.

▶ **Theorem 13.** *There exists a family of instances such that*

$$\frac{\mathcal{W}^*}{\mathcal{W}^*\left(\Pi^{\text{IF}}\right)} \geq |\mathcal{X}|\,.$$

Additionally, there exists a family of instances where individuals' preferences admit an expected utility representation and

$$\frac{\mathcal{W}^*}{\mathcal{W}^*\left(\Pi^{\text{IF}}\right)} \geq |C|\,.$$

Proof. The two claims of the theorem follow by similar constructions. Suppose every individual is considered similar according to d; that is, for all $i, j \in \mathcal{X} \times \mathcal{X}$, $d(i, j) = 0$. This means that any IF solution must assign every individual the same distribution over outcomes. To show the gaps, we will compare the best IF solution (i.e. constant allocation) to the welfare-maximizing solution, which is feasible under PIIF.

[3] Although not all preference relations admit this form, a rich class of preferences do. For example, different levels of tolerance towards risk can be captured within this framework (e.g., a risk-averse individual would have a utility function u which is concave). Von Neumann and Morgensterm [33] provide a complete characterization of this class of preference relations.

To begin, suppose we allow individuals to specify arbitrary utility functions; let each individual $i \in \mathcal{X}$ hold a distinct $p_i \in \Delta(C)$ (i.e., $p_i \neq p_j$ for all $i \neq j$) such that $u_i(p_i) = 1$ and $u_i(q) = 0$ for any $q \neq p_i$. In this case, the optimal social welfare is $\mathcal{W}^* = |\mathcal{X}|$. For any fixed allocation, however, the best social welfare is to choose a distribution over the set of $\{p_i\}$. Any such distribution will achieve welfare 1; thus, $\mathcal{W}^*(\Pi^{\text{IF}}) = 1$.

Now, suppose every individual is required to specify an expected utility representation. Let each individual choose some $c \in C$, such that a $1/|C|$-fraction of individuals prefer each outcome c; let $u_i(c) = 1$ for their preferred outcome and $u_i(c) = 0$, otherwise. Again, the optimal social welfare is $\mathcal{W}^* = |\mathcal{X}|$. Under any policy that assigns every individual the same fixed allocation p, the social welfare is given by $\sum_{i\in\mathcal{X}} \sum_{c\in C} p_c \cdot u_i(c) = \frac{|\mathcal{X}|}{|C|} \cdot \sum_{c\in C} p_c = \frac{|\mathcal{X}|}{|C|}$. Thus, $\mathcal{W}^*(\Pi^{\text{IF}}) = \frac{|\mathcal{X}|}{|C|}$. ◀

We note that the gaps demonstrated in Theorem 13 are optimal in their settings. In particular, any constant allocation will be IF, and we can always recoup a $1/|\mathcal{X}|$ fraction of the social welfare with a constant allocation tailored for the individual with the highest utility. Further, in the case where preferences admit an expected utility representation, we can choose the constant allocation on the $c \in C$ of maximum welfare. Finally, we note that one unsatisfying aspect of these constructions is that they rely on the fact that all individuals are similar according to d. In Section 5.1, we show that in the multiple-task setting of [13], such gaps exist even with nontrivial metrics that seem to be aligned with social welfare.

PIIF does not guarantee social welfare

Because PIIF is a relaxation of IF, the best social welfare achievable under PIIF is always at least that of IF. That said, because PIIF is a strict relaxation of IF, it does not necessarily guarantee that *every* allocation's social welfare improves under PIIF. In particular, when the decision-maker seeks to optimize a utility function that runs against individuals' utilities within the set of PIIF solution, the obtained social welfare may be arbitrarily worse under the PIIF constraints than IF constraints.

Suppose that the decision-maker has an additive utility function of the form $f(\pi) = \sum_{i\in\mathcal{X}} f_i(\pi(i))$, for $f_i : \Delta(C) \to \mathbb{R}$. Let $\pi_f^{\text{IF}} = \text{argmax}_{\pi\in\Pi^{\text{IF}}} f(\pi)$ and $\pi_f^{\text{PIIF}} = \text{argmax}_{\pi\in\Pi^{\text{PIIF}}} f(\pi)$ denote the optimal IF (resp., PIIF) solution in terms of $f(\cdot)$.

▶ **Proposition 14.** *There exists a family of instances and an additive utility function f such that*

$$\mathcal{W}(\pi_f^{\text{PIIF}}) = 0 < \mathcal{W}(\pi_f^{\text{IF}}).$$

Proof. Suppose there are two disjoint classes of individuals, S and T, which each make up half of $\mathcal{X}$, and all individuals are similar; for all $i, j \in \mathcal{X} \times \mathcal{X}$, $d(i, j) = 0$. Suppose there are three outcomes $C = \{p, q, r\}$. Consider the utility functions defined as follows.

	p	q	r
$f_{i\in S}$	$1/2 + \varepsilon$	1	0
$f_{j\in T}$	$1/2 + \varepsilon$	0	1
$u_{i\in S}$	1	0	0
$u_{j\in T}$	1	0	0

Under IF, the decision-maker must treat all individuals identically. Given this constraint, the outcome that maximizes f is allocating p deterministically to everyone. This allocation π_f^{IF} achieves $\mathcal{W}(\pi_f^{\text{IF}}) = 1$. Without the constraint that all individuals' allocations are identical,

the allocation that assigns q to individuals from S and r to individuals from T maximizes f. In fact, this allocation will be feasible for PIIF: note that every individual experiences 0 utility from everyone's allocation, so no one envies anyone else; under PIIF, envy-free solutions are feasible. Thus, $\mathcal{W}(\pi_f^{\text{PIIF}}) = 0$. In fact, this reveals that the proposition holds not only for PIIF, but also for any notion that relaxes EF. ◀

This construction demonstrates that the PIIF constraints alone do not guarantee improved social welfare compared to the IF constraints. We remark, however, that this is in line with our initial motivation for PIIF: decoupling the objective of provably preventing discrimination from the objective of ensuring beneficial outcomes in aggregate. Finally, we note that if this is a concern, it can be addressed within our framework by adding a constraint to the optimization program, discussed in detail in Section 4, that ensures the social welfare is above some baseline. In particular, an appropriate individually fair solution could act as this baseline, by first computing the social welfare obtained by it and then requiring that the resulting PIIF solution has at least this social welfare. At the extreme, we could even add such a constraint on the utility experienced by each individual; thus, obtaining the guarantee that any deviations from an IF solution are optimal, from the individuals' perspective, compared to some benchmark IF solution.

4 Optimization subject to PIIF

As we have argued, satisfying PIIF is always feasible: on the one hand, we can take any IF solution, including a trivial policy that treats all individuals identically; alternatively, we can take any EF solution, including the welfare-maximizing policy that gives everyone their most-preferred allocation. In this section, we study the question of efficient optimization of a decision-maker's utility function subject to PIIF constraints. As is standard in much of learning and optimization, we frame this task as the following minimization problem:

$$\begin{aligned} &\underset{\pi:\mathcal{X}\to\Delta(C)}{\text{minimize}} && f(\pi) \\ &\text{subject to} && \pi \in \Pi^{\text{PIIF}} \end{aligned}$$

In this section, we answer the question of feasibility of efficient optimization in the positive when f is convex, and the preferences arise from a structured, but rich class of relations.

4.1 Structured preferences

In principle, PIIF can be instantiated with any notion of preference. Without assuming anything about the preferences, however, the PIIF constraints could be difficult to handle: the space of allocations, over which the PIIF constraints are defined, is exponential. In realistic settings, where the number of individuals or outcomes is large, this exponential dependence may be intractable. Towards efficient optimization, we focus on two rich and structured preference classes.

First, we include the prominent class of preferences that admit an expected utility representation, as defined in Definition 12. Additionally, we include the class of *stochastic domination* preferences. Stochastic domination formalizes the intuition that for any distribution over outcomes, a shift of probability mass from less desirable outcomes to more desirable outcomes is considered preferable. Viewing an allocation $p \in \Delta(C)$ as a discrete probability distribution, we denote by $c \sim p$ an outcome randomly sampled from p.

▶ **Definition 15** (Stochastic domination). *For an individual with a utility function $u : C \to [0, M]$ and for any two allocations $p, q \in \Delta(C)$, p stochastically dominates q if*

$$p \succeq q \iff \forall x \in [0, M], \quad \mathop{\mathbf{Pr}}_{c\sim p}[u(c) \geq x] \geq \mathop{\mathbf{Pr}}_{c\sim q}[u(c) \geq x]$$

That is, an allocation p is (weakly) preferred over q if for every possible level of utility x, the probability of achieving at least x is no worse under p than it is under q. Note that stochastic domination represents an interesting example of a non-total preferences, as two allocations may be incomparable.[4]

4.2 Efficient optimization subject to PIIF

Here, we prove that when individuals' preferences are of the forms defined above, the PIIF constraints admit efficient optimization. Formally, the following theorem demonstrates that when the divergence over allocations D is taken to be total variation distance D_{tv}, and assuming oracle access to the individual-fairness metric d, we can write the PIIF constraints as a set of (polynomially-many) linear inequalities; thus, we can efficiently minimize any convex objective f.

▶ **Theorem 16.** *Let $\succeq = \{\succeq_i\}_{i\in\mathcal{X}}$ be the set of individuals' preferences. If every $\succeq_i$ is either the stochastic domination relation or admits an expected utility representation, then the set of $(D_{\text{tv}}, d, \succeq)$-PIIF allocations forms a convex polytope in $\mathbb{R}^k$, where $k = \text{poly}(|\mathcal{X}|, |C|)$.*

Proof. We specify the PIIF constraints using the following variables: for all $i \in \mathcal{X}$, let $\pi(i) \in \Delta(C)$ be a vector denoting the actual allocation; for every pair of individuals $(i, j) \in \mathcal{X} \times \mathcal{X}$, let $p^{i;j} \in \Delta(C)$ be a vector denoting the alternative allocation for i when comparing to j. We argue that the PIIF constraints given in (3) and (4) can each be written as linear inequalities over these variables.

First, since D is taken to be the total variation distance we can translate (3) as $\frac{1}{2} \cdot \sum_{c\in C} \left|p^{i;j}_c - \pi(j)_c\right| \leq d(i, j)$. This can be written as $2 \cdot |C| + 1$ linear inequalities (with the introduction of $|C|$ additional variables representing the absolute values).

Next, we turn to the constraint given in (4). First, consider the case of the preference relations admitting an expected utility form. Let u_i be the utility function for individual i. By definition,

$$\pi(i) \succeq_i p^{i;j} \iff \sum_{c\in C} \pi(i)_c \cdot u_i(c) \geq \sum_{c\in C} p^{i;j}_c \cdot u_i(c).$$

Thus, for every $i \in \mathcal{X}$, the PIIF constraint given in (4) with respect to $j \in \mathcal{X}$ can be written as a linear inequality in the variables $p^{i;j}$ and $\pi(i)$.

Next, we consider the case of the stochastic domination preference relation. We introduce some notation as follows. Fix an individual i and their allocation, $\pi(i)$. Suppose $|C| = k$, and that the outcomes in C are labeled in decreasing order according to i's preferences: $u_i(c_0) \geq u_i(c_1) \geq \cdots \geq u_i(c_{k-1})$. With this ordering in place, we have that for any allocation $p \in \Delta(C)$ and every rank $r \in [k]$, $\mathbf{Pr}_{c\sim p}[u_i(c) \geq u_i(c_r)] = \sum_{t=1}^{r} p_t$. Thus, for each $i \in \mathcal{X}$ we can write the stochastic domination condition as k linear inequalities for each $j \in \mathcal{X}$, where

$$\pi(i) \succeq_i p^{i;j} \iff \forall r \in [k] : \sum_{t\in[r]} \pi(i)_t \geq \sum_{t\in[r]} p^{i;j}_t.$$

Importantly, this demonstrates that for this preference relation, the constraint given in (4) can be enforced using an additional $O(|C|)$ linear constraints, one for every $r \in [k]$. ◀

[4] We remark that this preference notion is a special case of the statistical concept of *first-order* stochastic domination [16, 5].

Other notions of preference

Theorem 16 focuses on the case in which individuals' preferences satisfy one of the two forms discussed above and formalized in Definitions 12 and 15. Naturally, however, not all preference relations satisfy one of these two forms. Appealing examples include preferences where the individual deems some of the outcomes to be substitutes (i.e., interested in exactly one) or complements (i.e., only interested in the complete set) or possibly preferences that value diversity of outcomes. We leave the question of whether PIIF admits efficient optimization over such non-convex preferences as an interesting direction for future research.

5 Fairness in Targeted Advertising: Multiple-Task PIIF

In this section, we extend the definition and study of preference-informed individual fairness to the *multiple-task* setting, formalized and studied by Dwork and Ilvento [13]. This setting was introduced as a model in which to study fairness in targeted advertising, a form of online advertising where ad platforms allow advertisers to specify the characteristics of users they would like to reach, and then make algorithmic decisions as to which users will see which ads based on the advertiser specifications, predictions of ad relevance to individuals, and the ad platform's revenue objectives. Targeted advertising has become pervasive and increasingly moderates individuals' exposure to opportunities. In recent years, numerous concerns have been raised about its fairness and discrimination implications, ranging from concerns about discriminatory advertiser targeting practices enabled by the platforms [3, 30, 2] to concerns about the ad delivery and allocation algorithms run by the platforms introducing bias where none was intended by the advertiser [11, 24, 1, 31, 29]. As part of a lawsuit settlement, the most prominent targeted advertising platform, Facebook, has begun to take steps to ensure advertisers cannot discriminate in their targeting practices [28]. However, the question of how to ensure that the ad delivery and allocation algorithms do not lead to discrimination is wide open [1, 25], in part due to lack of agreement over fairness definition(s) and ad platforms' concerns that existing definitions will restrict allocations in ways that significantly impact their revenue. As such, the multiple-task setting in the presence of individual preferences provides an important model to investigate formal guarantees of non-discrimination without being overly-restrictive for the decision-maker.

In the multiple task setting, we think of the set of outcomes C as arising from a collection of distinct tasks, e.g. deciding whether to show an ad for a user of each ad campaign $c \in C$. Importantly, in this setting, a separate fairness metric d_c is specified for each task (ad campaign), which naturally models real-world concerns in advertising, where different types of ads (e.g. housing, employment, product) are subject to different regulations and standards of fairness.

▶ **Definition 17** (Multiple-task IF). *An allocation $\pi : \mathcal{X} \to \Delta(C)$ is said to be $(D, \{d_1, \dots d_k\})$-*individually fair *in the multiple-task setting if for every two individuals $i, j \in \mathcal{X} \times \mathcal{X}$, the task-specific Lipschitz condition holds for each task:*

$$\forall c \in C : \quad D(\pi(i)_c, \pi(j)_c) \leq d_c(i, j).$$

In this setting, and particularly its application to ad delivery in the targeted advertising context, the benefits of a preference-informed approach to ensuring fairness become particularly salient. For instance, consider the following example, due to [13]. Suppose there are two ad campaigns, one for a high-paying tech job and another for childrens' toys. The ad-specific metrics capture the fact that differentiating based on a particular criteria could be

permissible in some cases and not in others. For example, the metric associated with the tech ad should assign a small distance to individuals of similar qualifications regardless of their status as a parent, whereas the metric for toys might reasonably assign significant distance between parents and non-parents. However, under Multiple-task IF, a parent that is qualified for the tech job ad but is interested in toys must see the tech ad with the same probability as a qualified non-parent – an overly restrictive requirement. PIIF in the multiple-task setting addresses precisely this issue.

▶ **Definition 18** (Multiple-task PIIF). *An allocation $\pi : \mathcal{X} \to \Delta(C)$ satisfies $(D, \{d_1, \dots d_k\}, \preceq)$-preference-informed individual fairness in the multiple-task setting if for all individuals $i \in \mathcal{X}$, for all other individuals $j \in \mathcal{X}$, there exists an allocation $p^{i;j} \in \Delta(C)$ such that:*

$$\forall c \in C: \quad D\left(p_c^{i;j}, \pi(j)_c\right) \leq d_c(i,j)$$
$$\pi(i) \succeq_i p^{i;j}$$

Again, in the multiple-task setting, the preference-informed extension of IF will require that for every individual $i \in \mathcal{X}$, when comparing to every other individual $j \in \mathcal{X}$, the individual i prefers their actual allocation to some alternative allocation, $p^{i;j}$. The main distinction is that now $p^{i;j}$ has to satisfy multiple-task IF with respect to j's current allocation.

Efficient optimization. Our results regarding efficient optimization subject to PIIF from the single-task setting (Section 4) directly extend to the multiple-task setting. In particular, given the ad-specific metrics, individuals' utilities and the advertisers' bids, the platform can efficiently compute the revenue- (or social welfare-) maximizing PIIF allocation.

An interesting direction for future work is relaxing the full information assumption. In particular, an online model, in which allocations are determined on a per-user basis, could naturally be more applicable, as well as allow for the preferences to be "discovered" through the allocation procedure (see [15] for a similar approach wrt the metric itself). This may necessitate investigation of non-trivial tradeoffs, as learning individuals' preferences requires some exploration, which may be at odds with ensuring fair treatment.

5.1 Fairness and social welfare in the multiple-task setting

The construction of Proposition 13 demonstrates that in the single-task setting, the gap between the best social welfare obtainable under IF and PIIF can be large even under very structured classes of preferences. This construction can be generalized to the multiple-task setting; however, an unconvincing aspect of it is the requirement that every individual is identical according to the metric. In such a setting, it's not surprising that IF is overly-constrained.

Here, we describe a family of instances in the multiple-task setting where the per-task similarity is *perfectly aligned with individuals' utilities*; that is, if two individuals benefit similarly from an outcome c, then they are similar. In such instances, we'd expect that the metric constraints would be perfectly aligned with social welfare. Still, we show that this intuition does not carry through for multiple-task IF: for a set of tasks, there are instances where the optimal social welfare under PIIF approaches a factor $|C|$ larger than the best IF solution.

▶ **Theorem 19.** *For any constant $\varepsilon > 0$, there is a sufficiently large $|\mathcal{X}|$ such that there exists a distribution of multiple-task instances where for each task $c \in C$, $d_c(i,j) = |u_i(c) - u_j(c)|$ and*

$$\frac{\mathcal{W}^*}{\mathcal{W}^*\left(\Pi^{\text{IF}}\right)} \geq |C| - \varepsilon.$$

Intuition. Our proof is inspired by a construction of [13], which shows the impossibility of multiple-task IF under "naive composition." We begin by adapting their construction to our setting. Suppose there are two subpopulations of individuals $S \subseteq \mathcal{X}$ and $T = \mathcal{X} \setminus S$. We assume that each task-specific similarity metric d_c is determined by individuals' utility: $d_c(i,j) = |u_i(c) - u_j(c)|$. Additionally, suppose there are two ad campaigns c_0 and c_S. c_0 is a generic campaign where for all individuals $i \in \mathcal{X}$, $u_i(c_0) = 1$; thus, $d_{c_0}(i,j) = 0$ for all $i,j \in \mathcal{X} \times \mathcal{X}$. c_S is targeted where subpopulation S receives nontrivial utility, but the rest of the population receives no utility; thus, d_{c_S} treats pairs within $S \times S$ similarly, pairs from $T \times T$ similarly, but for $i,j \in S \times T$, is arbitrarily large, say $d_{c_S}(i,j) = 1$.

Given these campaigns, a natural allocation of ads to individuals, which we call $\pi^{\mathcal{W}}$, deterministically assigns $\pi^{\mathcal{W}}(i) = c_S$ to all individuals in $i \in S$ since they receive positive utility from c_S. Further, it assigns the untargeted campaign $\pi^{\mathcal{W}}(j) = c_0$ to individuals in $j \in T$ because they benefit positively from seeing c_0, whereas they get no benefit from c_S. Indeed, $\pi^{\mathcal{W}}$ maximizes the social welfare; everyone sees their favorite ad. But $\pi^{\mathcal{W}}$ violates multiple-task IF on c_0; that is, for $i,j \in S \times T$, $\left|\pi^{\mathcal{W}}(j)_{c_0} - \pi^{\mathcal{W}}(i)_{c_0}\right| = 1$ but $d_{c_0}(i,j) = 0$. Intuitively, under multiple-task IF, because everyone in $\mathcal{X}$ is similar according to c_0, the platform must decide whether it is more beneficial to show c_S to the individuals in S at the expense of not being able to show c_0 to the individuals outside of T. The proposition follows by extending this construction beyond the case of two campaigns and two subgroups, and carefully constructing utility functions for individuals.

Proof. Suppose $|C| = n$. Let $t \in \mathbb{N}$ be some constant. Suppose the universe of individuals $\mathcal{X} = S_0 \cup S_1 \cup \ldots S_{n-1}$ is partitioned into disjoint subpopulations ($S_\ell \cap S_m = \varnothing$ for $\ell \neq m$) for $|\mathcal{X}| \geq t^n$. The subpopulations will become progressively smaller as ℓ increases; for each $\ell > 0$, $\frac{|S_\ell|}{|\mathcal{X}|} = 1/t^\ell$ and let $\frac{|S_0|}{|\mathcal{X}|} = 1 - \sum_{\ell=1}^{n-1} \frac{|S_\ell|}{|\mathcal{X}|}$.

Notationally, for all $\ell \in [n]$, let $T_\ell = \bigcup_{m \geq \ell} S_m$. We construct individuals' utilities as follows: for each $\ell \in [n]$, for each individual $i \in \mathcal{X}$,

$$u_i(c_\ell) = \begin{cases} t^\ell & \text{if } i \in T^\ell \\ 0 & \text{otherwise} \end{cases}$$

First, note that for individuals $i \in S_\ell$, c_ℓ maximizes their utility $u_i(c_\ell) = t^\ell$. So consider the welfare-maximizing allocation that assigns every individual in S_ℓ to campaign c_ℓ; this allocation satisfies PIIF. The average social welfare can then be written as:

$$\begin{aligned} \frac{1}{|\mathcal{X}|} \cdot \sum_{\ell=0}^{n-1} \sum_{i \in S_\ell} u_i(c_\ell) &= \frac{|S_0|}{|\mathcal{X}|} + \sum_{\ell=1}^{n-1} \frac{|S_\ell|}{|\mathcal{X}|} \cdot t^\ell \\ &= \left(1 - \sum_{\ell=1}^{n-1} t^{-\ell}\right) + \sum_{\ell=1}^{n-1} t^{-\ell} \cdot t^\ell \\ &= n - \sum_{\ell=1}^{n-1} t^{-\ell} \end{aligned}$$

Next, consider similarity metrics defined by the utilities: For each task c_ℓ, we take $d_\ell(i,j) = |u_i(c_\ell) - u_j(c_\ell)|$. By the definition of u_i, under these similarity metrics, every pair of individuals $i,j \in T_\ell \times T_\ell$ are considered similar $d_\ell(i,j) = 0$. We fix D to be the total variation distance, in other words, $D(p_c, q_c) = |p_c - q_c|$. As such, any allocation π that satisfies IF must show c_ℓ to every individual $i \in T_\ell$ with some fixed probability $\alpha_\ell = \pi(i)_{c_\ell}$. We can compute the expression for the average social welfare of any such assignment as a function of the α_ℓ.

$$\frac{1}{|\mathcal{X}|} \cdot \sum_{\ell=0}^{n-1} \alpha_\ell \cdot \sum_{i \in T_\ell} u_i(c_\ell) = \alpha_0 \cdot \frac{|S_0|}{|\mathcal{X}|} + \sum_{\ell=1}^{n-1} \alpha_\ell \cdot t^\ell \cdot \frac{|T_\ell|}{|\mathcal{X}|}$$

$$= \alpha_0 \cdot \left(1 - \sum_{\ell=1}^{n-1} t^{-\ell}\right) + \sum_{\ell=1}^{n-1} \alpha_\ell \cdot t^\ell \cdot \sum_{m=\ell}^{n-1} t^{-m} \tag{12}$$

$$\leq \alpha_0 + \sum_{\ell=1}^{n-1} \alpha_\ell + \sum_{\ell=1}^{n-1} \alpha_\ell \cdot \sum_{m=\ell+1}^{n-1} t^{-m+\ell}$$

$$\leq \left(\sum_{\ell=0}^{n-1} \alpha_\ell\right) \cdot \left(1 + \sum_{m=2}^{n-1} t^{-m}\right) \tag{13}$$

$$= 1 + \sum_{m=2}^{n-1} t^{-m} \tag{14}$$

where (12) follows by expanding $\frac{|T_\ell|}{|\mathcal{X}|}$ in terms of $\frac{|S_m|}{|\mathcal{X}|} = t^{-m}$; (13) applies Hölder's inequality; and (14) uses the fact that individuals $i \in S_{n-1}$ are members $i \in T_\ell$ for all $\ell \in [n]$, so the sum of the probabilities $\sum_\ell \alpha_\ell \leq 1$.

Given a desired ε, we can take t large enough, the ratio between the social welfares exceeds $n - \varepsilon$. ◀

Optimality under IF

Intuitively, this construction highlights the fact that allowing further targeting and more ad campaigns to participate allows the gap in social welfare between the best IF and PIIF solutions to grow considerably. Note that this gap applies even if the platform's objective is to optimize social welfare, so the proof also shows a gap in worst-case utility achievable by the decision-maker under IF.

We remark that a corollary of our result is that the Dwork-Ilvento "RandomizeThen-Classify" mechanism [13] for composition under multi-task IF achieves worst-case optimal performance (in terms of both social welfare and utility to the platform). In particular, [13] give an algorithm (in a setting with limited information modeling "competitive composition") that allocates a fixed distribution $p \in \Delta(C)$ to all individuals – thus, satisfying IF – that achieves a $1/|C|$-fraction of the best unconstrained utility. Our result shows that no IF solution, even with full information, can achieve a better fraction of the achievable utility.

6 Discussion

In this section, we review additional related work, note some possible extensions within the preference-informed fairness framework, and conclude with a discussion of the strengths and limitations of our current approach.

6.1 Further related works

Since [12], a number of recent works have aimed to extend the "fairness through awareness" framework, including [27, 23, 15, 22, 20]. These works focus on translating the theoretical IF framework into practically-motivated settings.

Three recent works have suggested incorporating notions of individuals' *preferences* into the fairness definitions. First, [4] present EF as an alternative to IF and study its learning-theoretic properties. Their focus is on the question of generalization: given a classifier that is envy-free on a sample, is it approximately envy-free on the underlying distribution? Their main technical result is a positive answer to this question, when learning from a particular structured family of classifiers. An interesting open question is whether the generalization results for IF from [27] and EF from [4] can be combined to give generalization for PIIF.

Second, [34] considers two notions of fairness at the (weaker) group level: treatment parity and impact parity. Their main contribution is a relaxation of both definitions, allowing for solutions where every protected group is "better off" *on average*. From a technical perspective, achieving their notion requires solving a non-convex optimization problem even in the simple case of linear classifiers for two disjoint groups. Our approach is different in that it focuses on defining both fairness and preferences at the *individual* level. This allows for a significantly stronger fairness guarantee, as well as a much more general framework that supports any notion of benefit or preference individuals may have. Importantly, our notion provably admits efficient optimization for a rich class of preference relations.

Finally, independent of our work, [9] study and quantify trade-offs between individual fairness and utility in an online version of the targeted advertising problem. [9] also observe that IF can come at a high cost to utility in the multiple-task setting of [13], but propose a different relaxation. Under their notion, every individual i chooses a subset of the outcomes $S_i \subseteq C$ and is guaranteed that their probability of seeing an ad from S_i is greater than the probability of every other individual of seeing an ad from S_i. Such a guarantee can be viewed as a variant of EF over a more restricted class of preferences than those we consider. The main distinction from PIIF is that this notion ignores the distance metrics entirely and in this sense resembles envy-freeness more than individual-fairness.

6.2 Preference-informed group fairness

In this work, our focus was on incorporating individual preferences into the metric-based individual fairness framework Dwork et al. [12]. The space of fairness definitions, however, is large, and different definitions may be more appropriate in different contexts.

A different approach for defining fairness, often referred to as "group fairness," proceeds as follows. A protected attribute, such as race or gender, induces a partition of the individuals into a small number of groups. For simplicity, we focus on the case where there is a single protected group, S, where the rest of the population is denoted $T = \mathcal{X} \setminus S$. A classifier is considered fair if it achieves parity of some statistical measure across these groups. Group fairness notions are typically weaker than individual notions of fairness: they only provide a guarantee for the "average" member of the protected groups and might allow blatant unfairness towards a single individual or even large subgroups; indeed, the shortcomings of group notions motivated the original work on "fairness through awareness" and subsequent works [12, 21, 17, 23]. Although group fairness notions can be fragile, they are widely studied and used due to their simplicity and due to the fact that they are easier to enforce and implement (for example, they do not require a task-specific similarity metric).

In principle, much of the reasoning behind our argument for incorporating preferences into IF [12] also extends to group-fairness notions. In this section, we show how we might augment a common group notion, called *Statistical Parity* (SP), to incorporate preferences. When there is a clearly "desirable" outcome, SP aims to protect the group S by guaranteeing equal average exposure to the desired outcome.

▶ **Definition 20** (Statistical parity). *A binary classifier $h : \mathcal{X} \to \{\pm 1\}$ satisfies (exact) statistical parity with respect to S if*

$$\mathop{\mathbf{Pr}}_{i\sim S}[h(i) = 1] = \mathop{\mathbf{Pr}}_{i\sim T}[h(i) = 1]$$

In our context, when individuals have diverse preferences over the outcome space, enforcing SP may again come at a cost to members of S, the group that SP aims to protect. As a concrete example, suppose everyone in $\mathcal{X}$ prefers the outcome $+1$, with the exception of some fraction of S, denoted S', who prefer the outcome -1. In this case, the statistical parity constraints prevents the solution where $h(i) = +1$ for $i \in X \setminus S'$ and $h(i) = -1$ for $i \in S$, which from the individuals' perspective is optimal.

Building on this intuition, we extend the set of classifiers we deem fair. Assuming every individual $i \in X$ has a preference relation over $\{\pm 1\}$ (or even distributions over $\{\pm 1\}$), *preference-informed statistical parity* (PISP) allows deviations from SP, as long as they are aligned with the individuals' preferences.

▶ **Definition 21** (Preference-informed statistical parity). *A binary classifier $h : X \to \{\pm 1\}$ satisfies preference-informed statistical parity with respect to S if there exists an alternative classifier, $h' : X \to \{\pm 1\}$, such that:*

$$\forall j \in T, h'(j) = h(j)$$
$$\forall i \in S, h(i) \succeq_i h'(i)$$
$$\mathop{\mathbf{Pr}}_{i\sim S}[h'(i) = 1] = \mathop{\mathbf{Pr}}_{i\sim T}[h'(i) = 1]$$

That is, fixing the outcomes members of T receive under h, every single member of S prefers their current outcome over what they would have received under a classifier satisfying statistical parity. Importantly, the guarantee is still with respect to the preferences of the *individual* members of S.

We conclude with several remarks regarding PISP. First, note that PISP only enriches the set of solutions that satisfy SP; any classifier that satisfies SP also satisfies PISP, by taking the alternative $h' = h$. The classifier welfare-maximizing classifier, where each individual is assigned their favorite outcome, is considered fair. For example, revisiting our example above, the classifier that gives $+1$ to $\mathcal{X} \setminus S'$, and -1 to S' is fair, because the alternative classifier that gives *everyone* $+1$ satisfies the PISP constraints. Finally, we argue that PISP maintains the core of the fairness guarantee of SP. For example, consider the classifier that assigns $+1$ to members of T and -1 to members of S. This classifier benefits the members of T in a way that is *not* aligned with the preferences of all members in S; rightfully, it does not satisfy PISP, because $i \in S \setminus S'$ are harmed.

6.3 Revisiting the assumptions underlying PIIF

Three main assumptions underlie our work. The first is that the outcome space is taken as a given. This could be problematic if the outcomes themselves are biased, e.g., tailored to the preferences of the majority, or worse yet, harmful to the minority. A biased outcome space would also present a problem for both IF and EF, which PIIF does not escape entirely. Still, PIIF may ameliorate the issue, by allowing the minority to receive outcomes that they prefer. We see a study of fairness of the outcomes themselves as an exciting direction for further inquiry.

The second assumption is that any deviation from an IF solution that is aligned with individuals' preferences should still be considered fair. This assumption follow from the perspective that "fair" allocation algorithms should protect the welfare of individuals; this

perspective naturally extends the perspective underlying IF that similar individuals should be treated similarly. As discussed in [18, 19], in other (legal) settings, the notion of "fairness" may necessarily imply "treatment as equal," and notions of individual fairness may not apply. In such settings, the societal notion of fairness may require going against individual preferences. Handling such settings lies beyond the scope of our current work that focuses on the computer science notions of individual fairness, in which the objective is to provide strong protections from discrimination to the individuals themselves.

The third assumption is that the individuals' preferences are known. This is certainly the most nontrivial technical assumption we make; nevertheless, there are established techniques for learning utility-functions from observed behaviour [8]. We also note that accurately learning preferences could often be in the interest of the decision-maker (for example, ad platforms often claim that their implementations of targeted advertising are in line with users' interests [35]). Still, any practical estimation of the preferences runs the risk of injecting further bias during the learning process (for example, if the minority's preferences are estimated with lesser accuracy) and, therefore, mandates special attention in future research.

References

1 Muhammad Ali, Piotr Sapiezynski, Miranda Bogen, Aleksandra Korolova, Alan Mislove, and Aaron Rieke. Discrimination through optimization: How Facebook's ad delivery can lead to skewed outcomes. *22nd ACM Conference on Computer-Supported Cooperative Work and Social Computing (CSCW)*, 2019. arXiv:1904.02095.

2 Julia Angwin, Noam Scheiber, and Ariana Tobin. Dozens of Companies Are Using Facebook to Exclude Older Workers From Job Ads. *ProPublica*, Dec 20, 2017. URL: https://www.propublica.org/article/facebook-is-letting-job-advertisers-target-only-men.

3 Julia Angwin, Ariana Tobin, and Madeleine Varner. Facebook (Still) Letting Housing Advertisers Exclude Users by Race. *ProPublica*, Nov. 21, 2017. URL: https://www.propublica.org/article/facebook-advertising-discrimination-housing-race-sex-national-origin.

4 Maria-Florina Balcan, Travis Dick, Ritesh Noothigattu, and Ariel D Procaccia. Envy-Free Classification. *arXiv preprint*, 2018. arXiv:1809.08700.

5 Vijay S Bawa. Optimal rules for ordering uncertain prospects. *Journal of Financial Economics*, 2(1):95–121, 1975.

6 Katie Benner, Glenn Thrush, and Mike Isaac. Facebook Engages in Housing Discrimination With Its Ad Practices, U.S. Says. *The New York Times*, Mar 28, 2019. URL: https://www.nytimes.com/2019/03/28/us/politics/facebook-housing-discrimination.html.

7 Miranda Bogen. All the Ways Hiring Algorithms Can Introduce Bias. *Harvard Business Review*, May 6, 2019. URL: https://hbr.org/2019/05/all-the-ways-hiring-algorithms-can-introduce-bias.

8 Urszula Chajewska, Daphne Koller, and Dirk Ormoneit. Learning an agent's utility function by observing behavior. In *ICML*, pages 35–42, 2001.

9 Shuchi Chawla, Christina Ilvento, and Meena Jagadeesan. Individual Fairness in Sponsored Search Auctions. *arXiv preprint*, 2019. arXiv:1906.08732.

10 Jeffrey Dastin. Amazon scraps secret AI recruiting tool that showed bias against women. *Reuters*, Oct 10, 2018.

11 Amit Datta, Michael Carl Tschantz, and Anupam Datta. Automated experiments on ad privacy settings. *Proceedings on Privacy Enhancing Technologies*, 2015(1):92–112, 2015.

12 Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. Fairness through awareness. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference (ITCS)*, pages 214–226. ACM, 2012.

13 Cynthia Dwork and Christina Ilvento. Fairness Under Composition. In *10th Innovations in Theoretical Computer Science Conference, ITCS 2019, January 10-12, 2019, San Diego, California, USA*, pages 33:1–33:20, 2019.

14 Duncan K Foley. *Resource allocation and the public sector*. PhD thesis, Yale University, 1967.

15 Stephen Gillen, Christopher Jung, Michael Kearns, and Aaron Roth. Online Learning with an Unknown Fairness Metric. *NeurIPS*, 2018.

16 Josef Hadar and William R Russell. Rules for ordering uncertain prospects. *The American economic review*, 59(1):25–34, 1969.

17 Ursula Hébert-Johnson, Michael P Kim, Omer Reingold, and Guy N Rothblum. Multicalibration: Calibration for the (Computationally-Identifiable) Masses. *ICML*, 2018.

18 Deborah Hellman. Two concepts of discrimination. *Virgina Law Review*, 102:895, 2016.

19 Deborah Hellman. Measuring Algorithmic Fairness. *Virginia Law Review*, ssrn.3418528, 2019.

20 Christopher Jung, Michael Kearns, Seth Neel, Aaron Roth, Logan Stapleton, and Zhiwei Steven Wu. Eliciting and Enforcing Subjective Individual Fairness. *arXiv*, 2019. arXiv:1905.10660.

21 Michael Kearns, Seth Neel, Aaron Roth, and Zhiwei Steven Wu. Preventing Fairness Gerrymandering: Auditing and Learning for Subgroup Fairness. *ICML*, 2018.

22 Michael Kearns, Aaron Roth, and Saeed Sharifi-Malvajerdi. Average Individual Fairness: Algorithms, Generalization and Experiments. *arXiv*, 2019. arXiv:1905.10607.

23 Michael P. Kim, Omer Reingold, and Guy N. Rothblum. Fairness Through Computationally-Bounded Awareness. *NeurIPS*, 2018.

24 Anja Lambrecht and Catherine E Tucker. Algorithmic bias? An empirical study into apparent gender-based discrimination in the display of STEM career ads. *SSRN*, ssrn.2852260, 2018.

25 Laura Murphy. Facebook's Civil Rights Audit – Progress Report, June 30, 2019. URL: https://fbnewsroomus.files.wordpress.com/2019/06/civilrightaudit_final.pdf.

26 Cathy O'Neil. *Weapons of math destruction: How big data increases inequality and threatens democracy*. Broadway Books, 2017.

27 Guy Rothblum and Gal Yona. Probably Approximately Metric-Fair Learning. In *ICML*, 2018.

28 Sheryl Sandberg. Doing More to Protect Against Discrimination in Housing, Employment and Credit Advertising. *Facebook Newsroom*, Mar 19, 2019. URL: https://newsroom.fb.com/news/2019/03/protecting-against-discrimination-in-ads/.

29 Ariana Tobin. HUD sues Facebook over housing discrimination and says the company's algorithms have made the problem worse. *ProPublica*, Mar 28, 2019. URL: https://www.propublica.org/article/hud-sues-facebook-housing-discrimination-advertising-algorithms.

30 Ariana Tobin and Jeremy B. Merrill. Facebook Is Letting Job Advertisers Target Only Men. *ProPublica*, Sept 18, 2018. URL: https://www.propublica.org/article/facebook-is-letting-job-advertisers-target-only-men.

31 Upturn. Upturn Amicus Brief in Onuoha v. Facebook, Nov 16, 2018. URL: https://www.courtlistener.com/recap/gov.uscourts.cand.304918/gov.uscourts.cand.304918.76.1.pdf.

32 Hal Varian. Efficiency, equity and envy. *Journal of Economic Theory*, 9:63–91, 1974.

33 John Von Neumann and Oskar Morgenstern. *Theory of Games and Economic Behavior (Commemorative Edition)*. Princeton University Press, 2007.

34 Muhammad Bilal Zafar, Isabel Valera, Manuel Rodriguez, Krishna Gummadi, and Adrian Weller. From parity to preference-based notions of fairness in classification. In *Advances in Neural Information Processing Systems*, pages 229–239, 2017.

35 Mark Zuckerberg. The Facts About Facebook. *The Wall Street Journal*, Jan 24, 2019. Opinion in The Wall Street Journal https://www.wsj.com/articles/the-facts-about-facebook-11548374613.

On a Theorem of Lovász that $\hom(\cdot, H)$ Determines the Isomorphism Type of H

Jin-Yi Cai
Department of Computer Sciences, University of Wisconsin-Madison, USA
jyc@cs.wisc.edu

Artem Govorov[1]
Department of Computer Sciences, University of Wisconsin-Madison, USA
hovarau@cs.wisc.edu

Abstract

Graph homomorphism has been an important research topic since its introduction [14]. Stated in the language of binary relational structures in that paper [14], Lovász proved a fundamental theorem that the graph homomorphism function $G \mapsto \hom(G, H)$ for 0-1 valued H (as the adjacency matrix of a graph) determines the isomorphism type of H. In the past 50 years various extensions have been proved by Lovász and others [15, 9, 1, 19, 17]. These extend the basic 0-1 case to admit vertex and edge weights; but always with some restrictions such as all vertex weights must be positive. In this paper we prove a general form of this theorem where H can have arbitrary vertex and edge weights. An innovative aspect is that we prove this by a surprisingly simple and unified argument. This bypasses various technical obstacles and unifies and extends all previous known versions of this theorem on graphs. The constructive proof of our theorem can be used to make various complexity dichotomy theorems for graph homomorphism *effective*, i.e., it provides an algorithm that for any H either outputs a P-time algorithm solving $\hom(\cdot, H)$ or a P-time reduction from a canonical #P-hard problem to $\hom(\cdot, H)$.

2012 ACM Subject Classification Mathematics of computing → Graph theory; Theory of computation → Problems, reductions and completeness

Keywords and phrases Graph homomorphism, Partition function, Complexity dichotomy, Connection matrices and tensors

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.17

Related Version A full version of this paper is available at `https://arxiv.org/abs/1909.03693`.

Funding *Jin-Yi Cai*: Supported by NSF CCF-1714275.
Artem Govorov: Supported by NSF CCF-1714275.

Acknowledgements We thank the anonymous referees for helpful comments and suggestions. We also thank Guus Regts for telling us of his work, and for very insightful comments on this paper.

1 Introduction

More than 50 years ago the concept of graph homomorphism was introduced [14, 13]. Given two graphs G and H, a mapping from $V(G)$ to $V(H)$ is called a homomorphism if every edge of G is mapped to an edge of H. The graphs G and H can be either both directed or undirected. Presented in the language of binary relational structures, Lovász proved in that paper [14] the following fundamental theorem about graph homomorphism: If H and H' are two graphs, then they are isomorphic iff they define the same counting graph homomorphism

[1] Artem Govorov is the author's preferred spelling of his name, rather than the official spelling Artsiom Hovarau.

function, namely, for every G, the number of homomorphisms from G to H is the same as that from G to H'. This number is denoted by $\hom(G, H)$. (Formal definitions are in Section 2.)

In [14] the graph H is a 0-1 adjacency matrix; there are no vertex and edge weights. In [9] Freedman, Lovász and Schrijver define a weighted version of the homomorphism function $\hom(\cdot, H)$, where H has *positive* vertex weights and *real* edge weights. The paper [9] investigates what graph properties can be expressed as such graph homomorphism functions. They gave a necessary and sufficient condition for this expressibility. This work has been extended to the case with arbitrary vertex and edge weights in a field [5], and to "edge models", e.g., [20, 18]. A main technical tool introduced in [9] is the so-called graph algebras. In [15] Lovász further investigated these graph algebras and proved precise bounds for their dimensions. These dimensions are a quantitative account of the space of all isomorphisms from H to H'. They are expressed in a theory of labeled graphs. Schrijver [19] studied the function $\hom(\cdot, H)$ where H is an undirected graph with complex edge weights (but all vertex weights are restricted to 1). He also gave a characterization of a graph property expressible in this form, and proved that $\hom(\cdot, H) = \hom(\cdot, H')$ implies that $H \cong H'$ for undirected graphs with complex edge weights (but unit vertex weights). Regts in [18], in addition to finding interesting connections between edge-coloring models and invariants of the orthogonal group, also proved multiple theorems in the framework of graph homomophisms (corresponding to "vertex models") requiring that all (nonempty) sums of vertex weights be nonzero. The possibility that vertex weights may sum to zero has been a difficult point. Our main result is to extend this isomorphism theorem to both directed and undirected graphs with arbitrary vertex and edge weights. We also determine the precise values of the dimensions of the corresponding graph algebras. A variant of our result, in terms of dimensions of associated algebras was proved by Goodall, Regts and Vena [11]; please see Remark 12 in Section 5.

To prove our theorem, we introduce a surprisingly simple and completely elementary argument, which we call the *Vandermonde Argument*. All of our results are proved by this one technique.

Two vertices i and j in an unweighted graph H are called *twins* iff the neighbor sets of i and j are identical. For weighted graphs, i and j are called twins iff the edge weights $\beta(i, k) = \beta(j, k)$ (and for directed graphs also $\beta(k, i) = \beta(k, j)$) for all k. In order to identify the isomorphism class of H, a natural step is to combine twin vertices. This creates a super vertex with a combined vertex weight (even when originally all vertices are unweighted, i.e., have weight 1). After this "twin reduction" step, our isomorphism theorem can be stated. The following is a simplified form:

▶ **Theorem 1.** *Let $\mathbb{F}$ be a field of characteristic 0. Let H and H' be (directed or undirected) weighted graphs with arbitrary vertex and edge weights from $\mathbb{F}$. Without loss of generality all individual vertex weights are nonzero. Suppose H and H' are twin-free. If for all simple graphs G (i.e., without loops and multiedges),*

$$\hom(G, H) = \hom(G, H'), \tag{1}$$

then the graphs H and H' are isomorphic as weighted graphs, i.e., there is a bijective map from H to H' that preserves all vertex and edge weights.

Theorem 1 is the special case of $k = 0$ of the more general Theorem 2 which deals with k-labeled graphs. In Section 8 we also determine the dimensions of the corresponding graph algebras in terms of the rank of the so-called connection tensors, introduced in [5]. These improve the corresponding theorems in [15, 19, 18] as follows.

From the main theorem (Theorem 2.2) of [15] we generalize from positive vertex weights and real edge weights to arbitrary weights. The main technique in [15] is algebraic. The proof relies on the notion of quantum graphs and structures built from them, and uses idempotent elements in the graph algebras. Similarly, from the isomorphism theorem in [19] we generalize from unit vertex weights and complex edge weights to arbitrary weights. Also we allow directed and undirected weighted graphs H. Theorem 2 also weakens the condition (1) on G to simple graphs (i.e., no multiedges or loops). Schrijver's proof technique is different from that of Lovász [15], but is also algebraic and built on quantum graphs. He uses a Reynolds operator and the Möbius transform (of a graph). The results of Lovász [15] and Schrijver [19] are incomparable. While requiring all vertex weights to be positive is not unreasonable, it is nonetheless a severe restriction, and has been a technical obstacle to all existing proofs. In Regts' thesis [18], multiple theorems were proved with the explicit requirement that all (nonempty) sums of vertex weights be nonzero, which circumvented this issue. In this paper, we allow arbitrary vertex weights with no assumptions. In particular, H can have arbitrary complex vertex and edge weights.

However, more than the explicit strengthening of the theorems, we believe the most innovative aspect of this work is that we found a direct elementary argument that bypassed various technical obstacles and unified all previously known versions. We can also show that the only restriction – $\mathbb{F}$ has characteristic 0 – cannot be removed, and thus our results are the most general extensions on graphs. We give counterexamples for fields of finite characteristic in Section 7.

This line of work has already led to significant applications in the graph limit literature, such as on quasi-random graphs [16]. In [17] Lovász and B. Szegedy also studied these graph algebras where "contractors" and "connectors" are used. In our treatment these "contractors" and "connectors" can also be constructed with simple graphs.

In terms of applications to complexity theory, there has been a series of significant complexity dichotomy theorems on counting graph homomorphisms which show that the function $\hom(\cdot, H)$ is either P-time computable or #P-hard, *depending* on H [7, 8, 2, 12, 21, 10, 4, 6, 3]. These theorems differ in the scope of what types of H are allowed, from 0-1 valued to complex valued, from undirected to directed. In all these theorems a P-time tractability condition on H is given, such that if H satisfies the condition then $\hom(\cdot, H)$ is P-time computable, otherwise $\hom(\cdot, H)$ is #P-hard. In the latter case, the theorem asserts that there is a P-time reduction from a canonical #P-hard problem to $\hom(\cdot, H)$. *However*, various *pinning lemmas* are proved nonconstructively; for undirected complex weighted graphs [4] it was unknown how to make this constructive. Consequently, there was no known algorithm to produce a #P-hardness reduction from H. Because the proof in this paper is constructive, it can be applied as a crucial step in making all these dichotomy theorems *effective*, i.e., we can obtain an algorithm that for any H either outputs a P-time algorithm solving $\hom(\cdot, H)$ or a P-time reduction from a canonical #P-hard problem to $\hom(\cdot, H)$.

2 Preliminaries

We first recap the notion of weighted graph homomorphisms [9], but state it for an arbitrary field $\mathbb{F}$. We let $[k] = \{1, \ldots, k\}$ for integer $k \geq 0$. In particular, $[0] = \emptyset$. By covention $\mathbb{F}^0 = \{\emptyset\}$, and $0^0 = 1$ in $\mathbb{Z}$, $\mathbb{F}$, etc. Often we discuss both directed and undirected graphs together.

An ($\mathbb{F}$-)*weighted graph* H is a finite (di)graph with a weight $\alpha_H(i) \in \mathbb{F} \setminus \{0\}$ associated with each vertex i (0-weighted vertices can be deleted) and a weight $\beta_H(i, j) \in \mathbb{F}$ associated with each edge ij (or loop if $i = j$). For undirected graphs, $\beta_H(i, j) = \beta_H(j, i)$. It is convenient to assume that H is a complete graph with a loop at all nodes by adding all

missing edges and loops with weight 0. Then H is described by an integer $q = |V(H)| \geq 0$ (H can be the empty graph), a nowhere zero vector $\alpha = (\alpha_H(1), \ldots, \alpha_H(q)) \in \mathbb{F}^q$ and a matrix $B = (\beta_H(i,j)) \in \mathbb{F}^{q \times q}$. An isomorphism from H to H' is a bijective map from $V(H)$ to $V(H')$ that preserves vertex and edge weights.

According to [9], let G be an unweighted graph (with possible multiple edges, but no loops) and H a weighted graph given by (α, B), we define

$$\begin{aligned} \hom(G,H) &= \sum_{\phi\colon V(G)\to V(H)} \alpha_\phi \hom_\phi(G,H) \\ &= \sum_{\phi\colon V(G)\to V(H)} \prod_{u\in V(G)} \alpha_H(\phi(u)) \prod_{uv\in E(G)} \beta_H(\phi(u),\phi(v)). \end{aligned} \tag{2}$$

The unweighted case is when all node weights are 1 and all edge weights are 0-1 in H, and $\hom(G,H)$ is the number of homomorphisms from G into H.

A k-labeled graph ($k \geq 0$) is a finite graph in which k nodes are labeled by $1, 2, \ldots, k$ (the graph can have any number of unlabeled nodes). Two k-labeled graphs are isomorphic if there is a label-preserving isomorphism between them. U_k denotes the k-labeled graph on k nodes with no edges. In particular, U_0 is the empty graph with no nodes and no edges. The *product* of two k-labeled graphs G_1 and G_2 is defined as follows: take their disjoint union, and then identify nodes with the same label. Hence for two 0-labeled graphs, $G_1 G_2 = G_1 \sqcup G_2$ (disjoint union). Clearly, the graph product is associative and commutative with the identity U_k, so the set of all (isomorphism classes) of k-labeled graphs togegher with the product operation forms a commutative monoid which we denote by $\mathcal{PLG}[k]$. We denote by $\mathcal{PLG}^{\text{simp}}[k]$ the submonoid of simple graphs in $\mathcal{PLG}[k]$; these are graphs with no loops, at most one edge between any two vertices i and j, and no edge between labeled vertices. A directed labeled graph is simple if its underlying undirected one is simple; in particluar, for any i and j, we require that if $i \to j$ is an edge then $j \to i$ is not an edge. Clearly, $\mathcal{PLG}^{\text{simp}}[k]$ is closed under the product operation (for both directed and undirected types).

Fix a weighted graph $H = (\alpha, B)$. For any k-labeled graph G and mapping $\psi\colon [k] \to V(H)$, let

$$\hom_\psi(G,H) = \sum_{\substack{\phi\colon V(G)\to V(H) \\ \phi \text{ extends } \psi}} \frac{\alpha_\phi}{\alpha_\psi} \hom_\phi(G,H), \tag{3}$$

where ϕ extends ψ means that if $u_i \in V(G)$ is labeled by $i \in [k]$ then $\phi(u_i) = \psi(i)$, and $\alpha_\psi = \prod_{i=1}^k \alpha_H(\psi(i))$, $\alpha_\phi = \prod_{v\in V(G)} \alpha_H(\phi(v))$, so $\frac{\alpha_\phi}{\alpha_\psi}$ is the product of vertex weights of α_ϕ *not* in α_ψ. Then

$$\hom(G,H) = \sum_{\psi\colon [k]\to V(H)} \alpha_\psi \hom_\psi(G,H). \tag{4}$$

When $k = 0$, we only have the empty map $\emptyset$ with the domain $\emptyset$. Then $\hom(G,H) = \hom_\emptyset(G,H)$ for every $G \in \mathcal{PLG}[k]$. The functions $\hom_\psi(\cdot, H)$ where $\psi\colon [k] \to V(H)$ and $k \geq 0$ satisfy

$$\begin{cases} \hom_\psi(G_1G_2,H) = \hom_\psi(G_1,H)\hom_\psi(G_2,H), \quad G_1, G_2 \in \mathcal{PLG}[k], \\ \hom_\psi(U_k,H) = 1. \end{cases} \tag{5}$$

Given a directed or undirected $\mathbb{F}$-weighted graph H, we call two vertices $i, j \in V(H)$ *twins* if for every vertex $\ell \in V(H)$, $\beta_H(i,\ell) = \beta_H(j,\ell)$ and $\beta_H(\ell,i) = \beta_H(\ell,j)$. Note that the vertex weights $\alpha_H(w)$ do not participate in this definition. If H has no twins, we call it twin-free.

The twin relation partitions $V(H)$ into nonempty equivalence classes, $I_1, \ldots, I_s$ where $s \geq 0$. We can define a *twin contraction* graph $\widetilde{H}$, having $I_1, \ldots, I_s$ as vertices, with vertex weight $\sum_{t \in I_r} \alpha_H(t)$ for I_r, and edge weight from I_r to I_q to be $\beta_H(u, v)$ for some arbitrary $u \in I_r$ and $v \in I_q$. After that, we remove all vertices in $\widetilde{H}$ with zero vertex weights together with all incident edges (still called $\widetilde{H}$). This defines a twin-free $\widetilde{H}$. Clearly, $\hom(G, H) = \hom(G, \widetilde{H})$ for all G.

We denote by $\mathrm{Isom}(H, H')$ the set of $\mathbb{F}$-weighted graph isomorphisms from H to H' and by $\mathrm{Aut}(H)$ the group of ($\mathbb{F}$-weighted) graph automorphisms of H.

It is obvious that for directed (or undirected) $\mathbb{F}$-weighted graphs H and H', and the maps $\varphi\colon [k] \to V(H)$ and $\psi\colon [k] \to V(H')$ such that $\psi = \sigma \circ \varphi$ for some isomorphism $\sigma\colon V(H) \to V(H')$ from H to H', we have $\hom_\varphi(G, H) = \hom_\psi(G, H')$ for every $G \in \mathcal{PLG}[k]$.

3 Our results

Theorem 1 is a direct consequence of the case $k = 0$ of the following Main Theorem.

▶ **Theorem 2.** *Let $\mathbb{F}$ be a field of characteristic 0. Let H, H' be (directed or undirected) $\mathbb{F}$-weighted graphs such that H is twin-free and $m = |V(H)| \geq m' = |V(H')|$. Suppose $\varphi\colon [k] \to V(H)$ and $\psi\colon [k] \to V(H')$ where $k \geq 0$. If $\hom_\varphi(G, H) = \hom_\psi(G, H')$ for every $G \in \mathcal{PLG}^{\mathrm{simp}}[k]$ then there exists an isomorphism of $\mathbb{F}$-weighted graphs $\sigma\colon V(H) \to V(H')$ from H to H' such that $\psi = \sigma \circ \varphi$ (a fortiori, H' is twin-free and $m = m'$).*

In Section 8 we will give our results about the space of such isomorphisms, expressed in terms of the dimensions of the corresponding graph algebras.

In Corollaries 3 to 6, $\mathrm{char}\, \mathbb{F} = 0$. The following two corollaries extend Lovász's theorem (and lemmas that are of independent interest) in [15] from real edge weight and positive vertex weight. Furthermore, it holds for both directed and undirected graphs, and the condition on G is weakened so that it is sufficient to assume it for simple graphs only. The fact that the theorem holds under the condition $\hom(G, H) = \hom(G, H')$ for loopless graphs G is important in making the complexity dichotomies *effective* in the sense defined in Section 1.

Theorem 2 is stated in a technical way where we only assume H is twin-free and $|V(H)| \geq |V(H')|$. Corollary 3 is a symmetric statement.

▶ **Corollary 3.** *Let H, H' be (directed or undirected) $\mathbb{F}$-weighted twin-free graphs. Let $\varphi\colon [k] \to V(H)$ and $\psi\colon [k] \to V(H')$ where $k \geq 0$. If $\hom_\varphi(G, H) = \hom_\psi(G, H')$ for every $G \in \mathcal{PLG}^{\mathrm{simp}}[k]$, then there exists an isomorphism σ from H to H' such that $\psi = \sigma \circ \varphi$.*

▶ **Corollary 4.** *Let H, H' be $\mathbb{F}$-weighted twin-free graphs, either both directed or both undirected. If $\hom(G, H) = \hom(G, H')$ for every* simple *graph G, then H and H' are isomorphic as $\mathbb{F}$-weighted graphs.*

For edge weighted graphs with unit vertex weight, the requirement of twin-freeness can be dropped. The following two corollaries directly generalize Schrijver's theorem (Theorem 2) in [19]. Corollary 6 is a restatement of Corollary 5 using the terminology in [19]. Here we strengthen his theorem by requiring the condition $\hom(G, H) = \hom(G, H')$ for only simple graphs G. Also our result holds for fields $\mathbb{F}$ of characteristic 0 generalizing from $\mathbb{C}$, and for directed as well as undirected graphs.

▶ **Corollary 5.** *Let H, H' be (directed or undirected) $\mathbb{F}$-edge-weighted graphs. If $\hom(G, H) = \hom(G, H')$ for every* simple *graph G, then H and H' are isomorphic as $\mathbb{F}$-weighted graphs.*

▶ **Corollary 6.** *Let $A \in \mathbb{F}^{m\times m}$ and $A' \in \mathbb{F}^{m'\times m'}$. Then $\hom(G, A) = \hom(G, A')$ for every* simple *graph G iff $m = m'$ and there is a permutation matrix $P \in \mathbb{F}^{m\times m}$ such that $A' = P^T AP$.*

Our proof of Theorem 2 will show that for any given H, H', there is an explicitly constructed finite family of graphs in $\mathcal{PLG}^{\text{simp}}[k]$ such that the condition for all $G \in \mathcal{PLG}^{\text{simp}}[k]$ can be replaced with for all G in this family, thus we have an explicit finitary family of test graphs. Moreover, this provides an explicit set of "witnesses" that can be used to make various complexity dichotomy theorems for graph homomorphism *effective*, in particular, making the pinning steps in [4] computable, which was an open problem.

4 Technical statements

We start with an exceedingly simple lemma, based on which all of our results will be derived. We will call this lemma and its corollary the *Vandermonde Argument*.

▶ **Lemma 7.** *Let $n \geq 0$, and $a_i, x_i \in \mathbb{F}$ for $1 \leq i \leq n$. Suppose*

$$\sum_{i=1}^{n} a_i x_i^j = 0, \qquad \text{for all } \ 0 \leq j < n. \tag{6}$$

Then for any function $f\colon \mathbb{F} \to \mathbb{F}$, we have $\sum_{i=1}^{n} a_i f(x_i) = 0$.

▶ Remark 8. The statement is vacuously true if $n = 0$, since an empty sum is 0. If (6) is true for $1 \leq j \leq n$, then the same conclusion holds for any function f satisfying $f(0) = 0$.

Proof. We may assume $n \geq 1$. We partition $[n]$ into $\bigsqcup_{\ell=1}^{p} I_\ell$ such that i, i' belong to the same I_ℓ iff $x_i = x_{i'}$. Then (6) is a Vandermonde system of rank p with a solution $(\sum_{i\in I_\ell} a_i)_{\ell\in[p]}$. Thus $\sum_{i\in I_\ell} a_i = 0$ for all $1 \leq \ell \leq p$. It follows that $\sum_{i=1}^{n} a_i f(x_i) = 0$ for any function $f\colon \mathbb{F} \to \mathbb{F}$. We also note that if (6) is true for $1 \leq j \leq n$, then the same proof works except when some $x_i = 0$. In that case, we can separate out the term $\sum_{i\in I_{\ell_0}} a_i$ for the unique I_{ℓ_0} that contains this i, and we get a Vandermonde system of rank $p - 1$ on the other terms $(\sum_{i\in I_\ell} a_i)_{\ell\in[p],\ell\neq\ell_0}$, which must be all zero. ◀

By iteratively applying Lemma 7 we get the following Corollary.

▶ **Corollary 9.** *Let I be a finite (index) set, $s \geq 1$, and $a_i, b_{ij} \in \mathbb{F}$ for all $i \in I, j \in [s]$. Further, let $I = \bigsqcup_{\ell\in[p]} I_\ell$ be the partition of I into equivalence classes, where i, i' are equivalent iff $b_{ij} = b_{i'j}$ for all $j \in [s]$. If $\sum_{i\in I} a_i \prod_{j\in[s]} b_{ij}^{\ell_j} = 0$, for all choices of $(\ell_1, \ldots, \ell_s)$ where each $0 \leq \ell_j < |I|$, then $\sum_{i\in I_\ell} a_i = 0$ for every $\ell \in [p]$.*

Proof. We iteratively apply Lemma 7. First, we define an equivalence relation where i, i' belong to the same equivalence class $\widetilde{I}$ iff $b_{is} = b_{i's}$. For any $\widetilde{I}$, choose f with $f(x) = 1$ for $x = b_{is}$ where $i \in \widetilde{I}$, and $f(x) = 0$ otherwise. After the first application we get $\sum_{i\in\widetilde{I}} a_i \prod_{j\in[s-1]} b_{ij}^{\ell_j} = 0$, for an arbitrary $\widetilde{I}$, and all $0 \leq \ell_j < |I|, j \in [s-1]$. The Corollary follows after applying Lemma 7 s times. ◀

5 Proof of Main Theorem

In this conference version for the sake of simplicity of presentation we prove for undirected graphs; the full version has proofs for directed graphs as well as other results.

We may assume that H is on the vertex set $V(H) = [m]$, given by vertex and edge weights $(\alpha_i)_{i\in[m]} \in (\mathbb{F} \setminus \{0\})^m$, $(\beta_{ij})_{i,j\in[m]} \in \mathbb{F}^{m\times m}$. Similarly, H' is on $V(H') = [m']$, given by $(\alpha'_i)_{i\in[m']} \in (\mathbb{F} \setminus \{0\})^{m'}$ and $(\beta'_{ij})_{i,j\in[m']} \in \mathbb{F}^{m'\times m'}$.

We first make a technical condition of "super surjectivity"; it will be removed in Theorem 2.

▶ **Lemma 10.** *Let H, H' be undirected $\mathbb{F}$-weighted graphs such that H is twin-free and $m \geq m'$. Suppose $\varphi\colon [k] \to V(H)$ and $\psi\colon [k] \to V(H')$ where $k \geq 0$. Assume φ is "super surjective", namely: $|\varphi^{-1}(u)| \geq 2m^2$ for every $u \in V(H)$. If $\hom_\varphi(G, H) = \hom_\psi(G, H')$ for every $G \in \mathcal{PLG}^{\mathrm{simp}}[k]$ then there exists an isomorphism σ from H to H' such that $\psi = \sigma \circ \varphi$.*

Proof. Assume $m \geq 1$ (the case $m = 0$ is trivial). Taking any $u \in V(H)$, we get $k \geq |\varphi^{-1}(u)| \geq 2m^2 > 0$, thus $m' \geq |\psi([k])| \geq 1$. For each $\kappa = (b_i)_{i\in[k]} \in \{0,1\}^k$, we can define a graph $G_\kappa \in \mathcal{PLG}^{\mathrm{simp}}[k]$:

> $V(G_\kappa) = \{u_1, \ldots, u_k, v\}$, with each u_i labeled i. For each $i \in [k]$, there is an edge (v, u_i) iff $b_i = 1$. There are no other edges.

We now define a specific set of G_κ, given φ and ψ. We can partition $[k] = \bigsqcup_{i=1}^m I_i$ where each $I_i = \varphi^{-1}(i)$ and $|I_i| \geq 2m^2$. For every $i \in [m]$, since $|I_i| \geq 2mm'$, there exists $J_i \subseteq I_i$ such that $|J_i| \geq 2m > 0$ and the restriction $\psi_{|J_i}$ takes a constant value $s(i)$, for some function $s\colon [m] \to [m']$. Next, for each $i \in [m]$ and for every $0 \leq k_i < 2m$, we can fix $K_i \subset J_i$, with $|K_i| = k_i$. Then we let the tuple $\chi = \chi(K_1, \ldots, K_m) \in \{0,1\}^k$ take $\chi_{|K_i} = 1$, $i \in [m]$, and all other entries are 0. Let R be the set of all such tuples χ. Then $\hom_\varphi(G_\chi, H) = \hom_\psi(G_\chi, H')$ for every G_χ with $\chi \in R$ is expressed by: For all $0 \leq k_j < 2m$, $j \in [m]$,

$$\sum_{i=1}^{m} \alpha_i \prod_{j=1}^{m} \beta_{ij}^{k_j} = \sum_{i=1}^{m'} \alpha'_i \prod_{j=1}^{m} (\beta'_{is(j)})^{k_j}. \tag{7}$$

In (7) the sums on i come from assigning $v \in V(G_\chi)$ to $i \in V(H)$ or to $i \in V(H')$, respectively.

Because H is twin-free the m-tuples $(\beta_{ij})_{j\in[m]} \in \mathbb{F}^m$ for $1 \leq i \leq m$ are pairwise distinct. In (7) the sum in the LHS has m terms, while the sum in the RHS has $m' \leq m$ terms. Transferring the RHS to the LHS we get at most $2m$ terms. Now we apply Corollary 9 to the sum obtained by moving all terms of the RHS to the LHS in (7). By the pairwise distinctness of the m-tuples $(\beta_{ij})_{j\in[m]} \in \mathbb{F}^m$ for $1 \leq i \leq m$, and since there are only $m' \leq m$ terms from the RHS and every $\alpha_i \neq 0$, we see that each term from the LHS of (7) must be canceled by exactly one term from the RHS. And this can only occur if $m = m'$, and there is a bijective map $\sigma\colon [m] \to [m]$:

$$\alpha_i = \alpha'_{\sigma(i)} \quad \text{for } i \in [m], \qquad (\beta_{ij})_{j\in[m]} = (\beta'_{\sigma(i)s(j)})_{j\in[m]} \quad \text{for } i \in [m]. \tag{8}$$

Since H, H' are undirected graphs, we also have $\beta_{ij} = \beta_{ji} = \beta'_{\sigma(j)s(i)} = \beta'_{s(i)\sigma(j)}$ for $i, j \in [m]$.

Next we show that s is bijective. If for some $x, y \in [m]$ we have $s(x) = s(y)$, then

$$(\beta_{xj})_{j\in[m]} = (\beta'_{s(x)\sigma(j)})_{j\in[m]} = (\beta'_{s(y)\sigma(j)})_{j\in[m]} = (\beta_{yj})_{j\in[m]}.$$

Since H is twin-free we conclude that $x = y$. Thus the map $s\colon [m] \to [m]$ is injective and (since $[m]$ is finite) s is bijective. However, $\sigma\colon [m] \to [m]$ is also bijective, so it follows from (8) that the tuples $(\beta'_{ij})_{j\in[m]}$ for $i \in [m]$ are pairwise distinct, which means that H' is twin-free as well.

Next we show $\psi_{|I_i} = s(i)$ for all $i \in [m]$. If for all $i \in [m]$, we have $J_i = I_i$, then we are done. Otherwise, take any $w \in [m]$ such that J_w is a proper subset of I_w and we take any $t \in I_w \setminus J_w$. Observe that $t \notin K_i$ for all $i \in [m]$, in particular, $\chi(t) = 0$ for each $\chi \in R$.

For each $\chi \in R$, let χ_+ be the tuple obtained from χ by reassigning $\chi(t)$ (changing its t-th entry) from 0 to 1 and let R_+ be the set of all such χ_+. Then $\hom_\varphi(G_\kappa, H) = \hom_\psi(G_\kappa, H')$ for every G_κ with $\kappa \in R_+$ is expressed as (recall that we have already proved that $m' = m$)

$$\sum_{i=1}^{m} \alpha_i \beta_{iw} \prod_{j=1}^{m} \beta_{ij}^{k_j} = \sum_{i=1}^{m} \alpha'_i \beta'_{i\psi(t)} \prod_{j=1}^{m} (\beta'_{is(j)})^{k_j},$$

which can be compared to (7), and here for $\kappa \in R_+$ we have one extra edge (v, u_t) in G_κ, and $\varphi(t) = w$ since $t \in I_w$. So this holds for every $0 \le k_j < 2m$ where $j \in [m]$. Transferring the RHS to the LHS and using (8), we get

$$\sum_{i=1}^{m} (\alpha_i \beta_{iw} - \alpha'_{\sigma(i)} \beta'_{\sigma(i)\psi(t)}) \prod_{j=1}^{m} \beta_{ij}^{k_j} = 0,$$

for every $0 \le k_j < 2m$ where $j \in [m]$. Since $\alpha_i = \alpha'_{\sigma(i)} \neq 0$, and the tuples $(\beta_{ij})_{j\in[m]}$ for $1 \le i \le m$ are pairwise distinct, by Corollary 9, we get $\beta_{iw} - \beta'_{\sigma(i)\psi(t)} = 0$ for $i \in [m]$. On the other hand by (8), $\beta_{iw} = \beta'_{\sigma(i)s(w)}$ for $i \in [m]$. It follows that $\beta'_{\sigma(i)\psi(t)} = \beta'_{\sigma(i)s(w)}$ for $i \in [m]$. However, since $\sigma\colon [m] \to [m]$ is a bijection and, as shown before, H' is twin-free, this implies that $\psi(t) = s(w)$. Recall that $\psi_{|J_w} = s(w)$. This proves that on $I_w \setminus J_w$, ψ also takes the constant value $s(w)$. Thus $\psi_{|I_i} = s(i)$ for all $i \in [m]$.

Next we prove that $\beta_{ij} = \beta'_{\sigma(i)\sigma(j)}$ for all $i, j \in [m]$, i.e., σ preserves the edge weights.

For each $\lambda = (b_i)_{i\in[k]}$, $\tau = (c_i)_{i\in[k]} \in \{0,1\}^k$, we define a graph $G_{\lambda,\tau} \in \mathcal{PLG}^{\mathrm{simp}}[k]$:

$V(G_{\lambda,\tau}) = \{u_1, \dots, u_k, v, v'\}$, with each u_i labeled i. There is an edge (v, v') and, for each $i \in [k]$, there is an edge (v, u_i) if $b_i = 1$, and an edge (v', u_i) if $c_i = 1$. There are no other edges.

Let $R^2 = R \times R$. By the definition of R, every $(\lambda, \tau) \in R^2$ corresponds to a sequence of pairs (K_i, L_i), $i \in [m]$, where $K_i \subset J_i \subseteq I_i$ and $L_i \subset J_i \subseteq I_i$ such that $|K_i| = k_i, |L_i| = \ell_i$, where $0 \le k_i, \ell_i < 2m$, and $\lambda_{|K_i} = 1$, $\tau_{|L_i} = 1$, for $i \in [m]$, and all other entries are 0.

Then $\hom_\varphi(G_{\lambda,\tau}, H) = \hom_\psi(G_{\lambda,\tau}, H')$ for every $G_{\lambda,\tau}$ with $(\lambda, \tau) \in R^2$ is expressed as

$$\sum_{i,j=1}^{m} \alpha_i \alpha_j \beta_{ij} \prod_{r=1}^{m} (\beta_{ir}^{k_r} \beta_{jr}^{\ell_r}) = \sum_{i,j=1}^{m} \alpha'_i \alpha'_j \beta'_{ij} \prod_{r=1}^{m} \left((\beta'_{is(r)})^{k_r} (\beta'_{js(r)})^{\ell_r} \right).$$

This holds for all $0 \le k_r, \ell_r < 2m$ where $r \in [m]$. Transferring the RHS to the LHS and using (8), we get

$$\sum_{i,j=1}^{m} (\alpha_i \alpha_j \beta_{ij} - \alpha_i \alpha_j \beta'_{\sigma(i)\sigma(j)}) \prod_{r=1}^{m} (\beta_{ir}^{k_r} \beta_{jr}^{\ell_r}) = 0,$$

for every $0 \le k_r, \ell_r < 2m$, where $r \in [m]$. Since the tuples $(\beta_{ir}, \beta_{jr})_{r\in[m]}$ for $1 \le i, j \le m$ are pairwise distinct, and $\alpha_i \alpha_j \neq 0$, by Corollary 9,

$$\beta_{ij} = \beta'_{\sigma(i)\sigma(j)}, \quad \text{for } i, j \in [m]. \tag{9}$$

This means that the bijection $\sigma\colon [m] \to [m]$ preserves the edge weights in addition to the vertex weights by (8). Hence $\sigma\colon [m] \to [m]$ is an isomorophism of $\mathbb{F}$-weighted graphs from H to H'.

Finally, we show that $\psi = \sigma \circ \varphi$. From (8) and (9), we have $\beta'_{\sigma(i)s(j)} = \beta_{ij} = \beta'_{\sigma(i)\sigma(j)}$ for $i, j \in [m]$. As H' is twin-free and σ is bijective we get $\sigma(j) = s(j)$ for $j \in [m]$. Now let $x \in [k]$, then $x \in I_i$ for some $i \in [m]$. Therefore $\varphi(x) = i$ and so $\psi(x) = s(i) = \sigma(i) = \sigma(\varphi(x))$ confirming $\psi = \sigma \circ \varphi$. ◀

Proof of Theorem 2. Consider an arbitrary $\ell \geq k$ and let $G \in \mathcal{PLG}^{\mathrm{simp}}[\ell]$ be any ℓ-labeled graph. Let $G^* = \pi_{[k]}(G)$ be the graph obtained by unlabeling the labels not in $[k]$ from G (if $k = \ell$, then $G^* = G$). Clearly, $G^* \in \mathcal{PLG}^{\mathrm{simp}}[k]$ so $\hom_\varphi(G^*, H) = \hom_\psi(G^*, H')$. Expanding the sums on the LHS and the RHS of this equality representing the maps φ, ψ along the vertices formerly labeled by $[\ell] \setminus [k]$, and then regrouping the terms corresponding to the same extension maps of φ from $[\ell]$ to $[m]$ and of ψ from $[\ell]$ to $[m']$, respectively, and then bringing back the labels from $[\ell] \setminus [k]$, we get

$$\sum_{\substack{\mu\colon [\ell]\to[m] \\ \mu \text{ extends } \varphi}} \left(\prod_{k<i\leq\ell} \alpha_{\mu(i)} \right) \hom_\mu(G, H) = \sum_{\substack{\nu\colon [\ell]\to[m'] \\ \nu \text{ extends } \psi}} \left(\prod_{k<i\leq\ell} \alpha'_{\nu(i)} \right) \hom_\nu(G, H') \tag{10}$$

for every $G \in \mathcal{PLG}^{\mathrm{simp}}[\ell]$. (Here if $\ell = k$, the empty product $\prod_{k<i\leq\ell}$ is 1.)

Now choose $\ell \geq k$ so that we can extend φ to a map $\eta\colon [\ell] \to [m]$ such that $|\eta^{-1}(u)| \geq 2m^2$ for every $u \in V(H)$. Clearly $\ell \leq k + 2m^3$ suffices. (If φ already satisfies the property, we can take $\ell = k$ and $\eta = \varphi$.) We fix ℓ and η to be such. Define

$$I = \{\mu\colon [\ell] \to [m] \mid \big(\mu_{|[k]} = \varphi\big) \wedge \big((\exists \sigma \in \mathrm{Aut}(H))\ \mu = \sigma \circ \eta\big)\},$$
$$J = \{\nu\colon [\ell] \to [m'] \mid \big(\nu_{|[k]} = \psi\big) \wedge \big((\exists \sigma \in \mathrm{Isom}(H, H'))\ \nu = \sigma \circ \eta\big)\}.$$

Obviously, $\eta \in I$ so $I \neq \emptyset$. For now, we do not exclude the possibility $J = \emptyset$ but our goal is to show that this is not the case. If $\mu\colon [\ell] \to V(H)$ extends φ but $\mu \notin I$, then by Lemma 10, there exists a graph $G_{\eta,\mu} \in \mathcal{PLG}^{\mathrm{simp}}[\ell]$ such that $\hom_\eta(G_{\eta,\mu}, H) \neq \hom_\mu(G_{\eta,\mu}, H)$. Similarly, if $\nu\colon [\ell] \to V(H')$ extends ψ but $\nu \notin J$, then by Lemma 10, there exists a graph $G'_{\eta,\nu} \in \mathcal{PLG}^{\mathrm{simp}}[\ell]$ such that $\hom_\eta(G'_{\eta,\nu}, H) \neq \hom_\nu(G'_{\eta,\nu}, H')$. Let S be the set consisting of these graphs $G_{\eta,\mu}$ and $G'_{\eta,\nu}$ (we remove any repetitions). Note that if $\mu' \in I$, then $\hom_\eta(G, H) = \hom_{\mu'}(G, H)$ for any $G \in \mathcal{PLG}[\ell]$, and if $\nu' \in J$, then $\hom_\eta(G, H) = \hom_{\nu'}(G, H')$ for any $G \in \mathcal{PLG}[\ell]$. In particular, both equalities hold for each $G \in S$. We impose a linear order on S and regard any set indexed by S as a tuple. For any tuple $\bar{h} = (h_G)_{G\in S}$ of integers, where each $0 \leq h_G < 2m^\ell$, consider the graph $G_{\bar{h}} = \prod_{G\in S} G^{h_G} \in \mathcal{PLG}^{\mathrm{simp}}[\ell]$. Substituting $G = G_{\bar{h}}$ in (10) and using the multiplicativity of partial graph homomorphisms (5), we obtain

$$\sum_{\substack{\mu\colon [\ell]\to[m] \\ \mu \text{ extends } \varphi}} \left(\prod_{k<i\leq\ell} \alpha_{\mu(i)} \right) \prod_{G\in S} (\hom_\mu(G, H))^{h_G} = \sum_{\substack{\nu\colon [\ell]\to[m'] \\ \nu \text{ extends } \psi}} \left(\prod_{k<i\leq\ell} \alpha'_{\nu(i)} \right) \prod_{G\in S} (\hom_\nu(G, H'))^{h_G}$$

for every $0 \leq h_G < 2m^\ell$. By the previous observations and the fact that S contains each $G_{\eta,\mu}$ and $G'_{\eta,\nu}$, the tuple $(\hom_\eta(G, H))_{G\in S}$ coincides with the tuple $(\hom_\mu(G, H))_{G\in S}$ for $\mu \in I$ and it also coincides with the tuple $(\hom_\nu(G, H'))_{G\in S}$ for $\nu \in J$; on the other hand, this tuple $(\hom_\eta(G, H))_{G\in S}$ is different from the tuple $(\hom_\mu(G, H))_{G\in S}$ for each $\mu\colon [k] \to V(H)$ extending φ and not in I and it is also different from the tuple $(\hom_\nu(G, H'))_{G\in S}$ for each $\nu\colon [k] \to V(H')$ extending ψ and not in J. Transferring the RHS to the LHS and then applying Corollary 9, we conclude that

$$\sum_{\mu\in I} \left(\prod_{k<i\leq\ell} \alpha_{\mu(i)} \right) = \sum_{\nu\in J} \left(\prod_{k<i\leq\ell} \alpha'_{\nu(i)} \right). \tag{11}$$

If $\mu \in I$, then $\mu = \sigma \circ \eta$ for some $\sigma \in \mathrm{Aut}(H)$, and therefore $\alpha_{\mu(i)} = \alpha_{\sigma(\eta(i))} = \alpha_{\eta(i)}$ for $1 \le i \le \ell$. Hence $\prod_{k<i\le\ell} \alpha_{\mu(i)} = \prod_{k<i\le\ell} \alpha_{\eta(i)}$ (if $k = \ell$, then both sides are 1), so (11) transforms to

$$|I|_{\mathbb{F}} \cdot \prod_{k<i\le\ell} \alpha_{\eta(i)} = \sum_{\nu\in J} \left(\prod_{k<i\le\ell} \alpha'_{\nu(i)} \right). \tag{12}$$

Here we let$|I|_{\mathbb{F}} = |I| \cdot 1_{\mathbb{F}} = 1_{\mathbb{F}} + \ldots + 1_{\mathbb{F}} \in \mathbb{F}$ ($1_{\mathbb{F}}$ occurs $|I|$ times). Since all $\alpha_i \neq 0$, we have $\prod_{k<i\le\ell} \alpha_{\eta(i)} \neq 0$ (if $k = \ell$, this product is $1_{\mathbb{F}} \neq 0$). Because $I \neq \emptyset$ (as $\eta \in I$) we have $|I| \ge 1$; but $\mathrm{char}\,\mathbb{F} = 0$ so $|I|_{\mathbb{F}} \neq 0$ and therefore $|I|_{\mathbb{F}} \cdot \prod_{k<i\le\ell} \alpha_{\eta(i)} \neq 0$. This implies that the RHS of (12) is nonzero as well which can only occur when $J \neq \emptyset$. Take $\xi \in J$. Then $\xi = \sigma \circ \eta$ for some isomorphism of $\mathbb{F}$-weighted graphs $\sigma\colon V(H) \to V(H')$ from H to H'. Restricting to $[k]$, we obtain $\psi = \sigma \circ \varphi$ which completes the proof. ◀

▶ Remark 11. Theorem 2 shows that the condition $\hom_\varphi(G, H) = \hom_\psi(G, H')$ for all $G \in \mathcal{PLG}^{\mathrm{simp}}[k]$ is equivalent to the existence of an isomorphism from H to H' such that $\psi = \sigma \circ \varphi$. This condition is effectively checkable. However, for the purpose of effectively producing a #P-hardness reduction in the dichotomy theorems, e.g., in [4], we need a witness G such that $\hom_\varphi(G, H) \neq \hom_\psi(G, H')$.

The proof of Theorem 2 gives an *explicit finite* list to check. For $\ell = k + 2m^3$, let $\mathcal{P}_{\ell,m} = \{\prod_{G\in S} G^{h_G} \mid 0 \le h_G < 2m^\ell\}$, where S is from the proof of Theorem 2 using the construction of Lemma 10. We then define $\mathcal{Q}_{k,m} = \pi_{[k]}(\mathcal{P}_{\ell,m}) \subseteq \mathcal{PLG}^{\mathrm{simp}}[k]$. Then the proof of Theorem 2 shows that: the existence of an isomorphism $\sigma\colon V(H) \to V(H')$ from H to H' such that $\varphi = \sigma \circ \psi$, is equivalent to $\hom_\varphi(G, H) = \hom_\psi(G, H')$ for every $G \in \mathcal{Q}_{k,m}$.

▶ Remark 12. In their proof of Theorem 2.1 in [11], Goodall, Regts and Vena noted in a footnote on p. 268 that "Even though Lovász [9] (this is [15]) works over $\mathbb{R}$ and assumes $a_i > 0$ for all i, it is easy to check that the proof of his Lemma 2.4 remains valid in our setting." The following is a sketch of this approach due to Goodall, Regts and Vena; we thank Guus Regts for enlightening discussions. Notations below can be found in [15].

Following the proof of Lemma 2.4 by Lovász in [15] one comes to Claim 4.2. On p. 968 the set Ψ is defined as consisting of all maps equivalent to μ. This set appears in the sum on line 2 of p. 969. Let Ψ_0 be the subset of all maps $\eta \in \Psi$ that restrict to ϕ. Since $\mu \in \Psi$ and μ restricts to ϕ, the subset Ψ_0 is nonempty; however it could have cardinality > 1. Then in the sum on line 2 of p. 969 when one collects all terms with the same restriction, ϕ appears with the coefficient $\sum_{\eta\in\Psi_0} \alpha(\eta(k+1))$. The crucial step in the proof of Claim 4.2 is that this sum is nonzero. In [15] this is nonzero since all vertex weights are positive. Without this positivity, it is possible that such a sum is 0. Thus one cannot directly follow the proof in [15] in the general case when a nonempty subset of vertex weights can sum to 0.

However, instead of Claim 4.2 in [15] on line 6 of p. 970 after the proof of Claim 4.4, one can extend ϕ to a surjective map $\mu : [\ell] \to [m]$ for some $\ell \ge k$. Then apply the trace operator of [15] $\ell - k$ times to an analogous sum defined on line -1 of p. 968, which is in $\mathcal{A}''_\ell$. This gives a sum $\Sigma = \sum_{\eta\in\Psi} \prod_{i\in[k+1:\ell]} \alpha(\eta(i))\eta_{\mathrm{res}}$, where η_{res} is the restriction of η to $[k]$.

Then use Claim 4.4, all $\alpha(\eta(i))$ are the same for $\eta \in \Psi$ and thus $P = \prod_{i\in[k+1:\ell]} \alpha(\eta(i))$ is a common factor. Since each $\alpha(\eta(i)) \neq 0$, this $P \neq 0$. Since $\mu \in \Psi$, $\phi = \mu_{\mathrm{res}}$ appears as a term in Σ with a coefficient $|S|P$, where S is the subset of Ψ consisting of maps that restrict to ϕ. As $\mathbb{F}$ has characteristic 0, this $|S|P \neq 0$.

Now since ψ is given as equivalent to ϕ, and the sum Σ is in $\mathcal{A}''_k$, ψ must have the same coefficient as that of ϕ, which has the form $|S'|P$, where S' is the subset of Ψ that restrict to ψ. So $|S'| = |S|$, in particular S' is nonempty. Thus there is some $\eta \in \Psi$ such that η extends ψ, and η is equivalent to μ.

This leads to the following theorem:

▶ Theorem 13. *Let $\mathbb{F}$ be a field of characterisitic 0. Let H be an undirected $\mathbb{F}$-weighted twin-free graph. For any $k \geq 0$, if $\varphi, \psi \colon [k] \to V(H)$ and $\hom_\phi(G, H) = \hom_\psi(G, H')$ for all undirected graphs G, where G may have multiloops and multiedges, then there is an automorphism σ of H such that $\phi = \sigma \circ \psi$.*

6 Effective GH Dichotomies

We briefly discuss how to use Theorem 2 to make complexity dichotomies for graph homomorphism effective. A long and fruitful sequence of work [7, 8, 2, 12, 21, 10] led to the following complexity dichotomy for weighted graph homomorphisms [4] which unifies these previous ones: There is a tractability condition $\mathcal{P}$ such thatfor any (algebraic) complex symmetric matrix H, if H satisfies $\mathcal{P}$ then $\hom(\cdot, H)$ is P-time computable, otherwise there is a P-time reduction from a canonical #P-hard problem to $\hom(\cdot, H)$. However, in the long sequence of reductions in [4] there are nonconstructive steps, a prominent example is the first pinning lemma (Lemma 4.1, p. 937). This involves condensing H by collapsing "equivalent" vertices, while introducing vertex weights. Consider all 1-labeled graphs G. We say two vertices $u, v \in V(H)$ are "equivalent" if $\hom_u(G, H) = \hom_v(G, H)$ where the notation $\hom_u(G, H)$ denotes the partial sum of $\hom(G, H)$ where we restrict to all mappings which map the labeled vertex of G to u, and similarly for $\hom_v(G, H)$. This is just the special case $k = 1$ in Theorem 2 (note that we first apply the twin compression step to H). Previously the P-time reduction was proved existentially. Using Theorem 2 (see Remark 11 after the proof), this step can be made effective.

There is a finer distinction between making the dichotomy effective in the sense discussed here versus the decidability of the dichotomy. The dichotomy criterion in [4] is decidable in P-time (measured in the size of the description of H); however, according to the proof in [4] which involves nonconstructive steps, when the decision algorithm decides $\hom(\cdot, H)$ is #P-hard it does not produce a reduction. The results in this paper can constructively produce such a reduction when $\hom(\cdot, H)$ is #P-hard.

Previous versions of Theorem 2 (e.g., [14, 15]) show that the above equivalence on u, v for suitably restricted classes of H can be decided by testing for graph isomorphism (with pinning). However, to actually obtain the promised P-time reduction one has to search for "witness" graphs G to $\hom_u(G, H) \neq \hom_v(G, H)$. Having no graph isomorphism mapping u to v does not readily yield such a "witness" graph G, although an open ended search is guaranteed to find one. Thus Theorem 2 gives a double exponential time (in the size of H) algorithm to find a reduction algorithm, while directly applying previous versions of the theorem gives a computable process with no definite time bound. (But we emphasize that no previous versions of Theorem 2 apply to the dichotomy in [4].)

7 Counterexample for fields of finite characteristic

In Lemma 10, the field $\mathbb{F}$ is arbitrary. By contrast, for Theorem 2 the proof uses the assumption that $\operatorname{char} \mathbb{F} = 0$. We show that this assumption cannot be removed, for any fixed k, by an explicit counterexample. The counterexample also applies to Corollaries 3 to 6.

Let $\operatorname{char} \mathbb{F} = p > 0$. For $n \geq 2$ and $\ell_1 > \ldots > \ell_n > 0$, define an (undirected) $\mathbb{F}$-weighted graph $H = H_{n,\ell_1,\ldots,\ell_n}$ with the vertex set $U \cup \bigcup_{i=1}^n V_i$ where $U = \{u_1, \ldots, u_n\}$ and $V_i = \{v_{i,j} \mid 1 \leq j \leq \ell_i p\}$, for $i \in [n]$, and the edge set being the union of the edge sets that form a copy of the complete graph K_n on U and $K_{1+\ell_i p}$ on $\{u_i\} \cup V_i$ for $i \in [n]$. H

is a simple graph with no loops. To make H an $\mathbb{F}$-weighted graph, we assign each vertex and edge weight 1. (So H is really unweighted.) It is easy to see that H is twin-free: First, any two distinct vertices from U or from the same V_i are not twins because H is loopless. (Note that for vertices i, j to be twin in an undirected graph, if (i, j) is an edge, then the loops (i, i), (j, j) must also exist.) Second, for any $i \in [n]$, $u_i \in U$ and any $v \in V_i$ are not twins by $\deg(u_i) > \deg(v)$. Third, $u_i \in U$ (or any $v \in V_i$) and any $w \in V_j$, for $j \neq i$, are not twins because w has some neighbor in V_j while u_i (or v) do not. Let $\sigma \in \mathrm{Aut}(H)$ be an automorphism of H. Each vertex $u \in U$ has the property that u has two neighbors (one in U and one not in U) such that they are not neighbors to each other. This property separates U from the rest. Furthermore $\deg(u_1) > \ldots > \deg(u_n)$. Therefore σ must fix U pointwise. Then it is easy to see that σ must permute each V_i.

For any $\varphi\colon [k] \to U \subset V(H)$ where $k \geq 0$, we claim that $\hom_\varphi(G, H) = \hom_\varphi(G, K_U)$ for every $G \in \mathcal{PLG}[k]$, where K_U is the complete graph with the vertex set U.

Let $\mathfrak{S}_N$ denote the symmetry group on N letters. We define a group action of $\prod_{i=1}^n \mathfrak{S}_{\ell_i p}$ on $\{\xi \mid \xi\colon V(G) \to V(H)\}$ which permutes the images of ξ within each of $V_1, \ldots, V_n$, and fixes U pointwise. Thus for $g = (g_1, \ldots, g_n) \in \prod_{i=1}^n \mathfrak{S}_{\ell_i p}$, if $\xi(w) \in V_i$, then $\xi^g(w) = g_i(\xi(w))$. This group action partitions all ξ into orbits. Consider any $\xi\colon V(G) \to V(H)$ extending φ, such that the image $\xi(V(G)) \not\subseteq U$. Let η be in the same orbit of ξ. The nonzero contributions to $\hom_\xi(G, H)$ and $\hom_\eta(G, H)$ come from either edge weights within U, where they are identical, or within each $\{u_i\} \cup V_i$. Hence by the definition of the group action, $\hom_\xi(G, H) = \hom_\eta(G, H)$. The stabilizer of ξ consists of those g such that each g_i fixes the image set $\xi(V(G)) \cap V_i$ pointwise. Since $\xi(V(G)) \not\subseteq U$, the orbit has cardinality, which is the index of the stablizer, divisible by some $\ell_i p$. In particular it is 0 mod p. Thus the total contribution from each orbit is zero in $\mathbb{F}$, except for those ξ with $\xi(V(G)) \subseteq U$. The claim follows.

For $k \geq 1$, we take $H' = H$. We say that maps $\varphi, \psi\colon [k] \to U$ have the *same type* if for every $i, j \in [k]$, $\varphi(i) = \varphi(j)$ iff $\psi(i) = \psi(j)$. Thus the inverse image sets $\varphi^{-1}(\varphi(i))$ and $\psi^{-1}(\psi(i))$ have the same cardinality for every $i \in [k]$. It follows that the image sets $\varphi([k])$ and $\psi([k])$, consisting of the elements of U having a nonempty inverse image sets under φ^{-1} and ψ^{-1} respectively, are of the same cardinality. So there is a bijection σ of U, mapping $U \setminus \varphi([k])$ to $U \setminus \psi([k])$, and also for every $i \in [k]$, $(\sigma \circ \varphi)(i) = \psi(i)$. Thus $\sigma \in \mathrm{Aut}(K_U)$ and $\sigma \circ \varphi = \psi$. Take $\varphi, \psi\colon [k] \to U$ such that $\varphi \neq \psi$, and they have the same type. For example, we can take $\varphi(i) = u_1$ and $\psi(i) = u_2$ for every $i \in [k]$. Since φ and ψ have the same type, clearly $\hom_\varphi(G, K_U) = \hom_\psi(G, K_U)$. For every $G \in \mathcal{PLG}[k]$, we have already shown that $\hom_\varphi(G, K_U) = \hom_\varphi(G, H)$, and since $H' = H$, $\hom_\psi(G, K_U) = \hom_\psi(G, H')$, implying that $\hom_\varphi(G, H) = \hom_\psi(G, H')$. If Theorem 2 were to hold for the field $\mathbb{F}$ of $\mathrm{char}\,\mathbb{F} = p > 0$, there would be an automorphism $\widehat{\sigma} \in \mathrm{Aut}(H)$ such that $\widehat{\sigma} \circ \varphi = \psi$. But every automorphism of H must fix U pointwise, and thus it restricts to the identity map on U. And since $\varphi([k]) \subseteq U$, we have $\widehat{\sigma} \circ \varphi = \varphi \neq \psi$, a contradiction.

When $k = 0$, in addition to $H = H_{n,\ell_1,\ldots,\ell_n}$ we also take $H' = H_{n,\ell'_1,\ldots,\ell'_n}$ on the vertex set $U \cup \bigcup_{i=1}^n V'_i$, where $n \geq 2$, $\ell'_1 > \ldots > \ell'_n > 0$ and $(\ell'_1, \ldots, \ell'_n) \neq (\ell_1, \ldots, \ell_n)$. As $k = 0$, the only possible choices are the empty maps $\varphi = \emptyset$ and $\psi = \emptyset$, and $\hom(G, H) = \hom(G, K_U) = \hom(G, H')$ still holds for every $G \in \mathcal{PLG}[0]$. However, the same property that every vertex $u \in U$ has two neighbors such that they are not neighbors to each other separates U from the rest in both H and H'. Then the monotonicity $\deg(u_1) > \ldots > \deg(u_n)$ within both H and H' shows that any isomorphism from H to H', if it exists, must fix U pointwise. Then it is easy to see that σ must be a bijection from V_i of H to the corresponding copy V'_i in H'. This forces $(\ell_1, \ldots, \ell_n) = (\ell'_1, \ldots, \ell'_n)$, a contradiction.

8 Rank of Connection Tensors and Dimension of Graph Algebras

The purpose of this section is to summarize the extensions of the main results from [15]. These are stated as Theorems 14 and 15.

An $\mathbb{F}$-valued *graph parameter* is a function from finite graph ismorphism classes to $\mathbb{F}$. For convinience, we think of a graph parameter as a function defined on finite graphs and invariant under graph isomorphism. We allow multiple edges in our graphs, but no loops, as input to a graph parameter. A graph parameter f is called *multiplicative*, if for any disjoint union $G_1 \sqcup G_2$ of graphs G_1 and G_2 we have $f(G_1 \sqcup G_2) = f(G_1)f(G_2)$. A graph parameter on a labeled graph ignores its labels. Every weighted graph homomorphism $f_H = \hom(\cdot, H)$ is a multiplicative graph parameter.

A (k-labeled, $\mathbb{F}$-)*quantum graph* is a finite formal $\mathbb{F}$-linear combination of finite k-labeled graphs. $\mathcal{G}[k] = \mathbb{F}\mathcal{PLG}[k]$ is the monoid algebra of k-labeled $\mathbb{F}$-quantum graphs. We denote by $\mathcal{G}^{\text{simp}}[k]$ the monoid algebra of *simple* k-labeled $\mathbb{F}$-quantum graphs; it is a subalgebra of $\mathcal{G}[k]$. U_k is the multiplicative identity and the empty sum is the additive identity in both $\mathcal{G}[k]$ and $\mathcal{G}^{\text{simp}}[k]$.

Let f be any graph parameter. For all integers $k, n \geq 0$, we define the following n-dimensional array $T(f, k, n) \in \mathbb{F}^{(\mathcal{PLG}[k])^n}$, which can be identified with $(V^{\otimes n})^*$, the dual space of $V^{\otimes n}$, where $V = \bigoplus_{\mathcal{PLG}[k]} \mathbb{F}$ is the infinite dimensional vector space with coordinates indexed by $\mathcal{PLG}[k]$. The entry of $T(f, k, n)$ at coordinate $(G_1, \ldots, G_n)$ is $f(G_1 \cdots G_n)$; when $n = 0$, we define $T(f, k, n)$ to be the scalar $f(U_k)$. The arrays $T(f, k, n)$ are symmetric with respect to its coordinates, i.e., $T(f, k, n) \in \text{Sym}(\mathbb{F}^{(\mathcal{PLG}[k])^n})$. Fix f, k and n, we call the n-dimensional array $T(f, k, n)$ the (k-th, n-dimensional) *connection tensor* of the graph parameter f. When $n = 2$, a connection tensor is exactly a *connection matrix* of the graph parameter f studied in [9], i.e., $T(f, k, 2) = M(f, k)$.

For graph parameters of the form $f_H = \hom(\cdot, H)$, where H has positive vertex weights and real edge weights, the main results of [15] are to compute the rank of the corresponding connection matrices, and the dimension of graph algebras, etc. We will prove these results for arbitrary $\mathbb{F}$-weighted graphs (without vertex or edge weight restrictions). Moreover we will prove these for connection tensors (see [5]). Below we let H be a (directed or undirected) $\mathbb{F}$-weighted graph.

For $k \geq 0$, let $N(k, H)$ be the matrix whose rows are indexed by maps $\varphi\colon [k] \to V(H)$ and columns are indexed by $\mathcal{PLG}[k]$, and the row indexed by φ is $\hom_\varphi(\cdot, H)$. We have a group action of $\text{Aut}(H)$ on the k-tuples from $V(H)^k = \{\varphi\colon [k] \to V(H)\}$ by $\varphi \mapsto \sigma \circ \varphi$ for $\sigma \in \text{Aut}(H)$ and $\varphi\colon [k] \to V(H)$. We use $\text{orb}_k(H)$ to denote the number of its orbits.

As mentioned before, $f_H = \hom(\cdot, H)$ ignores labels on a labeled graph, so we can think of f_H as defined on $\mathcal{PLG}[k]$ and then by linearity as defined on $\mathcal{G}[k]$. Then we can define the following bilinear symmetric form on $\mathcal{G}[k]$:

$$\langle x, y \rangle = f_H(xy), \quad x, y \in \mathcal{G}[k].$$

Let

$$\mathcal{K}_{[k]} = \{x \in \mathcal{G}[k]\colon f_H(xy) = \langle x, y \rangle = 0 \ \forall y \in \mathcal{G}[k]\}.$$

Clearly, $\mathcal{K}_{[k]}$ is an ideal in $\mathcal{G}[k]$, so we can form a quotient algebra $\mathcal{G}'[k] = \mathcal{G}[k]/\mathcal{K}_{[k]}$. It is easy to see that $h \in \mathcal{K}_{[k]}$ iff $M(f_H, k)h = 0$.

In order to be consistent with the notation in [15], when $f = f_H = \hom(\cdot, H)$ for an $\mathbb{F}$-weighted (directed or undirected) graph H, we let $T(k, n, H) = T(f_H, k, n)$ where $k, n \geq 0$ and $M(k, H) = M(f_H, k)$ where $k \geq 0$.

The following theorems extend Theorem 2.2, Corollary 2.3 and the results of Section 3 in [15].

▶ **Theorem 14.** *Let $\mathbb{F}$ be a field of characteristic 0. Let H be a (directed or undirected) $\mathbb{F}$-weighted twin-free graph. Then $\mathcal{G}'[k] \cong \mathbb{F}^{r_k}$ as isomorphic algebras, where*

$$r_k = \dim \mathcal{G}'[k] = \mathrm{rk}_{\mathsf{S}}\, T(k,n,H) = \mathrm{rk}\, T(k,n,H) = \mathrm{rk}\, M(k,H) = \mathrm{rk}\, N(k,H) = \mathrm{orb}_k(H)$$

for $k \geq 0$ and $n \geq 2$. Here rk_{S} denotes symmetric tensor rank, and rk on T and on M, N denote tensor and matrix rank, respectively. In particular, if H has no nontrivial $\mathbb{F}$-weighted automorphisms, then the above quantities are all equal to $|V(H)|^k$.

The following theorem generalizes Lemma 2.5 in [15]. Let $k \geq 0$ be an integer and H be an $\mathbb{F}$-weighted graph. We say that a vector $f\colon V(H)^k \to \mathbb{F}$ is *invariant under the automorphisms* of H if $f(\sigma \circ \varphi) = f(\varphi)$ for every $\sigma \in \mathrm{Aut}(H)$ and $\varphi \in V(H)^k$.

▶ **Theorem 15.** *Let $\mathbb{F}$ be a field of characteristic 0. Let H be a (directed or undirected) $\mathbb{F}$-weighted twin-free graph. Then for $k \geq 0$, the column space of $N(k,H)$ consists of precisely those vectors $f\colon V(H)^k \to \mathbb{F}$ that are invariant under the authomorphisms of H. Moreover, every such vector can be obtained as a finite linear combination of the columns of $N(k,H)$ indexed by $\mathcal{PLG}^{\mathrm{simp}}[k]$.*

From this theorem, we can immediately conclude the existence of simple contractors and connectors from $\mathcal{PLG}^{\mathrm{simp}}[k]$ when $\mathrm{char}\,\mathbb{F} = 0$ for GH functions (see [17] for the definitions).

References

1 Christian Borgs, Jennifer T. Chayes, László Lovász, Vera T. Sós, and Katalin Vesztergombi. Convergent sequences of dense graphs I: Subgraph frequencies, metric properties and testing. *Advances in Mathematics*, 219(6):1801–1851, 2008.

2 Andrei Bulatov and Martin Grohe. The complexity of partition functions. *Theoretical Computer Science*, 348(2-3):148–186, 2005.

3 Jin-Yi Cai and Xi Chen. A decidable dichotomy theorem on directed graph homomorphisms with non-negative weights. *Computational Complexity*, 28(3):345–408, 2019.

4 Jin-Yi Cai, Xi Chen, and Pinyan Lu. Graph Homomorphisms with Complex Values: A Dichotomy Theorem. *SIAM Journal on Computing*, 42(3):924–1029, 2013.

5 Jin-Yi Cai and Artem Govorov. Perfect Matchings, Rank of Connection Tensors and Graph Homomorphisms. In *Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 476–495, 2019. Full version available at `arXiv:1909.03179`.

6 Martin E. Dyer, Leslie A. Goldberg, and Mike Paterson. On counting homomorphisms to directed acyclic graphs. *Journal of the ACM*, 54(6):27, 2007.

7 Martin E. Dyer and Catherine S. Greenhill. The complexity of counting graph homomorphisms. *Random Structures and Algorithms*, 17(3-4):260–289, 2000.

8 Martin E. Dyer and Catherine S. Greenhill. Corrigendum: The complexity of counting graph homomorphisms. *Random Structures and Algorithms*, 25(3):346–352, 2004.

9 Michael Freedman, László Lovász, and Alexander Schrijver. Reflection positivity, rank connectivity, and homomorphism of graphs. *Journal of the American Mathematical Society*, 20(1):37–51, 2007.

10 Leslie A. Goldberg, Martin Grohe, Mark Jerrum, and Marc Thurley. A Complexity Dichotomy for Partition Functions with Mixed Signs. *SIAM Journal on Computing*, 39(7):3336–3402, 2010.

11 Andrew Goodall, Guus Regts, and Lluís Vena. Matroid invariants and counting graph homomorphisms. *Linear Algebra and its Applications*, 494:263–273, 2016.

12 Martin Grohe and Marc Thurley. Counting Homomorphisms and Partition Functions. In *Model Theoretic Methods in Finite Combinatorics*, volume 558 of *Contemporary Mathematics*, pages 243–292. American Mathematical Society, 2011.

13 Pavol Hell and Jaroslav Nešetřil. *Graphs and homomorphisms*, volume 28 of *Oxford lecture series in mathematics and its applications*. Oxford University Press, 2004.

14 László Lovász. Operations with structures. *Acta Mathematica Hungarica*, 18(3-4):321–328, 1967.

15 László Lovász. The rank of connection matrices and the dimension of graph algebras. *European Journal of Combinatorics*, 27(6):962–970, 2006.

16 László Lovász and Vera T. Sós. Generalized quasirandom graphs. *Journal of Combinatorial Theory, Series B*, 98(1):146–163, 2008.

17 László Lovász and Balázs Szegedy. Contractors and connectors of graph algebras. *Journal of Graph Theory*, 60(1):11–30, 2009.

18 Guus Regts. *Graph Parameters and Invariants of the Orthogonal Group*. PhD thesis, Universiteit van Amsterdam, 2013.

19 Alexander Schrijver. Graph invariants in the spin model. *Journal of Combinatorial Theory, Series B*, 99(2):502–511, 2009.

20 Balázs Szegedy. Edge coloring models and reflection positivity. *Journal of the American Mathematical Society*, 20(4):969–988, 2007.

21 Marc Thurley. *The Complexity of Partition Functions*. PhD thesis, Humboldt Universität zu Berlin, 2009.

Tarski's Theorem, Supermodular Games, and the Complexity of Equilibria

Kousha Etessami
School of Informatics, University of Edinburgh, UK
kousha@inf.ed.ac.uk

Christos Papadimitriou
Dept. of Computer Science, Columbia University, NY, USA
christos@cs.columbia.edu

Aviad Rubinstein
Dept. of Computer Science, Stanford University, CA, USA
aviad@cs.stanford.edu

Mihalis Yannakakis
Dept of Computer Science, Columbia University, NY, USA
mihalis@cs.columbia.edu

Abstract

The use of monotonicity and Tarski's theorem in existence proofs of equilibria is very widespread in economics, while Tarski's theorem is also often used for similar purposes in the context of verification. However, there has been relatively little in the way of analysis of the complexity of finding the fixed points and equilibria guaranteed by this result. We study a computational formalism based on monotone functions on the d-dimensional grid with sides of length N, and their fixed points, as well as the closely connected subject of supermodular games and their equilibria. It is known that finding some (any) fixed point of a monotone function can be done in time $\log^d N$, and we show it requires at least $\log^2 N$ function evaluations already on the 2-dimensional grid, even for randomized algorithms. We show that the general Tarski problem of finding some fixed point, when the monotone function is given succinctly (by a boolean circuit), is in the class PLS of problems solvable by local search and, rather surprisingly, also in the class PPAD. Finding the greatest or least fixed point guaranteed by Tarski's theorem, however, requires $d \cdot N$ steps, and is NP-hard in the white box model. For supermodular games, we show that finding an equilibrium in such games is essentially computationally equivalent to the Tarski problem, and finding the maximum or minimum equilibrium is similarly harder. Interestingly, two-player supermodular games where the strategy space of one player is one-dimensional can be solved in $O(\log N)$ steps. We also show that computing (approximating) the value of Condon's (Shapley's) stochastic games reduces to the Tarski problem. An important open problem highlighted by this work is proving a $\Omega(\log^d N)$ lower bound for small fixed dimension $d \geq 3$.

2012 ACM Subject Classification Theory of computation → Computational complexity and cryptography

Keywords and phrases Tarski's theorem, supermodular games, monotone functions, lattices, fixed points, Nash equilibria, computational complexity, PLS, PPAD, stochastic games, oracle model, lower bounds

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.18

Related Version A full version [11] is available at: `https://arxiv.org/abs/1909.03210`.

Funding This work was partially supported by NSF Grants CCF-1703295, CCF-1763970.

Acknowledgements Thanks to Alexandros Hollender for pointing us to [2].

1 Introduction

Equilibria are paramount in economics, because guaranteeing their existence in a particular strategic or market-like framework enables one to consider "What happens at equilibrium?" without further analysis. Equilibrium existence theorems are nontrivial to prove. The best known example is Nash's theorem [18], whose proof in 1950, based on Brouwer's fixed point theorem, transformed game theory, and inspired the Arrow-Debreu price equilibrium results [1], among many others. Decades later, complexity analysis of these theorems and corresponding solution concepts by computer scientists has created a fertile and powerful field of research [19].

Not all equilibrium theorems in economics, however, rely on Brouwer's fixed point theorem for their proof (even though, in a specific sense made clear and proved in this paper, they could have...). Many of the exceptions ultimately rely on *Tarski's fixed point theorem* [22], stating that all monotone functions on a complete lattice have a fixed point – and in fact a whole sublattice of fixed points with a largest and smallest element [23, 17, 24]. In contrast to the equilibrium theorems whose proof relies on Brouwer's fixed point theorem, there has been relatively little complexity analysis of Tarski's fixed point theorem and the equilibrium results it enables. (We discuss prior related work at the end of this introduction.)

Here we present several results in this direction. Let $[N] = \{1, \ldots, N\}$. To formulate the basic problem, we consider a monotone function f on the d-dimensional grid $[N]^d$, that is, a function $f : [N]^d \mapsto [N]^d$ such that for all $x, y \in [N]^d$, $x \geq y$ implies $f(x) \geq f(y)$; in the black-box oracle model, we can query this function with specific vectors $x \in [N]^d$; in the white-box model we assume that the function is presented by a boolean circuit[1]. Thus, d and N are the basic parameters to our model; it is useful to think of d as the dimensionality of the problem, while N is something akin to the inverse of the desired approximation ϵ.

- Tarski's theorem in the grid framework is easy to prove. Let $\bar{1} = (1, \ldots, 1)$ denote the (d-dimensional) all-1 vector. Consider the sequence of grid points $\bar{1}, f(\bar{1}), f(f(\bar{1})), \ldots, f^i(\bar{1}), \ldots$. From monotonicity of f, by induction on i we get, for all $i \geq 0$, $f^i(\bar{1}) \leq f^{i+1}(\bar{1})$. Unless a fixed point is arrived at, the sum of the coordinates must increase at each iteration. Therefore, after at most dN iterations of f applied to $\bar{1}$, a fixed point is found. In other words $f^{dN}(\bar{1}) = f^{dN+1}(\bar{1})$.
- This immediately suggests an $O(dN)$ algorithm. But an $O(\log^d N)$ algorithm is also known[2]: Consider the $(d-1)$-dimensional function obtained by fixing the "input value" in the d'th coordinate of the function f with some value r_d (initialize $r_d := \lceil N/2 \rceil$). Find a fixed point $z^* \in [N]^{d-1}$ of this $(d-1)$-dimensional monotone function $f(z, r_d)$ (recursively). If the dth coordinate $f_d(z^*, r_d)$ of $f(z^*, r_d)$, is equal to r_d, then (z^*, r_d) is a fixed point of the overall function f, and we are done. Otherwise, a binary search on the d'th coordinate is enabled: we need to look for a larger (smaller) value of r_d if $f_d(z^*, r_d) > r_d$ (respectively, if $f_d(z^*, r_d) < r_d$). By an easy induction, this establishes the $O(\log^d N)$ upper bound ([5]).
- We conjecture that this algorithm is essentially optimal in the black box sense, for small fixed constant dimension d. In Theorem 7 we prove this result for the $d = 2$ case. We provide a class of monotone functions that we call *the herringbones*: two monotonic paths,

[1] Naturally, one could have addressed the more general problem in which the lattice is itself presented in a general way through two functions *meet* and *join;* however, this framework (a) leads quickly and easily to intractability; and (b) does not capture any more applications in economics than the one treated here.

[2] This algorithm appears to have been first observed in [5].

one starting from $\bar{1}$ and the other from $\bar{N}$, meeting at the fixed point, while all other points in the $N \times N$ grid are mapped diagonally: $f(x) = x + (-1, +1)$ or $x + (+1, -1)$, whichever of the points is closer to the monotonic path that contains the fixed point. We prove that any randomized algorithm needs to make $\Omega(\log^2 N)$ queries (in expectation) to find the fixed point.

- Can this lower bound result be generalized to fixed $d \geq 3$? This is a key question left open by this paper. There are several obstacles to a proof establishing, e.g., a $\Omega(\log^3 N)$ lower bound in the 3-dimensional case ($d = 3$), and some possible ways for overcoming them. First, it is not easy to identify a suitable "herringbone-like" function in three or more dimensions – a monotone family of functions built around a path from $\bar{1}$ to $\bar{N}$. It nevertheless seems plausible that $\log^d N$ should still be (close to) a lower bound on any such algorithm (assuming of course that N is sufficiently larger than d, so that the dN algorithm does not violate the lower bound). We prove one encouraging result in this context: We give an alternative proof of the $d = 2$ lower bound, in which we establish that any *deterministic* black-box algorithm for TARSKI in two dimensions *must solve a sequence of* $\Omega(\log N)$ *one-dimensional problems* (Theorem 13), a result pointing to a possible induction on d (recall that this is precisely the form of the $\log^d N$ algorithm).
- Tarski's theorem further asserts that there is a greatest and a least fixed point, and these fixed points are especially useful in the economic applications of the result (see for example [17]). It is not hard to see, however, that finding these fixed points is **NP**-hard, and takes $\Omega(dN)$ time in the black box model (see Proposition 1).
- In terms of complexity classes, the problem TARSKI is obviously in the class **TFNP** of total function (total search) problems. But where exactly? We show (Theorem 4) that it belongs in the class PLS of local optimum search problems.
- Surprisingly, TARSKI is also in the class P^{PPAD} of problems reducible to a Brouwer fixed point problem (Theorem 5), and thus, by the known fact that the class PPAD is closed under polynomial time Turing reductions ([2]) it is in PPAD (Corollary 6). This result presents a heretofore unsuspected connection between two main sources of equilibrium results in economics.
- *Supermodular games* [23, 17, 24] – or games with strategic complementarities – comprise a large and important class of economic models, with complete lattices as strategy spaces, in which a player's best response is a monotone function (or monotone correspondence) of the other player's strategies. They always have pure Nash equilibria due to Tarski's theorem. We show that finding an equilibrium for a supermodular game with (discrete) Euclidean grid strategy spaces is essentially computationally equivalent to the problem of finding a Tarski fixed point of a monotone map (Proposition 14 and Theorem 16). If there are two players and one of them has a one-dimensional strategy space, we show that a Nash equilibrium can be found in logarithmic time (in the size of the strategy spaces).
- *Stochastic games* [21, 4]. We show that the problems of computing the (irrational) value of Shapley's discounted stochastic games to desired accuracy, and computing the exact value of Condon's simple stochastic games (SSG), are both P-time reducible to the Tarski problem. The proofs employ known characterizations of the value of both Shapley's stochastic games and Condon's SSGs in terms of monotone fixed point equations, which can also be viewed as monotone "polynomially contracting" maps with a unique fixed point, and from properties of polynomially contracting maps, see [12].

Prior related work. In recent years a number of technical reports and papers by Dang, Qi, and Ye, have considered the complexity of computational problems related to Tarski's theorem [5, 7, 6]. In particular, in [5] the authors provided the already-mentioned $\log^d N$

algorithm for computing a Tarski fixed point for a discrete map, $f : [N]^d \to [N]^d$, which is monotone under the coordinate-wise order. In [5] they also establish that determining the uniqueness of the fixed point of a monotone map under coordinate-wise order is coNP-hard, and that uniqueness under lexicographic order is also coNP-hard (already in one dimension). In [7] the authors studied another variant of the Tarski problem, namely computing another fixed point of a monotone function in an expanded domain where the smallest point is a fixed point; this variant is NP-hard (the claim in the paper that this problem is in PPA has been withdrawn by the authors [8]). In earlier work, Echenique [10], studied algorithms for computing all pure Nash equilibria in supermodular games (and games with strategic complementaries) whose strategy spaces are discrete grids. Of course computing all pure equilibria is harder than computing *some* pure equilibrium; indeed, we show that computing the least (or greatest) pure equilibrium of such a supermodular game is already NP-hard (Corollary 17). In earlier work Chang, Lyuu, and Ti [3] considered the complexity of Tarski's fixed point theorem over a general finite lattice given via an oracle for its partial order (not given it explicitly) and given an oracle for the monotone function, and they observed that the total number of oracle queries required to find some fixed point in this model is linear in the number of elements of the lattice. They did not study monotone functions on euclidean grid lattices, and their results have no bearing on this setting.

Organization of the paper. The rest of the paper is organized as follows. Section 2 provides basic definitions on lattices and monotone functions, and presents some simple basic results. In Section 3 we define the TARSKI problem and show that it is in PLS and PPAD. Section 4 proves the lower bound of $\log^2 N$ on black-box algorithms. Section 5 concerns supermodular games. Section 6 reduces Condon's and Shapley's stochastic games to the TARSKI problem. Finally, Section 7 concludes and discusses open problems. Several of the proofs are only sketched or omitted; more details can be found in the full version of the paper [11].

2 Basics

A partial order $(L, \leq)$ is a *complete lattice* if every nonempty subset S of L has a least upper bound (or supremum or join, denoted $\sup S$ or $\vee S$) and a greatest lower bound (or infimum or meet, denoted $\inf S$ or $\wedge S$) in L. A function $f : L \to L$ is *monotone* if for all pairs of elements $x, y \in L$, $x \leq y$ implies $f(x) \leq f(y)$. A point $x \in L$ is a *fixed point* of f if $f(x) = x$. Tarski's theorem ([22]) states that the set $Fix(f)$ of fixed points of f is a nonempty complete lattice under the same partial order $\leq$; in particular, f has a greatest fixed point (GFP) and a least fixed point (LFP).

In this paper we will take as our underlying lattice L a finite discrete Euclidean grid, which we fix for simplicity to be the integer grid $[N]^d$, for some positive integers N, d, where $[N] = \{1, \ldots, N\}$. Comparison of points is componentwise, i.e. $x \leq y$ if $x_i \leq y_i$ for all $i = 1, \ldots, d$. We will also consider the corresponding continuous box, $[1, N]^d$ that includes all real points in the box. Both, the discrete and continuous box are clearly complete lattices.

Given a monotone function f on the integer grid $[N]^d$, the problem is to compute a fixed point of f (any point in $Fix(f)$). A generally harder problem is to compute specifically the LFP of f or the GFP of f. We consider mostly the oracle model, in which the function f is given by a black-box oracle, and the complexity of the algorithm is measured in terms of the number of queries to the oracle. Alternatively, we can consider also an explicit model in which f is given explicitly by a polynomial-time algorithm (a polynomial-size Boolean circuit), and then the complexity of the algorithm is measured in the ordinary Turing model. Note

that the number of bits needed to represent a point in the domain is $d \log N$, so polynomial time here means polynomial in d and $n = \log N$. The number N^d of points in the domain is exponential.

Tarski's value iteration algorithm provides a simple way to compute the LFP of f: Starting from the lowest point of the lattice, which here is the all-1 vector 1, apply repeatedly f. This generates a monotonically increasing sequence of points $1 \leq f(1) \leq f^2(1) \leq \ldots$ until a fixed point is reached, which is the LFP of f. In every step of the sequence, at least one coordinate is strictly increased, therefore a fixed point is reached in at most $(N-1)d$ steps. In the worst case, the process may take that long, which is exponential in the bit size $d \log N$. Similarly, the GFP can be computed by applying repeatedly f starting from the highest point of the lattice, i.e., from the all-N point, until a fixed point is reached.

Another way to compute some fixed point of a monotone function f (not necessarily the LFP or the GFP) is by a divide-and-conquer algorithm. In one dimension, we can use binary search: If the domain is the set $L(l,h) = \{x \in Z | l \leq x \leq h\}$ of integers between the lowest point l and the highest point h, then compute the value of f on the midpoint $m = (l+h)/2$. If $f(m) = m$ then m is a fixed point; if $f(m) < m$ then recurse on the lower half $L(l,m)$, and if $f(m) > m$ then recurse on the upper half $L(m,h)$. The monotonicity of f implies that f maps the respective half interval into itself. Hence the algorithm correctly finds a fixed point in at most $\log N$ iterations, where N is the number of points.

In the general d-dimensional case, suppose that the domain is the set of integer points in the box defined by the lowest point l and the highest point h, i.e. $L(l,h) = \{x \in Z^d | l \leq x \leq h\}$. Consider the set of points with d-th coordinate equal to $m = (l+h)/2$; their first $d-1$ coordinates induce a $(d-1)$-dimensional lattice $L'(l,h) = \{x \in Z^{d-1} | l_i \leq x_i \leq h_i, i = 1, \ldots d-1\}$. Define the function g on $L'(l,h)$ by letting $g(x)$ consist of the first $d-1$ components of $f(x,m)$. It is easy to see that g is a monotone function on $L'(l,h)$. Recursively, compute a fixed point x^* of g. If $f_d(x^*,m) = m$, then (x^*,m) is a fixed point of f (this holds in particular if $l = h$). If $f_d(x^*,m) > m$, then recurse on $L(f(x^*,m),h)$. If $f_d(x^*,m) < m$, then recurse on $L(l,f(x^*,m))$. In either case, monotonicity implies that if the algorithm recurses, then f maps the smaller box into itself and thus has a fixed point in it. An easy induction shows that the complexity of this algorithm is $O((\log N)^d)$, ([5]).

Computing the least or the greatest fixed point is in general hard, even in one dimension, both in the oracle and in the explicit model.

▶ **Proposition 1.** *Computing the LFP or the GFP of an explicitly given polynomial-time monotone function in one dimension is NP-hard. In the oracle model, the problem requires $\Omega(N)$ queries for a domain of size N.*

Proof. We prove the claim for the LFP; the GFP is similar. Reduction from Satisfiability. Given a Boolean formula ϕ in n variables, let the domain $D = \{0, 1, \ldots, 2^n\}$, and define the function f as follows. For $x \leq 2^n - 1$, viewing x as an n-bit binary number, it corresponds to an assignment to the n variables of ϕ; let $f(x) = x$ if the assignment x satisfies ϕ, and let $f(x) = x+1$ otherwise. Define $f(2^n) = 2^n$. Clearly f is a monotone function and it can be computed in polynomial time. If ϕ is not satisfiable then the LFP of f is 2^n, while if ϕ is satisfiable then the LFP is not 2^n.

For the oracle model, use the same domain D and let f map every $x \leq 2^n - 1$ to x or $x+1$, and $f(2^n) = 2^n$. The LFP is not 2^n iff there exists an $x \leq 2^n - 1$ such that $f(x) = x$, which in the oracle model requires trying all possible $x \leq 2^n - 1$. ◀

In the case of a continuous domain $[1,N]^d$, we may not be able to compute an exact fixed point, and thus we have to be content with approximation. Given an $\epsilon > 0$, an ϵ-approximate fixed point is a point x such that $|f(x) - x| \leq \epsilon$, where we use the L_∞ (max)

norm, i.e. $|f(x) - x| = \max\{|f_i(x) - x_i| | i = 1, \ldots, d\}$. In this context, polynomial time means polynomial in $\log N, d$, and $\log(1/\epsilon)$ (the number of bits of the approximation). An ϵ-approximate fixed point need not be close to any actual fixed point of f. A problem that is generally harder is to compute a point that approximates some actual fixed point, and an even harder task is to approximate specifically the LFP or the GFP of f. Tarski's value iteration algorithm, starting from the lowest point converges in the limit to the LFP (and if started from the highest point, it converges to the GFP), but there is no general bound on the number of iterations needed to get within ϵ of the LFP (or the GFP). The algorithm reaches however an ϵ-approximate fixed point within Nd/ϵ iterations (note, this is exponential in $\log N, \log(1/\epsilon)$).

It is easy to see that the approximate fixed point problem for the continuous case reduces to the exact fixed point problem for the discrete case.

▶ **Proposition 2.** *The problem of computing an ϵ-approximate fixed point of a given monotone function on the continuous domain $[1, N]^d$ reduces to the exact fixed point problem on a discrete domain $[N/\epsilon]^d$.*

Proof. Given the monotone function f on the continuous domain $D_1 = [1, N]^d$, consider the discrete domain $D_2 = \{x \in Z^d | k \leq x_i \leq Nk, i = 1, \ldots, d\}$, where $k = \lceil 1/\epsilon \rceil$, and define the function g on D_2 as follows. For every $x \in D_2$, let $g(x)$ be obtained from $kf(x/k)$ by rounding each coordinate to the nearest integer, with ties broken (arbitrarily) in favor of the ceiling. Since f is monotone, g is also monotone. If x^* is a fixed point of g, then $kf(x^*/k)$ is within $1/2$ of x^* in every coordinate, and hence $f(x^*/k)$ is within $1/2k < \epsilon$ of x^*/k. Thus x^*/k is an ϵ-approximate fixed point of f. ◀

3 Computing a Tarski fixed point is in PLS ∩ PPAD

For a monotone function $f : [N]^d \to [N]^d$ (with respect to the coordinate-wise ordering), we are interested in computing a fixed point $x^* \in \text{Fix}(f)$, which we know exists by Tarski's theorem. We shall formally define this as a discrete total search problem, using a standard construction to avoid the "promise" that f is monotone.

Recall that a general discrete *total search problem* (with polynomially bounded outputs), Π, has a set of *valid input instances* $D_\Pi \subseteq \{0,1\}^*$, and associates with each valid input instance $I \in D_\Pi$, a non-empty set $\mathcal{O}_I \subseteq \{0,1\}^{p_\Pi(|I|)}$ of *acceptable outputs*, where $p_\Pi(\cdot)$ is some polynomial. (So the bit encoding length of every acceptable output is polynomially bounded in the bit encoding length of the input I.) We are interested in the complexity of the following total search problem:

▶ **Definition 3.** `Tarski`:

Input: *A function $f : [N]^d \to [N]^d$ with $N = 2^n$ for some $n \geq 1$, given by a boolean circuit, C_f, with $(d \cdot n)$ input gates and $(d \cdot n)$ output gates.*

Output: *Either a (any) fixed point $x^* \in \text{Fix}(f)$, or else a witness pair of vectors $x, y \in [N]^d$ such that $x \leq y$ and $f(x) \not\leq f(y)$.*

Note `Tarski` is a *total* search problem: If f is monotone, it will contain a fixed point in $[N]^d$, and otherwise it will contain such a witness pair of vectors that exhibit non-monotonicity. (If it is non-monotone it may of course have both witnesses for non-monotonicity and fixed points.)

Tarski ∈ PLS

Recall that a total search problem, Π, is in the complexity class PLS (*Polynomial Local Search*) if it satisfies all of the following conditions (see [16, 26]):

1. For each valid input instance $I \in D_\Pi \subseteq \{0,1\}^*$ of Π, there is an associated non-empty set $S_I \subseteq \{0,1\}^{p(|I|)}$ of *solutions*, and an associated *payoff function*[3], $g_I : S_I \to \mathbb{Z}$. For each $s \in S_I$, there is an associated set of *neighbors*, $\mathcal{N}_I(s) \subseteq S_I$.
   A solution $s \in S_I$ is called a *local optimum* (local maximum) if for all $s' \in \mathcal{N}_I(s)$, $g_I(s) \geq g_I(s')$. We let $\mathcal{O}_I$ denote the set of all local optima for instance I. (Clearly $\mathcal{O}_I$ is non-empty, because S_I is non-empty.)
2. There is a polynomial time algorithm, A_Π, that given a string $I \in \{0,1\}^*$, decides whether I is a valid input instance $I \in D_\Pi$, and if so outputs some solution $s_0 \in S_I$.
3. There is a polynomial time algorithm, B_Π, that given valid instance $I \in D_\Pi$ and a string $s \in \{0,1\}^{p(|I|)}$, decides whether $s \in S_I$, and if so, outputs the payoff $g_I(s)$.
4. There is a polynomial time algorithm, H_Π, that given valid instance $I \in D_\Pi$ and $s \in S_I$, decides whether s is a local optimum, i.e., whether $s \in \mathcal{O}_I$, and otherwise computes a strictly improving neighbor $s' \in \mathcal{N}_I(s)$, such that $g_I(s') > g_I(s)$.

▶ **Theorem 4.** Tarski ∈ PLS.

Proof Sketch. Each valid input instance $I_f \in D_{\texttt{Tarski}} \subseteq \{0,1\}^*$ of Tarski is an encoding of a function $f : [N]^d \to [N]^d$ via a boolean circuit C_f. We can view the problem Tarski as a polynomial local search problem, as follows:

Define the set of "solutions" associated with valid input I_f to be the disjoint union $S_{I_f} = S'_{I_f} \cup S''_{I_f}$, where $S'_{I_f} = \{x \in [N]^d \mid x \leq f(x)\}$ and $S''_{I_f} = \{(x,y) \in [N]^d \times [N]^d \mid x \leq y \wedge f(x) \not\leq f(y)\}$. Clearly, $\mathrm{Fix}(f) \subseteq S'_{I_f} \subseteq S_{I_f}$. Let the payoff function $g_{I_f} : S_{I_f} \to \mathbb{Z}$, be defined as follows. For $x \in S'_{I_f}$, $g_{I_f}(x) := \sum_{i=1}^d x_i$; for $(x,y) \in S''_{I_f}$, $g_{I_f}(x,y) := (dN)+1$. We define the neighbors of solutions as follows. For any $x \in S'_{I_f}$, if $f(x) \leq f(f(x))$ then let the neighbors of x be the singleton-set $\mathcal{N}_{I_f}(x) := \{f(x)\}$. Note that in this case again $f(x) \in S'_{I_f}$. Otherwise, if $f(x) \not\leq f(f(x))$, then let $\mathcal{N}_{I_f}(x) := \{(x, f(x))\}$. Note that in this case $(x, f(x)) \in S''_{I_f}$, since $f(x) \not\leq f(f(x))$. For $(x,y) \in S''_{I_f}$, let $\mathcal{N}_{I_f}(x,y) := \emptyset$ be the empty set. Thus, the set of local optima is by definition $\mathcal{O}_{I_f} = \{x \in S'_{I_f} \mid \sum_{i=1}^d x_i \geq \sum_{i=1}^d f_i(x)\} \cup S''_{I_f}$.

Observe that in fact $\mathcal{O}_{I_f} = \mathrm{Fix}(f) \cup S''_{I_f}$. Indeed, if $x \in \mathcal{O}_{I_f}$ then $x \in S'_{I_f}$ meaning $x \leq f(x)$, and also $\sum_{i=1}^d x_i \geq \sum_{i=1}^d f_i(x)$. But this is only possible if $f(x) = x$, i.e., $x \in \mathrm{Fix}(f)$. Likewise, if $(x,y) \in \mathcal{O}_{I_f}$ then $(x,y) \in S''_{I_f}$. On the other hand, if $x \in \mathrm{Fix}(f)$, then clearly $x \in S'_{I_f}$ and $\sum_{i=1}^d x_i = \sum_{i=1}^d f_i(x)$, hence $x \in \mathcal{O}_{I_f}$.

It is possible then to define polynomial time algorithms $A_{\texttt{Tarski}}$, $B_{\texttt{Tarski}}$ and $H_{\texttt{Tarski}}$, as required in conditions 2, 3, 4 in the definition of PLS; see the full paper for details. It follows that Tarski is in PLS. ◀

Tarski ∈ PPAD

To show that Tarski ∈ PPAD, we first show that Tarski $\in P^{\mathsf{PPAD}}$ meaning that the total search problem Tarski can be solved by a polynomial time algorithm, $\mathcal{M}$, with oracle access to PPAD. The algorithm $\mathcal{M}$ should take an input $I_f \in \{0,1\}^*$, and firstly decide whether it is a valid instance $I_f \in D_{\texttt{Tarski}}$, and if so it can make repeated, adaptive, calls to an

[3] Or, cost function, if we were considering local minimization. But here we focus on local maximization.

oracle for solving a PPAD total search problem. After at most polynomial time (and hence polynomially many such oracle calls) as a function of the input size $|I_f|$, $\mathcal{M}$ should output either an integer vector $x \in \mathrm{Fix}(f)$, or else output a pair of vectors $x, y \in [N]^d$ with $x \leq y$ and $f(x) \not\leq f(y)$, which witness non-monotonicity of the function $f : [N]^d \to [N]^d$ defined by the input instance I_f.

Once we have established that $\texttt{Tarski} \in P^{\mathsf{PPAD}}$, the fact that $\texttt{Tarski} \in \mathsf{PPAD}$ will follow as a simple corollary, using a prior result of Buss and Johnson [2], who showed that PPAD is closed under polynomial-time Turing reductions.

There are a number of equivalent ways to define the total search complexity class PPAD. Rather than give the original definition ([20]), we will use an equivalent characterization of PPAD (a.k.a., linear-FIXP) from [12]. Informally, according to this characterization, a discrete total search problem, Π, is in PPAD if and only if it can be reduced in P-time to computing a Brouwer fixed point of an associated "polynomial piecewise-linear" continuous function that maps a non-empty convex polytope to itself. That is, every instance I of Π can be associated with a polynomial piecewise-linear function F_I on a polytope $W(I)$, such that from any rational fixed point of F_I we can obtain in polynomial time an acceptable output for the instance I. By Brouwer's theorem, the set $\mathrm{Fix}(F_I) = \{x \in W(I) \mid F_I(x) = x\}$ of fixed points of F_I is non-empty. Moreover, because of the "polynomial piecewise-linear" nature of F_I, $\mathrm{Fix}(F_I)$ must also contain a *rational* fixed point x^*, with polynomial bit complexity as a function of $|I|$ (see [12], Theorem 5.2). See [12], section 5, for more details on this characterization of PPAD.

Given two vectors $l \leq h \in \mathbb{Z}^d$, let $L(l, h) = \{x \in \mathbb{Z}^d \mid l \leq x \leq h\}$, and let $B(l, h) = \{x \in \mathbb{R}^d \mid l \leq x \leq h\}$.

▶ **Theorem 5.** $\texttt{Tarski} \in P^{\mathsf{PPAD}}$.

Proof Sketch. Suppose we are given an instance $I_f \in D_{\texttt{Tarski}}$ of Tarski, corresponding to a function $f : [N]^d \to [N]^d$ (given by a boolean circuit C_f).

Let $a = \mathbf{1} \in \mathbb{Z}^d$, and $b = \mathbf{N} \in \mathbb{Z}^d$, denote the all 1, and all N, vectors respectively. We first extend the discrete function f to a (polynomial piecewise-linear) continuous function $f' : B(a, b) \to B(a, b)$, by a suitable linear interpolation. For this purpose we use a specific simplicial subdivision of $B(a, b)$, known as *Freudenthal's simplicial division* [15], which has a certain monotonicity property that is important for the proof. By Brouwer's theorem, f' has a fixed point in $B(a, b)$, and since it is polynomial piecewise-linear, finding a fixed point x^* is in PPAD. However, f' may have non-integer fixed points that do not correspond to (and are not close to) any fixed point of f (indeed, since we do not apriori know that f is monotone, there may not be any integer fixed points). Nevertheless, we show that finding any such fixed point x^* of f' allows us to make progress towards either finding a discrete fixed point of f (if it is monotone), or finding witnesses for a violation of monotonicity of f.

Specifically, we argue that there are two (integer) vertices $u \geq v$ of the simplex of the subdivision that contains x^* such that, if f is monotone, then $f(u) \geq u$ and $f(v) \leq v$ (in all coordinates). If $f(u) \not\geq u$, or $f(v) \not\leq v$, then f is not monotone, and we show that we can find a witness pair for the non-monotonicity of f.

Assume on the other hand that $f(u) \geq u$ and $f(v) \leq v$. Note that in that case, if f is monotone, then f maps the sublattice $L(u, b)$ to itself, and it also maps the disjoint sublattice $L(a, v)$ to itself. Thus, if f is monotone, f must have an integer fixed point in both $L(a, v)$ and $L(u, b)$.

So, we can choose the smaller of these two sublattices, consider the function f restricted to that sublattice, and continue recursively to find a fixed point in that sublattice (if f is monotone) or a violation of monotonicity. If f is not monotone, it is possible that it maps

some points in the sublattice $L(a, v)$ (or $L(u, b)$) to points outside. Therefore, in the recursive call for the sublattice, when we define the piecewise-linear function f' on the corresponding box $B(a, v)$ (or $B(u, b)$) we take the maximum with a and minimum with v (or u and b respectively), i.e., threshold it, so that it maps the box to itself, and hence it is a Brouwer function. When the PPAD oracle gives us back a fixed point for this (possibly thresholded) function f', we argue that if the thresholding mattered in this regard, then we can detect it and produce a violating pair to monotonicity.

Every iteration decreases the total number of points in our current lattice by a factor of 2, from the number of points in the original lattice $L(a, b)$. So after a polynomial number of iterations in $(d + \log N)$, we either find a fixed point of f, or we find a witness pair of integer vectors that witness the non-monotonicity of f. ◀

▶ **Corollary 6.** `Tarski` $\in$ PPAD.

Proof. This follows immediately from Theorem 5, combined with a result due to Buss and Johnson ([2], Theorem 6.1), who showed that PPAD is closed under polynomial-time Turing reductions. ◀

4 The 2-dimensional lower bound

Consider a monotone function defined on the $N \times N$ grid $f : [N]^2 \mapsto [N]^2$. Let A be any (randomized) *black-box algorithm* for finding a fixed point of the function by computing a sequence of queries of the form $f(x, y) =?$; A can of course be *adaptive* in that any query can depend in arbitrarily complex ways on the answers to the previous queries. For example, the divide-and-conquer algorithm described in the introduction is a black box algorithm. The following result suggests that this algorithm is optimal for two dimensions.

▶ **Theorem 7.** *Given black-box access to a monotone function $f : [N]^2 \to [N]^2$, any (randomized) algorithm for finding a fixed point of f requires $\Omega(\log^2 N)$ queries (in expectation).*

Below, we *sketch* a proof. For the full proof see [11]. The proof constructs a hard distribution of such functions.

The basic construction

Given a monotone path from $(1, 1)$ to (N, N) on the $N \times N$ grid graph and a point (i^*, j^*) on the path, we construct f as follows:

- We let (i^*, j^*) be the unique fixed point of f, i.e. $f(i^*, j^*) \triangleq (i^*, j^*)$.
- At all other points on the path, f is directed towards the fixed point. For a point (x, y) on the path that is dominated by (i^*, j^*), we let $f(x, y)$ be the next point on the path, i.e. $f(x, y) = (x + 1, y)$ or $f(x, y) = (x, y + 1)$. Similarly, for a point (x, y) that is on the path and dominates (i^*, j^*), we let $f(x, y)$ be the previous point on the path.
- For all points outside the path, f is directed towards the path. Observe that the path partitions $[N]^2$ into three (possibly empty) subsets: below the path, the path, and above the path. For a point (x, y) below the path, we set $f(x, y) \triangleq (x - 1, y + 1)$. Similarly, for a point (x, y) above the path, $f(x, y) \triangleq (x + 1, y - 1)$.

An example of such a function $f : [5]^2 \to [5]^2$ is given in Figure 1.

▷ Claim 8. For any choice of path and point (i^*, j^*) on the path, f constructed as above is monotone.

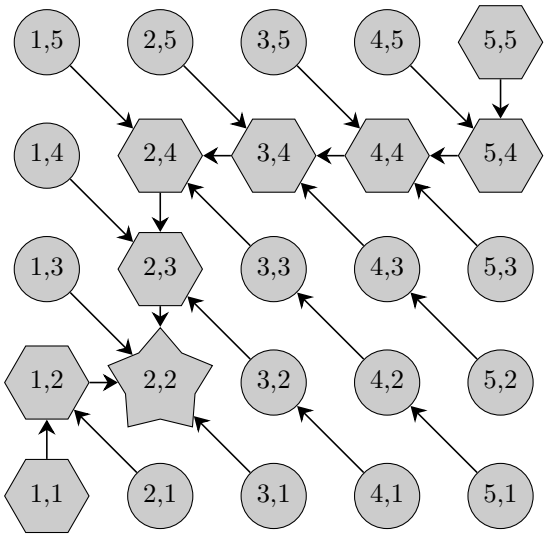


Figure 1 A 2-dimensional "*herringbone*" monotone function.

Choosing the fixed point

In our hard distribution, once we fix a path, we choose (i^*, j^*) uniformly at random among all points on the path.

▷ Claim 9. Given oracle access to f and the path, any (randomized) algorithm that finds a point (i', j') on the path that is within $\sqrt{N}$ (Manhattan distance) from (i^*, j^*) requires (in expectation) querying of f at $\Omega(\log N)$ points on the path that are pairwise at least $\sqrt{N}$ apart.

Proof. Observe that once we fix the path, the values of f outside the path do not reveal information about the location of (i^*, j^*). The lower bound now follows from the standard lower bound for binary search. ◁

Choosing the central path

Our goal now is to prove that it is hard to find many distant points on the path. To simplify the analysis, we will only consider the special case where all points (x, y) on the path satisfy $x - y \in \left[-N^{1/4}, N^{1/4}\right]$. We partition the $N \times N$ grid into $\Theta\left(\sqrt{N}\right)$ regions of the form $R_a \triangleq \left\{(x, y) \mid x + y \in [a, a + \sqrt{N})\right\}$. Notice that each region intersects the path at exactly $\sqrt{N}$ points. The path enters each region[4] at a point (x, y) for a value $x - y$ chosen uniformly at random among $\left[-N^{1/4}, N^{1/4}\right]$. We will argue (Lemma 12 below) that in order to find a point on the path in any region R_a, the algorithm must query the function at $\Omega(\log N)$ points in R_a or its neighboring regions.

Each region is further partitioned into $\Theta\left(N^{1/4}\right)$ sub-regions $S_a \triangleq \left\{(x, y) \mid x + y \in [a, a + 2N^{1/4})\right\}$. For each region, we choose a special sub-region uniformly at random. In all non-special sub-regions, the path proceeds while maintaining

[4] For the first and last region, the path is obviously forced to start at $(1, 1)$ (respectively end at (N, N)); but those two regions can only account for two of the $\Omega(\log N)$ distant path points required by Claim 9, so we can safely ignore them.

$x-y$ fixed, up to ± 1. Inside the special sub-region, the value of $x-y$ for path points changes from the value chosen at random for the current region, to the value chosen at random for the next region.

Given a choice of random $x-y$ entry point for each region, and a random special sub-region for each region, we consider an arbitrary path that satisfies the description above. This completes the description of the construction.

▷ Claim 10. Finding (i.e., querying any point in) the special sub-region in region R_a requires $\Omega(\log N)$ queries (in expectation) to points in R_a.

Let S_a and S_b be the special sub-regions of two consecutive regions. Let $T \triangleq \{(x,y) \mid x+y \in [a+2N^{1/4}, b)\}$ be the union of all the sub-regions between S_a and S_b. Observe that the value of $x-y$ remains fixed (up to ± 1) for all points in the intersection of the path with T. Also, the construction of f outside $S_a \cup T \cup S_b$ does not depend at all on this value.

▷ Claim 11. If the algorithm does not query any point in $S_a \cup S_b$, then in order to find (i.e., query) any point in the intersection of the path and T, the algorithm must query (in expectation) $\Omega(\log N)$ points from T.

By Claim 10, finding S_a or S_b requires at least $\Omega(\log N)$ queries to the regions containing them. Therefore, the above two claims together imply:

▶ **Lemma 12.** *In order to query a point in the intersection of the path and region R_a, any algorithm must query at least $\Omega(\log N)$ points (in expectation) in R_a or its neighboring regions.*

Therefore, in order to find $\Omega(\log N)$ points on the path that are pairwise at least $\sqrt{N}$ apart, the algorithm must make a total of $\Omega\left(\log^2 N\right)$ queries (in expectation), completing the proof of Theorem 7.

An alternative proof

In the full version of this paper (see [11]) we provide an alternative proof, showing that any *deterministic* black box algorithm requires $\Omega(\log^2 N)$ oracle queries to find a Tarski fixed point of a monotone function $f : [N]^2 \to [N]^2$ given by an oracle.

▶ **Theorem 13.** *Any deterministic black box algorithm for finding a Tarski fixed point in two dimensions needs $\Omega(\log^2 N)$ queries.*

The alternative proof appears to be more promising for generalization to higher dimensions. However, the underlying monotone functions $f : [N]^2 \to [N]^2$ on which the alternative lower bound is established are again the "*herringbone*" functions used in the proof of Theorem 7. The alternative proof uses a potential argument, and its gist amounts to showing that any such algorithm must solve $\Omega(\log N)$ *independent* one-dimensional problems.

5 Supermodular Games

A brief intro to supermodular games

A *supermodular game* is a game in which the set S_i of strategies of each player i is a complete lattice, and the utility (payoff) functions u_i satisfy certain conditions. Let k be the number of players and let $S = \Pi_{i=1}^{k} S_i$ be the set of strategy profiles. As usual, we use s_i to denote a strategy for player i and s_{-i} to denote a tuple of strategies for the other players. The conditions on the utility functions u_i are the following:

C1. $u_i(s_i, s_{-i})$ is upper semicontinuous in s_i for fixed s_{-i}, and it is continuous in s_{-i} for each fixed s_i, and has a finite upper bound.

C2. $u_i(s_i, s_{-i})$ is supermodular in s_i for fixed s_{-i}.

C3. $u_i(s_i, s_{-i})$ has increasing differences in s_i and s_{-i}.

A function $f : L \to \mathbb{R}$ is *supermodular* if for all $x, y \in L$, it holds $f(x) + f(y) \leq f(x \wedge y) + f(x \vee y)$. A function $f : L_1 \times L_2 \to \mathbb{R}$, where L_1, L_2 are lattices, has *increasing differences* in its two arguments, if for all $x' \geq x$ in L_1 and all $y' \geq y$ in L_2, it holds that $f(x', y') - f(x, y') \geq f(x', y) - f(x, y)$.

The broader class of *games with strategic complementarities* (GSC) relaxes somewhat the conditions C2 and C3 into C2', C3' which depend only on ordinal information on the utility functions, i.e. how the utilities compare to each other rather than their precise numerical values. The supermodularity requirement of C2 is relaxed to quasi-supermodularity, where a function $f : L \to \mathbb{R}$ is *quasi-supermodular* if for all $x, y \in L$, $f(x) \geq f(x \wedge y)$ implies $f(y) \leq f(x \vee y)$, and if the first inequality is strict, then so is the second. The increasing differences requirement of C3 is relaxed to the *single-crossing condition*, where a function $f : L_1 \times L_2 \to \mathbb{R}$, satisfies the single crossing condition, if for all $x' > x$ in L_1 and all $y' > y$ in L_2, it holds that $f(x', y) \geq f(x, y)$ implies $f(x', y') \geq f(x, y')$, and if the first inequality is strict then so is the second. All the structural and algorithmic properties below of supermodular games hold also for games with strategic complementarities.

We will consider here games where each S_i is a discrete (or continuous) finite box in d_i dimensions of size N in each coordinate. We let $d = \sum_{i=1}^{k} d_i$ be the total number of coordinates. In the discrete case, condition C1 is trivial. Condition C2 is trivial if $d_i = 1$ (all functions in one dimension are supermodular), but nontrivial for 2 or more dimensions. C3 is nontrivial.

Supermodular games (and GSC) have pure Nash equilibria. Furthermore, the pure Nash equilibria form a complete lattice [17], thus there is a highest and a lowest equilibrium. Another important property is that the best response correspondence $\beta_i(s_{-i})$ for each player i has the property that (1) both $\sup \beta_i(s_{-i})$ and $\inf \beta_i(s_{-i})$ are in $\beta_i(s_{-i})$, and (2) both functions $\sup \beta_i(\cdot)$ and $\inf \beta_i(\cdot)$ are monotone functions [24]. The function $\bar{\beta}(s) = (\sup \beta_1(s_{-1}), \ldots, \sup \beta_k(s_{-k}))$ of the supremum best responses is a monotone function from S to itself, and its greatest fixed point is the highest Nash equilibrium of the game. The function $\underline{\beta}(s) = (\inf \beta_1(s_{-1}), \ldots, \inf \beta_k(s_{-k}))$ of the infimum best responses is also a monotone function, and its least fixed point is the lowest Nash equilibrium of the game.

Complexity of equilibrium computation in supermodular games

Given a supermodular game, the relevant problems include: (a) find a Nash equilibrium (anyone)[5], and (b) find the highest or the lowest equilibrium. In the case of continuous domains, we again have to relax to an approximate solution. We assume that we have access to a best response function, e.g. $\bar{\beta}(\cdot)$ and/or $\underline{\beta}(\cdot)$, as an oracle or as a polynomial-time function. The monotonicity of these functions implies then easily the following:

[5] Whenever we speak of finding a Nash Equilibrium (NE) for a supermodular game, we mean a *pure* NE, as we know that these exists.

▶ **Proposition 14.**

1. *The problem of computing a pure Nash equilibrium of a k-player supermodular game over a discrete finite strategy space $\Pi_{i=1}^{k}[N]^{d_i}$ reduces to the problem of computing a fixed point of a monotone function over $[N]^d$ where $d = \sum_{i=1}^{k} d_i$. Computing the highest (or lowest) Nash equilibrium reduces to computing the greatest (or lowest) fixed point of a monotone function.*
2. *For games with continuous box strategy spaces, $\Pi_{i=1}^{k}[1, N]_i^d$, and Lipschitz continuous utility functions with Lipschitz constant K, the problem of computing an ϵ-approximate Nash equilibrium reduces to exact fixed point computation point for a monotone function with a discrete finite domain $[NK/\epsilon]^d$.*

Proof.

1. Follows from the monotonicity of $\bar{\beta}(\cdot)$ and $\underline{\beta}(\cdot)$. If s is fixed point of $\bar{\beta}(\cdot)$, then $s_i = \sup \beta_i(s_{-i})$ is a best response to s_{-i} for all i (since $\sup \beta_i(s_{-i}) \in \beta_i(s_{-i})$), therefore s is a Nash equilibrium of the game. The GFP of $\bar{\beta}(\cdot)$ is the highest Nash equilibrium. Similarly, every fixed point of $\underline{\beta}(\cdot)$ is an equilibrium of the game, and the LFP of $\underline{\beta}(\cdot)$ is the lowest equilibrium.
2. Suppose that the utility functions are Lipschitz continuous with Lipschitz constant K. To compute an ϵ-approximate Nash equilibrium of the game, it suffices to find a ϵ/K-approximate fixed point of the function $\bar{\beta}(\cdot)$. For, if s is such an approximate fixed point and $s' = \bar{\beta}(s)$, then $|s' - s| \leq \epsilon/K$ in every coordinate. Hence $|u_i(s_i, s_{-i}) - u_i(s_i', s_{-i})| \leq \epsilon$, and s_i' is a best response to s_{-i}, hence s is an ϵ-approximate equilibrium. Computing an ϵ/K-approximate fixed point of the function $\bar{\beta}(\cdot)$ on the continuous domain, reduces by Proposition 2 to the exact fixed point problem for the discrete domain $[NK/\epsilon]^d$. ◀

Not every monotone function can be the (sup or inf) best response function of a game. In particular, a best response function has the property that the output values for the components corresponding to a player depend only on the input values for the other components corresponding to the other players. Thus, for example, for two one-dimensional players, if the function $f(x, y)$ is the best response function of a game, it must satisfy $f_1(x, y) = f_1(x', y)$ for all x, x', y, and $f_2(x, y) = f_2(x, y')$ for all x, y, y'. This property helps somewhat in improving the time needed to find a fixed point, and thus an equilibrium of the game, as noted below. For example, in the case of two one-dimensional players, an equilibrium can be computed in $O(\log N)$ time, instead of the $\Omega(\log^2 N)$ time needed to find a fixed point of a general monotone function in two dimensions.

▶ **Theorem 15.** *Given a supermodular game with two players with discrete strategy spaces $[N]^{d_i}$, $i = 1, 2$ with access to the sup (or inf) best response function $\bar{\beta}(\cdot)$ (or $\underline{\beta}(\cdot)$), we can compute an equilibrium in time $O((\log N)^{\min(d_1, d_2)})$. More generally, for k players with dimensions $d_1, \ldots, d_k$, an equilibrium can be computed in time $O((\log N)^{d'})$, where $d' = \sum_i d_i - \max_i d_i$.*

Proof. Suppose that we have access to the sup best response $\bar{\beta}(\cdot)$. Assume without loss of generality that the first player has the maximum dimension, $d_1 = \max_i d_i$. We apply the divide-and-conquer algorithm, but take advantage of the property of the monotone function $\bar{\beta}$ that the first d_1 components of $\bar{\beta}(x)$ do not depend on the first d_1 coordinates of x. As a consequence, for any fixed assignment to the other coordinates, i.e. choice of a strategy profile s_{-1} for all the players except the first player, the induced function on the first d_1 coordinates maps every point to the best response $\bar{\beta}_1(s_{-1})$ of player 1. Thus the fixed point of the induced function is simply $\bar{\beta}_1(s_{-1})$, it can be computed with one call to $\bar{\beta}$, and there is no need to recurse on the first d_1 coordinates. It follows that the algorithm takes time at most $O((\log N)^{d'})$, where $d' = \sum_i d_i - \max_i d_i$. ◀

Conversely, we can reduce the fixed point computation problem for an arbitrary monotone function to the equilibrium computation problem for a supermodular game with two players.

▶ **Theorem 16.**

1. *Given a monotone function f on $[N]^d$ (resp. $[1,N]^d$) we can construct a supermodular game G with two players, each with strategy space $[N]^d$ (resp. $[1,N]^d$), so that the pure Nash equilibria of G correspond to the fixed points of f.*
2. *More generally, the fixed point problem for a monotone function f in d dimensions can be reduced to the pure Nash equilibrium problem for a supermodular game with any number $k \geq 2$ of players with any dimensions $d_1, \ldots, d_k$, provided that $\sum_i d_i \geq 2d$ and $\sum_i d_i - \max_i d_i \geq d$.*

Proof Sketch.

1. We will define the utility functions u_i so that the best responses β_i of both players are functions (i.e. are unique). For player 1, the best response will be $\beta_1(y) = y$, for all $y \in [N]^d$, and for player 2, the best response will be $\beta_2(x) = f(x)$, for all $x \in [N]^d$. If x is a fixed point of f, then (x,x) is an equilibrium of the game, since $\beta(x,x) = (x, f(x)) = (x,x)$. Conversely, if (x,y) is an equilibrium of the game, then $\beta(x,y) = (x,y)$, therefore $x = y$ and $y = f(x)$, hence $x = f(x)$. Thus, the set of equilibria of G is $\{(x,x) | x \in Fix(f)\}$.
   The utility function for player 1 is set to $u_1(x,y) = -(x-y)^2 = -\sum_{j=1}^{d}(x_j - y_j)^2$. The utility function for player 2 is $u_2(x,y) = -(f(x)-y)^2 = -\sum_{j=1}^{d}(f_j(x) - y_j)^2$. Obviously, the best response functions are as stated above, $\beta_1(y) = y$ and $\beta_2(x) = f(x)$.
   It can be verified that the utility functions u_1, u_2 satisfy conditions C2 and C3 in the definition of supermodular games (C2 with equality actually).
2. Order the players in increasing order of their dimension, let T be the ordering of all the $\sum_{i=1}^{k} d_i$ coordinates consisting first of the set $Co(1)$ of coordinates of player 1 (in any order), then the set $Co(2)$ of coordinates of player 2, and so forth. Number the coordinates in the order T from 1 to $\sum_{i=1}^{k} d_i$, and label them cyclically with the labels $1, \ldots, d$.
   We define the (unique) best response function β as follows. For every coordinate $j \leq d$ (in the ordering T), we set $\beta_j(x) = f_j(x')$, where x' is a subvector of x with d coordinates that have distinct labels $1, \ldots, d$ and which belong to different players than coordinate j. The subvector x' is defined as follows. Suppose that coordinate j belongs to player r ($j \in Co(r)$), and let $t = \sum_{i=1}^{r-1} d_i$. If $d_r \leq d$, then x' is the subvector of x that consists of the first t coordinates (in the order T) and the coordinates $t+1+d, \ldots, 2d$; note that all these coordinates do not belong to player r. If $d_r > d$, then $r < k$ (since $\sum_i d_i - \max_i d_i \geq d$). In this case, let x' be the subvector of x consisting of the last d coordinates (in T); all of these belong to player $k \neq r$. For coordinates $j > d$, we set $\beta_j(x) = x_{j'}$, where $j' \in [d]$ is equal to $j \mod d$, unless j' belongs to the same player r as j, in which case $d_r > d$, hence $r \neq k$; in this case we set $\beta_j(x) = x_{j"}$ for some (any) coordinate $j"$ of the last player k that is labeled j'.
   We define the utility functions of the players so that they yield the above best response function β. Namely, we define the utility function of player i to be $u_i(x) = -\sum_{j \in Co(i)}(x_j - \beta_j(x))^2$. It can be verified as in part 1 that the utility functions satisfy conditions C2 and C3. It can be easily seen also that at any equilibrium of the game, all coordinates with the same label must have the same value, and the corresponding d-vector x is a fixed point of f. Conversely, for any fixed point x of f, the corresponding strategy profile of the game is an equilibrium. ◀

Since the 2-dimensional monotone fixed point problem requires $\Omega(\log^2 N)$ queries by Theorem 7, it follows that the equilibrium problem for two 2-dimensional players also requires $\Omega(\log^2 N)$ queries, which is tight because it can be also solved in $O(\log^2 N)$ time by Theorem 15. Similarly, for higher dimensions d, if the monotone fixed point problem requires $\Omega(\log^d N)$ queries then the equilibrium problem for two d-dimensional players is also $\Theta(\log^d N)$.

The same reduction from monotone functions to supermodular games of Theorem 16, combined with Proposition 1 implies the hardness of computing the highest and lowest equilibrium.

▶ **Corollary 17.** *It is NP-hard to compute the highest and lowest equilibrium of a supermodular game with two 1-dimensional players with explicitly given polynomial-time best response (and utility) functions.*

6 Condon's and Shapley's stochastic games reduce to `Tarski`

In this section, we show that computing the exact (rational) value of Condon's simple stochastic games ([4]), as well as computing the (irrational) value of Shapley's more general (stopping/discounted) stochastic games [21] to within a given desired error $\epsilon > 0$ (given in binary), is polynomial time reducible to `Tarski`.

A *simple stochastic game*[6] (SSG) is a 2-player zero-sum game, played on the vertices of an edge-labeled directed graph, specified by $G = (V, V_0, V_1, V_2, \delta)$, whose vertices $V = \{v_1, \ldots, v_n\}$ include two special sink vertices, a **0**-sink, v_{n-1}, and a **1**-sink, v_n, and where the rest of the vertices $V \setminus \{v_{n-1}, v_n\} = \{v_1, \ldots, v_{n-2}\}$ are partitioned into three disjoint sets V_0 (random), V_1 (max), and V_2 (min). The labeled directed edge relation is $\delta \subseteq (V \setminus \{v_{n-1}, v_n\}) \times ((0, 1] \cup \bot) \times V$. For each "random" node $u \in V_0$, every outgoing edge $(u, p_{u,v}, v) \in \delta$ is labeled by a positive probability $p_{u,v} \in (0, 1]$, such that these probabilities sum to 1, i.e., $\sum_{\{v \in V | (u,p_{u,v},v) \in \delta\}} p_{u,v} = 1$. We assume, for computational purposes, that the probabilities $p_{u,v}$ are rational numbers (given as part of the input, with numerator and denominator given in binary). The outgoing edges from "max" (V_1) and "min" (V_2) nodes have an empty label, "$\bot$". We assume each vertex $u \in V \setminus \{v_{n-1}, v_n\}$ has at least one outgoing edge. Thus in particular, for any node $u \in V_1 \cup V_2$ there exists an outgoing edge $(u, \bot, v) \in \delta$ for some $v \in V$. Finally, there is a designated start vertex $s \in V$.

A play of the game transpires as follows: a token is initially placed on s, the start node. Thereafter, during each "turn", when the token is currently on a node $u \in V$, unless u is already a sink node (in which case the game halts), the token is moved across an outgoing edge of u to the next node by whoever "controls" u. For a random node $u \in V_0$, which is controlled by "nature", the outgoing edge is chosen randomly according to the probabilities $(p_{u,v})_{v \in V}$. For $u \in V_1$, the outgoing edge is chosen by player 1, the max player, who aims to maximize the probability that the token will eventually reach the **1**-sink. For $u \in V_2$, the outgoing edge is chosen by player 2, the min player, who aims to minimize the probability that the token will eventually reach the **1**-sink. The game halts if the token ever reaches either of the two sink nodes.

[6] The definition we give here for SSGs is slightly more general than Condon's original definition in [4]. Specifically, Condon allows edge probabilities of 1/2 only, and also assumed that the game is a "stopping game", meaning it halts with probability 1, regardless of the strategies of the two players. It is well known that our more general definition does not alter the difficulty of computing the game value and optimal strategies: solving general SSGs can be reduced in P-time to solving SSGs in Condon's more restricted form.

For every possible start node $s = v_i \in V$, this zero-sum game has a well defined *value*, $q_i^* \in [0,1]$. This is, by definition, a probability such that player 1, the max player (and, respectively, player 2, the min player) has a strategy to "force" reaching the 1-sink with probability at least (respectively, at most) q_i^*, irrespective of what the other player's strategy is. In other words, these games are *determined*. Moreover q_i^* is a rational value whose encoding size, with numerator and denominator in binary, is polynomial in the bit encoding size of the SSG ([4]). Furthermore, both players have deterministic, memoryless (a.k.a., pure, positional) optimal strategies in the game (which do not depend on the specific start node s), in which for each vertex $u \in V_1$ (or $u \in V_2$) the max player (respectively the min player) chooses the same specific outgoing edge every time the token visits vertex u, regardless of the prior history of play prior to that visit to u.

Given an SSG, the goal is to compute the value of the game (starting at each vertex). Condon ([4]) already showed that the problem of deciding whether the value is $> 1/2$ is in NP $\cap$ co-NP, and it is a long-standing open problem whether this is in P-time. Moreover, the search problem of computing the value for an SSG is known to be in both PLS and PPAD (see [25] and [12]). In the full paper we show the following:

▶ **Proposition 18.** *The following total search problem is polynomial-time reducible to* `Tarski`*: Given an instance G of Condon's simple stochastic game, and given a start vertex $s = v_i \in V$, compute the exact (rational) value q_i^* of the game.*

Shapley's original stochastic games are more general, and involve simultaneous (independent) choices by the two players at each state (they are thus imperfect information games). The value of the game is in general irrational (even when all the input data is rational). We show that approximating the value of a Shapley game to within any given desired accuracy, $\epsilon > 0$ (given in binary as part of the input), is polynomial time reducible to `Tarski`.

Shapley's games are a class of two-player zero-sum "stopping", or equivalently "discounted", stochastic games. An instance of Shapley's stochastic game is given by $G = (V, A, P, s)$, where $V = \{v_1, \ldots, v_n\}$ is a set of n vertices (or "states"). $A = (A^1, A^2, \ldots, A^n)$ is a n-tuple of matrices, where, for each vertex, $v_i \in V$, A^i is an associated $m_i \times n_i$ *reward matrix*, where m_i and n_i are positive integers denoting, respectively, the number of distinct "actions" available to player 1 (the maximizer) and player 2 (the minimizer) at vertex v_i, and where for each pair of such actions, $j \in [m_i]$ and $k \in [n_i]$, $A^i_{j,k} \in \mathbb{Q}$ is a reward for player 1 (which we assume, for computational purposes, is a rational number given as input by giving its numerator and denominator in binary). Furthermore, for each vertex $v_i \in V$, and each pair of actions $j \in [m_i]$ and $k \in [n_i]$, $P^i_{j,k} \in [0,1]^n$ is a vector of probabilities on the vertices V, such that $0 \leq P^i_{j,k}(r)$, and $\sum_{r=1}^n P^i_{j,k}(r) < 1$, i.e., the probabilities sum to *strictly less than* 1. Again, we assume each such probability $P^i_{j,k}(r) \in \mathbb{Q}$ is a rational number given as input in binary. Finally, the game specifies a designated start vertex $s \in V$.

A play of Shapley's game transpires as follows: a token is initially placed on s, the start node. Thereafter, during each "round" of play, if the token is currently on some node $v_i \in V$, both players simultaneously and independently choose respective actions $j \in [m_i]$ and $k \in [n_i]$, and player 1 then receives the corresponding reward $A^i_{j,k}$ from player 2; thereafter, for each $r \in [n]$ with probability $P^i_{j,k}(r)$ the token is moved from node v_i to node v_r, and with the remaining positive probability $q^i_{j,k} = 1 - \sum_{r=1}^n P^i_{j,k}(r) > 0$, the game "halts". Let $q = \min\{q^i_{j,k} \mid i,j,k\} > 0$ be the minimum such halting probability at any state, and under any pair of actions. Since q is positive, i.e., since there is positive probability $\geq q > 0$ of halting after each round, a play of the game eventually halts with probability 1. The goal of player 1 (player 2) is to maximize (minimize, respectively) the expected total reward that player 1 receives from player 2 during the entire play. A *strategy* for each player

specifies, based in principle on the entire history of play thusfar, a probability distribution on the actions available at the current token location. Given strategies σ_1 and σ_2 for player 1 and 2, respectively, let $r_i(\sigma_1, \sigma_2)$ denote the expected total payoff to player 1, starting at node $s = v_i \in V$. Shapley [21] showed that these games have a *value*, meaning that $\sup_{\sigma_1} \inf_{\sigma_2} r_i(\sigma_1, \sigma_2) = \inf_{\sigma_2} \sup_{\sigma_1} r_i(\sigma_1, \sigma_2)$. In fact, Shapley showed that both players have optimal stationary (but randomized) strategies in such games, i.e., optimal strategies that only depend on the current node where the token is located, not the prior history of play, but where players can randomize on their choice of actions at each node.

Let $r_i^* = \sup_{\sigma_1} \inf_{\sigma_2} r_i(\sigma_1, \sigma_2)$ denote the game value starting at vertex $s = v_i \in V$.[7] We show the following in the full paper.

▶ **Proposition 19.** *The following total search problem is polynomial-time reducible to* `Tarski`*: Given an instance G of Shapley's stochastic game, and given $\epsilon > 0$ (in binary), compute a vector $r' \in \mathbb{Q}^n$ such that $\|r^* - r'\|_\infty < \epsilon$.*

7 Conclusions and Discussion

We have studied the complexity of computing a Tarski fixed point for a monotone function over a finite discrete Euclidean grid, and we have shown that this problem essentially captures the complexity of computing a (ϵ-approximate) pure Nash equilibrium of a supermodular game. We have also shown that computing the value of Condon's and Shapley's stochastic games reduces to this `Tarski` fixed point problem, where the monotone function is given succinctly (by a boolean circuit).

We have provided several upper bounds for the `Tarski` problem, showing that it is contained in both PLS and PPAD. On the other hand, in the oracle model, for 2-dimensional monotone functions $f : [N]^2 \to [N]^2$, we have shown a $\Omega(\log^2 N)$ lower bound for the (expected) number of (randomized) queries required to find a Tarski fixed point, which matches the $O(\log^d N)$ upper bound for $d = 2$.

A key question left open by our work is to improve the lower bounds in the oracle model to higher dimensions. It is tempting to conjecture that for small (fixed) dimension d, a lower bound close to $\Omega(\log^d N)$ holds. On the other hand, we know that this cannot hold for arbitrary d and N, because we also have the dN upper bound, which is better than $\log^d N$ when $d = \omega(\frac{\log N}{\log \log N})$.

Another interesting open question is the relationship between the `Tarski` problem and the total search complexity class CLS [9], as well as the closely related recently defined class EOPL (which stands for "End of Potential Line" [13, 14]). EOPL is contained in CLS, which is contained in both PLS and PPAD. Is `Tarski` in CLS (or in EOPL)? That would be remarkable, as the proof that it is in PPAD is currently quite indirect. Conversely, can `Tarski` be proved to be CLS-hard (EOPL-hard)? (Recall from the previous section that some key problems in CLS related to stochastic games do reduce to `Tarski`.)

Another question worth considering is the complexity of the `unique-Tarski` problem, where the monotone function is further assumed (promised) to have a *unique* fixed point. Is `unique-Tarski` easier than `Tarski`? Note that our $\Omega(\log^2 N)$ lower bound in the oracle model, in dimension $d = 2$, applies on the family of "herringbone" functions which do have a unique fixed point.

[7] Note that we could also define r_i^* as $r_i^* = \max_{\sigma_1} \min_{\sigma_2} r_i(\sigma_1, \sigma_2)$, due to the existence of optimal strategies.

Finally, in this paper we have studied the Tarski fixed point problem only in the setting of monotone functions on the Euclidean grid $[N]^d$. Let us remark, however, that it is possible to consider a more general black box model, for monotone functions $f : L \to L$, over an arbitrary finite lattice $(L, \preceq)$, where the lattice's elements $L \subseteq \{0,1\}^n$ are encoded as binary strings of some given length n, and where we assume the entire lattice $(L, \preceq)$ is known *explicitly* by the querier, who moreover has unbounded computational power, but who only has oracle access to the monotone function $f : L \to L$. In the full version of this paper ([11]), generalizing the $\log^d N$ algorithm for Euclidean grids, we show that in this black box model there is a deterministic algorithm that computes a fixed point of $f : L \to L$ using $O(\log^d(|L|))$ queries to the function f, where d is the *dimension* of the lattice $(L, \preceq)$. The *dimension* of a lattice $(L, \preceq)$, and more generally the dimension of any partial order, can be defined as the smallest integer $d \geq 1$ such that the relation $\preceq$ is the intersection of d total orders on the same underlying set L. Equivalently, it is the smallest $d \geq 1$ such that there is an injective embedding of $(L, \preceq)$ in the Euclidean grid $([|L|]^d, \leq)$, where $\leq$ is the standard coordinate-wise partial order on $[|L|]^d$. Note that a lower bound of $\Omega(\log^2(|L|))$ queries for computing a fixed point of a monotone function $f : L \to L$ in this black box model follows directly from our lower bound of $\Omega(\log^2 N)$ for monotone functions on the 2D grid $f : [N]^2 \to [N]^2$. At present, we do not know any better lower bound than $\Omega(\log^2(|L|))$ in this black box model for arbitrary finite lattices.[8]

References

1 K. J. Arrow and G. Debreu. Existence of an equilibrium for a competitive economy. *Econometrica*, pages 265–290, 1954.

2 S. R. Buss and A. S. Johnson. Propositional proofs and reductions between NP search problems. *Annals of Pure and Applied Logic*, 163:1163–1182, 2012.

3 C.-L. Chang, Y.-D. Lyuu, and Y.-W. Ti. The complexity of Tarski's fixed point theorem. *Theoretical Computer Science*, 401:228–235, 2008.

4 A. Condon. The complexity of stochastic games. *Information and Computation*, 96:203–224, 1992.

5 C. Dang, Q. Qi, and Y. Ye. Computational models and complexities of Tarski's fixed points. Technical report, Stanford University, 2012.

6 C. Dang and Y. Ye. On the complexity of a class of discrete fixed point problems under the lexicographic ordering. Technical Report CY2018-3, City University of Hong Kong, 2018.

7 C. Dang and Y. Ye. On the complexity of an expanded Tarski's fixed point problem under the componentwise ordering. *Theoretical Computer Science*, 732:26–45, 2018.

8 C. Dang and Y. Ye. Personal commuication with the authors, March 2019.

9 C. Daskalakis and C. H. Papadimitriou. Continuous Local Search. In *Proceedings of 22nd ACM-SIAM Symp. on Discrete Algorithms (SODA'11)*, pages 790–804, 2011.

10 F. Echenique. Finding all equilibria in games with strategic complements. *Journal of Economic Theory*, 135(1):514–532, 2007.

11 K. Etessami, C. Papadimitriou, A. Rubinstein, and M. Yannakakis. Tarski's theorem, supermodular games, and the complexity of equilibria. arXiv preprint, 2019. `arXiv:1909.03210`.

12 K. Etessami and M. Yannakakis. On the complexity of Nash equilibria and other fixed points. *SIAM Journal on Computing*, 39(6):2531–2597, 2010.

[8] Note that this black box model is very different from the one considered in [3], where the lattice itself is not known explicitly, but is only accessible via an oracle for its partial order. Hence the linear $\Omega(|L|)$ lower bound on the number of queries (including queries to the partial order itself) given in [3] for finding a fixed point has no bearing on the black box model described here, where the lattice itself is explicitly known, and only the monotone function is given by an oracle.

13 J. Fearnley, S. Gordon, R. Mehta, and R. Savani. End of Potential Line. arXiv preprint, 2018. arXiv:1804.03450.

14 J. Fearnley, S. Gordon, R. Mehta, and R. Savani. Unique End of Potential Line. In *Proc. of 46th Int. Coll. on Automata, Languages, and Programming (ICALP'19)*, page 15 pages, 2019. (Full preprint: arXiv:1811.03841, 90 pages).

15 H. Freudenthal. Simplizialzerlegungen von beschrankter flachheit. *Annals of Mathematics*, 43:580–582, 1942.

16 D. S. Johnson, C. H. Papadimitriou, and M. Yannakakis. How easy is local search? *Journal of Computer and System Sciences*, 37:79–100, 1988.

17 P. Milgrom and J. Roberts. Rationalizability, learning, and equilibrium in games with strategic complementarities. *Econometrica*, pages 1255–1277, 1990.

18 J. Nash. Non-cooperative Games. *Annals of Mathematics*, pages 286–295, 1951.

19 N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani, editors. *Algorithmic Game Theory*. Cambridge University Press, 2007.

20 C. H. Papadimitriou. On the Complexity of the Parity Argument and Other Inefficient Proofs of Existence. *Journal of Computer and System Sciences*, 48(3):498–532, 1994.

21 L. S. Shapley. Stochastic Games. *Proceeding of the National Academy of Sciences*, 39(10):1095–1100, 1953.

22 A. Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5(2):285–309, 1955.

23 D. M. Topkis. Equilibrium Points in Nonzero-Sum n-Person Submodular Games. *SIAM Journal on control and optimization*, 17(6):773–787, 1979.

24 D. M. Topkis. *Supermodularity and Complementarity*. Princeton University Press, 2011.

25 M. Yannakakis. The analysis of local search problems and their heuristics. In *Proc. of Symp. on Theoretical Aspects of Computer Science (STACS'90)*, Springer LNCS 415, pages 298–311, 1990.

26 M. Yannakakis. Chapter 2: Computational Complexity. In E. Aarts and J. K. Lenstra, editors, *Local Search in Combinatorial Optimization*. John Wiley & Sons, 1997.

Resolution with Counting: Dag-Like Lower Bounds and Different Moduli

Fedor Part
JetBrains Research, St. Petersburg, Russia
fedor.part@jetbrains.com

Iddo Tzameret
Department of Computer Science, Royal Holloway, University of London, UK
Iddo.Tzameret@rhul.ac.uk

Abstract

Resolution over linear equations is a natural extension of the popular resolution refutation system, augmented with the ability to carry out basic counting. Denoted $\mathrm{Res}(\mathrm{lin}_R)$, this refutation system operates with disjunctions of linear equations with boolean variables over a ring R, to refute unsatisfiable sets of such disjunctions. Beginning in the work of [26], through the work of [17] which focused on tree-like lower bounds, this refutation system was shown to be fairly strong. Subsequent work (cf. [18, 17, 19, 13]) made it evident that establishing lower bounds against general $\mathrm{Res}(\mathrm{lin}_R)$ refutations is a challenging and interesting task since the system captures a "minimal" extension of resolution with counting gates for which no super-polynomial lower bounds are known to date.

We provide the first super-polynomial size lower bounds on general (dag-like) resolution over linear equations refutations in the large characteristic regime. In particular we prove that the subset-sum principle $1 + x_1 + \cdots + 2^n x_n = 0$ requires refutations of exponential-size over $\mathbb{Q}$. Our proof technique is nontrivial and novel: roughly speaking, we show that under certain conditions every refutation of a subset-sum instance $f = 0$, where f is a linear polynomial over $\mathbb{Q}$, must pass through a fat clause containing an equation $f = \alpha$ for each α in the image of f under boolean assignments. We develop a somewhat different approach to prove exponential lower bounds against tree-like refutations of any subset-sum instance that depends on n variables, hence also separating tree-like from dag-like refutations over the rationals.

We then turn to the finite fields regime, showing that the work of Itsykson and Sokolov [17] who obtained tree-like lower bounds over $\mathbb{F}_2$ can be carried over and extended to every finite field. We establish new lower bounds and separations as follows: (**i**) for every pair of distinct primes p, q, there exist CNF formulas with short tree-like refutations in $\mathrm{Res}(\mathrm{lin}_{\mathbb{F}_p})$ that require exponential-size tree-like $\mathrm{Res}(\mathrm{lin}_{\mathbb{F}_q})$ refutations; (**ii**) random k-CNF formulas require exponential-size tree-like $\mathrm{Res}(\mathrm{lin}_{\mathbb{F}_p})$ refutations, for every prime p and constant k; and (**iii**) exponential-size lower bounds for tree-like $\mathrm{Res}(\mathrm{lin}_{\mathbb{F}})$ refutations of the pigeonhole principle, for *every* field $\mathbb{F}$.

2012 ACM Subject Classification Theory of computation → Proof complexity

Keywords and phrases Proof complexity, concrete lower bounds, resolution, satisfiability, combinatorics

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.19

Related Version `https://eccc.weizmann.ac.il/report/2018/117/`

Acknowledgements We wish to thank Dima Itsykson and Dima Sokolov for very helpful comments concerning this work, and telling us about the lower bound on random k-CNF formulas for tree-like $\mathrm{Res}(\mathrm{lin}_{\mathbb{F}_2})$ that can be achieved using the results of Garlik and Kołodziejczyk. We thank Edward Hirsch for spotting a gap in the initial proof of the dag-like lower bound concerning the use of the contraction.

1 Introduction

The resolution refutation system is among the most prominent and well-studied propositional proof systems, and for good reasons: it is a natural and simple refutation system, that, at least in practice, is capable of being easily automatized. Furthermore, while being non-trivial, it is simple enough to succumb to many lower bound techniques.

Formally, a resolution refutation of an unsatisfiable CNF formula is a sequence of clauses $D_1, \ldots, D_l = \emptyset$, where $\emptyset$ is the empty clause, such that each D_i is either a clause of the CNF or is derived from previous clauses $D_j, D_k, j \leq k < i$ by means of applying the following *resolution rule*: from the clauses $C \vee x$ and $D \vee \neg x$ derive $C \vee D$.

The *tree-like* version of resolution, where every occurrence of a clause in the refutation is used at most once as a premise of a rule, is of particular importance, since it helps us to understand certain kind of satisfiability algorithms known as DPLL algorithms (cf. [23]). DPLL algorithms are simple recursive algorithms for solving SAT that are the basis of successful contemporary SAT-solvers. The transcript of a run of DPLL on an unsatisfiable formula is a decision tree, which can be interpreted as a tree-like resolution refutation. Thus, lower bounds on the size of tree-like resolution refutations imply lower bounds on the run-time of DPLL algorithms (though it is important to clarify that contemporary SAT-solvers utilize more than the strength of tree-like resolution).

In contrast to the apparent practical success of SAT-solvers, a variety of hard instances that require exponential-size refutations have been found for resolution during the years. Many classes of such hard instances are based on principles expressing some sort of counting. One famous example is the *pigeonhole principle*, denoted PHP^m_n, expressing that there is no (total) injective map from a set with cardinality m to a set with cardinality n if $m > n$ [15]. Another important example is *Tseitin tautologies*, denoted TS_G, expressing that the sum of the degrees of vertices in a graph G must be even [28].

Since such counting tautologies are a source of hard instances for resolution, it is useful to study extensions of resolution that can efficiently count, so to speak. This is important firstly, because such systems may become the basis of more efficient SAT-solvers and secondly, in order to extend the frontiers of lower bound techniques against stronger and stronger propositional proof systems. Indeed, there are many works dedicated to the study of weak systems operating with De Morgan formulas with counting connectives; these are variations of resolution that operate with disjunctions of certain arithmetic expressions.

One such extension of resolution was introduced by Raz and Tzameret [26] under the name *resolution over linear equations* in which literals are replaced by linear equations. Specifically, the system R(lin), which operates with disjunctions of linear equations over $\mathbb{Z}$ was studied in [26]. This work demonstrated the power of resolution with counting over the integers, and specifically provided polynomial upper bounds for the pigeonhole principle and the Tseitin formulas, as well as other basic counting formulas. It also established exponential lower bounds for a subsystem of R(lin), denoted R^0(lin). Subsequently, Itsykson and Sokolov [17] studied resolution over linear equations over $\mathbb{F}_2$, denoted Res($\oplus$). They demonstrated the power of resolution with counting mod 2 as well as its limitations by means of several upper bounds and tree-like lower bounds. Moreover, [17] introduced DPLL algorithms, which can "branch" on arbitrary linear forms over $\mathbb{F}_2$, as well as parity decision trees, and showed a correspondence between parity decision trees and tree-like Res($\oplus$) refutations. In both [26] and [17] the dag-like lower bound question for resolution over linear equations remained open.

Apart from being a very natural refutation system, understanding the proof complexity of resolution over linear equations is important for the following reason: proving super-polynomial dag-like lower bounds against resolution over linear equations for prime fields and for the integers can be viewed as a first step towards the long-standing open problems of $\mathsf{AC}^0[p]$-Frege and TC^0-Frege lower bounds, respectively. We explain this in what follows.

Resolution operates with clauses, which are De Morgan formulas ($\neg$, unbounded fan-in $\vee$ and $\wedge$) of a particular kind, namely, of depth 1. Thus, from the perspective of proof complexity, resolution is a fairly weak version of the propositional-calculus, where the latter operates with arbitrary De Morgan formulas. Under a natural and general definition, propositional-calculus systems go under the name *Frege systems*: they can be (axiomatic) Hilbert-style systems or sequent-calculus style systems. A particular choice of the formalism is not important: a classical result by Reckhow [27] assures us that all Frege systems are polynomially equivalent. The task of proving lower bounds for general Frege systems is notoriously hard: no nontrivial lower bounds are known to date. Basically, the strongest fragment of Frege systems, for which lower bounds are known are AC^0-Frege systems, which are Frege proofs operating with constant-depth formulas. For example, both PHP_n^m and TS_G do not admit sub-exponential proofs in AC^0-Frege [1, 24, 20, 7, 16]. However, if we extend the De Morgan language with counting connectives such as unbounded fan-in mod p ($\mathsf{AC}^0[p]$-Frege) or threshold gates (TC^0-Frege), then we step again into the darkness: proving super-polynomial lower bounds for these systems is a long-standing open problem on what can be characterized as the "frontiers" of proof complexity. Recent works by Krajíček [18], Garlik-Kołodziejczyk [13] and Krajíček-Oliveira [19] had suggested possible approaches to attack dag-like $\mathrm{Res}(\mathrm{lin}_{\mathbb{F}_2})$ lower bounds (though this problem remains open to date).

1.1 Our Results and Techniques

In this work we prove a host of new lower bounds, separations and upper bounds for resolution over linear equations. Our main novel technical contribution is a dag-like refutation lower bound over large characteristic fields. Conceptually, the proof idea exploits two main properties that recently have been found useful in proof complexity:

(i) Single axiom: the hard instance consists of a single unsatisfiable axiom (for boolean assignments)

$$1 + x_1 + \cdots + 2^n x_n = 0 \tag{1}$$

(unlike, for instance, a set of clauses).

(ii) Large coefficients: the hard instance uses coefficients of exponential magnitude.

Although employing different approaches, both of these properties played a recent role in proof complexity lower bounds. Forbes et al. [12] used subset-sum variants (that is, unsatisfiable linear equations with boolean variables) to establish lower bounds on subsystems of the ideal proof system (IPS) over large characteristic fields, where IPS is the strong proof system introduced by Grochow and Pitassi [14]. It is essential in both [12] and our work that the hard instance takes the form of a single unsatisfiable axiom. Subsequently, in a very recent work, Alekseev et al. [3] established conditional exponential-size lower bounds on full IPS refutations over the rationals of the same subset-sum instance (1), where the use of big coefficients is again essential to the lower bound. We explain our dag-like lower bound in Section 1.1.2.

The other novel contribution we make is a systematic development of new kinds of lower bound techniques against *tree-like* resolution over linear equations, both over the rationals and over finite fields. To this end we develop new and extend existing combinatorial techniques such as the Prover-Delayer game method as originated in Pudlak and Impagliazzo [25] for resolution, and developed further by Itsykson and Sokolov [17]. Moreover, we provide new applications in proof complexity of different combinatorial results; this include bounds on the size of essential coverings of the hypercube from Linial and Radhakrishnan [21], a result about the hyperplane coverings of the hypercube by Alon and Füredi [4] and the notion of immunity from Alekhnovich and Razborov [2]. We further non-trivially extend the well-established principle of size-width tradeoffs in resolution [8] to the setting of Res(lin_R) (though it is important to note that most of our lower bounds do not follow from this tradeoff result).

1.1.1 Background

For a ring R, the refutation system Res(lin_R) is defined as an extension of the resolution refutation system as follows (see Raz and Tzameret [26]). The *proof-lines* of Res(lin_R) are called ***linear clauses*** (sometimes called simply *clauses*), which are defined as disjunctions of linear equations (with duplicate equations contracted). More formally, they are disjunctions of the form:

$$\left(\sum_{i=1}^{n} a_{1i}x_i + b_1 = 0\right) \vee \cdots \vee \left(\sum_{i=1}^{n} a_{ki}x_i + b_k = 0\right),$$

where k is some number (the *width* of the clause), and $a_{ji}, b_j \in R$. The *resolution rule* is the following:

$$\text{from } (C \vee f = 0) \text{ and } (D \vee g = 0) \text{ derive } (C \vee D \vee (\alpha f + \beta g) = 0),$$

where $\alpha, \beta \in R$, and where C, D are linear clauses. A Res(lin_R) *refutation* of an unsatisfiable over 0-1 set of linear clauses $C_1, \ldots, C_m$ is a sequence of proof-lines, where each proof-line is either C_i, for $i \in [m]$, a boolean axiom $(x_i = 0 \vee x_i = 1)$ for some variable x_i, or was derived from previous proof-lines by the above resolution rule, or by the *weakening rule* that allows to extend clauses with arbitrary disjuncts, or a *simplification rule* allowing to discard false constant linear forms (e.g., $1 = 0$) from a linear clause. The last proof-line in a refutation is the empty clause (standing for the truth value false).

The ***size*** of a Res(lin_R) refutation is the total size of all the clauses in the derivation, where the size of a clause is defined to be the total number of occurrences of variables in it plus the total size of all the coefficient occurring in the clause. The size of a coefficient when using integers (or integers embedded in characteristic zero rings) is the standard size of the binary representation of integers (nevertheless, when we talk about "big" or "exponential" coefficients and "polynomially bounded" coefficients, etc., we mean that the *magnitude* of the coefficients is big (exponential) or polynomially bounded).

We are generally interested in the following questions:

(Q1) For a given ring R, what kind of counting can be efficiently performed in Res(lin_R) and tree-like Res(lin_R)?

(Q2) Can dag-like Res(lin_R) be separated from tree-like Res(lin_R)?

(Q3) Can tree-like systems for different rings R be separated?

1.1.1.1 Tree-like Res(lin_R) with semantic weakening

In order to be able to do some non-trivial counting in *tree-like* versions of resolution over linear equations we define a semantic version of the system as follows.

The system $\mathrm{Res}_{sw}(\mathrm{lin}_R)$ is obtained from $\mathrm{Res}(\mathrm{lin}_R)$ by replacing the weakening and the simplification rules, as well as the boolean axioms, with the *semantic weakening* rule (the symbol $\models$ will denote in this work semantic implication *with respect to 0-1 assignments*):[1]

$$\frac{C}{D}\,(C \models D)\,.$$

The reason for studying $\mathrm{Res}_{sw}(\mathrm{lin}_R)$ is mainly the following: Let Γ be an arbitrary set of tautological R-linear clauses. Then, lower bounds for tree-like $\mathrm{Res}_{sw}(\mathrm{lin}_R)$ imply lower bounds for tree-like $\mathrm{Res}(\mathrm{lin}_R)$ with formulas in Γ as axioms. For example, in case $\mathbb{F}$ is a field of characteristic 0, the possibility to do counting in tree-like $\mathrm{Res}(\mathrm{lin}_{\mathbb{F}})$ is quite limited. For instance, we show that $2x_1 + \cdots + 2x_n = 1$ requires refutations of exponential in n size (Theorem 35). On the other hand, such contradictions *do* admit short tree-like $\mathrm{Res}(\mathrm{lin}_{\mathbb{F}})$ refutations in the presence of the following *generalized boolean axioms* (which is a tautological linear clause):

$$\mathsf{Im}(f) := \bigvee_{A \in im_2(f)} (f = A), \tag{2}$$

where $im_2(f)$ is the image of a linear polynomial f under 0-1 assignments. Similar to the way the boolean axioms $(x_i = 0) \vee (x_i = 1)$ state that the possible value of a variable is either zero or one, the $\mathsf{Im}(f)$ axiom states all the possible values that the linear form f can have. If a lower bound holds for tree-like $\mathrm{Res}_{sw}(\mathrm{lin}_{\mathbb{F}})$ it also holds, in particular, for tree-like $\mathrm{Res}(\mathrm{lin}_{\mathbb{F}})$ with the axioms $\mathsf{Im}(f)$, and this makes tree-like $\mathrm{Res}_{sw}(\mathrm{lin}_{\mathbb{F}})$ a useful system, for which lower bounds against are sufficiently interesting.

1.1.2 Characteristic Zero Lower Bounds

For characteristic zero fields we will use mainly the rational number field $\mathbb{Q}$ (though many of the results hold over any characteristic zero rings). First, we show that over $\mathbb{Q}$, whenever $\alpha_1 x_1 + \cdots + \alpha_n x_n + \beta = 0$ is unsatisfiable (over 0-1 assignments), it has polynomial dag-like $\mathrm{Res}(\mathrm{lin}_{\mathbb{Q}})$ refutations if the coefficients are polynomially bounded in magnitude, while it requires exponential dag-like $\mathrm{Res}(\mathrm{lin}_{\mathbb{Q}})$ refutations for some subset-sum instances with exponential-magnitude coefficients. Note that $\alpha_1 x_1 + \cdots + \alpha_n x_n + \beta = 0$ expresses the *subset-sum principle*: $\alpha_1 x_1 + \cdots + \alpha_n x_n = -\beta$ is satisfiable iff there is a subset of the integral coefficients α_i whose sum is precisely $-\beta$. The lower bound is stated in the following theorem:

▶ **Theorem** (Theorem 21; Main dag-like lower bound). *Any* $\mathrm{Res}(\mathrm{lin}_{\mathbb{Q}})$ *refutation of* $x_1 + 2x_2 + \cdots + 2^n x_n + 1 = 0$ *requires size* $2^{\Omega(n)}$.

The proof of this theorem introduces a new lower bound technique. We show that every (dag- or tree-like) refutation π of $x_1 + 2x_2 + \cdots + 2^n x_n + 1 = 0$ can be transformed without much increase in size into a derivation of a certain "fat" (exponential-size) clause C_π from boolean axioms only.[2] In order to prove that C_π is fat, we ensure that every disjunct $g = 0$

[1] Let $k = char(R)$ be the characteristic of the ring R. In case $k \notin \{1, 2, 3\}$, deciding whether an R-linear clause D is a tautology (that is, holds for every 0-1 assignment to its variables) is at least as hard as deciding whether a 3-DNF is a tautology (because over characteristic $k \notin \{1, 2, 3\}$ linear equations can express conjunction of three conjuncts). For this reason $\mathrm{Res}_{sw}(\mathrm{lin}_R)$ proofs cannot be checked in polynomial time and thus $\mathrm{Res}_{sw}(\mathrm{lin}_R)$ is not a Cook-Reckhow proof system unless $\mathsf{P} = \mathsf{coNP}$ (namely, the correctness of proofs in the system cannot necessarily be checked in polynomial-time, as required by a Cook-Reckhow propositional proof system [11]; see Section 2.2).

[2] The notion of showing that a refutation must go though a fat (i.e., wide) clause is well established in resolution lower bounds. However, we note that our lower bound is completely different from the known size-width based resolution lower bounds (as formulated in a generic way in the work of Ben-Sasson and Wigderson [8]).

in C_π has at most 2^{cn} satisfying boolean assignments, for some constant $c < 1$. Because C_π is derived from boolean axioms alone, it must be a boolean tautology, that is, it must have 2^n satisfying assignment. Since every disjunct in C_π is satisfied by at most 2^{cn} assignments, the number of disjuncts in the clause is at least $2^{(1-c)n}$. Since our constructed derivation is not much larger than the original refutation, the size of the original refutation must be $2^{\Omega(n)}$.

This proof relies in an essential way on the fact that the coefficients of the linear form have exponential magnitude. Indeed, every contradiction of the form $f = 0$ can be shown to admit polynomial-size dag-like Res($\text{lin}_\mathbb{Q}$) refutations whenever the coefficients of f are polynomially bounded. A natural question is whether in the case of bounded coefficients, $f = 0$ can be efficiently refuted already by tree-like Res($\text{lin}_\mathbb{Q}$) refutations. The question turns out to be non-trivial, and we provide a negative answer:

▶ **Theorem** (Theorem 35; Subset-sum tree-like lower bounds). *Let f be any linear polynomial over $\mathbb{Q}$, which depends on n variables. Then tree-like* Res($\text{lin}_\mathbb{Q}$) *refutations of $f = 0$ are of size $2^{\Omega(\sqrt{n})}$.*

The proof is in two stages. First, we use a transformation analogous to the one used for the dag-like lower bound to reduce the lower bound problem for refutations of $f = 0$ to a lower bound problem for derivations of clauses of a certain kind. Namely, we transform any tree-like refutation π of $f = 0$ to a tree-like derivation of C_π from boolean axioms without much increase in size. The only difference is that this time we ensure that in every disjunct $g = 0$ of C_π, the linear polynomial g depends on at least $\frac{n}{2}$ variables.

Second, we prove that tree-like Res($\text{lin}_\mathbb{Q}$) derivations of such a C_π are large:

▶ **Theorem** (Theorem 33). *Any tree-like* Res($\text{lin}_\mathbb{Q}$) *derivation of any tautology of the form $\bigvee_{j\in[N]} g_j = 0$, for some positive N, where each g_j is linear over $\mathbb{Q}$ and depends on at least $\frac{n}{2}$ variables, is of size $2^{\Omega(\sqrt{n})}$.*

To prove this, as well as some other lower bounds, we extend the Prover-Delayer game technique as originated in Pudlak-Impagliazzo [25] for resolution, and developed further by Itsykson-Sokolov [17] for Res($\text{lin}_{\mathbb{F}_2}$), to general rings, including characteristic zero rings (see Sec. 5.2).[3]

We define a non-trivial strategy for Delayer in the corresponding game and prove that it guarantees $\sqrt{n}$ coins using a bound on the size of essential coverings of the hypercube from Linial and Radhakrishnan [21]. The relation between Prover-Delayer games and tree-like Res($\text{lin}_\mathbb{Q}$) refutations allows us to conclude that the size of tree-like Res($\text{lin}_\mathbb{Q}$) refutations must be $2^{\Omega(\sqrt{n})}$.

Moreover, as a corollary of Theorem 33 we obtain a lower bound on tree-like Res($\text{lin}_\mathbb{Q}$) *derivations* (in contrast to refutations) of $\mathsf{Im}(f)$:

▶ **Corollary** (Corollary 34). *Let f be any linear polynomial over $\mathbb{Q}$ that depends on n variables. Then tree-like* Res($\text{lin}_\mathbb{Q}$) *derivations of $\mathsf{Im}(f)$ are of size $2^{\Omega(\sqrt{n})}$.*

We also use Prover-Delayer games to prove an exponential-size $2^{\Omega(n)}$ lower bound on tree-like Res$_{sw}$($\text{lin}_\mathbb{F}$) refutations of the pigeonhole principle PHP^m_n *for every field* $\mathbb{F}$ (including finite fields). This extends a previous result by Itsykson and Sokolov [17] for tree-like Res($\text{lin}_{\mathbb{F}_2}$).

[3] We note here (see Remark 1 in the next sub-section) that the lower bounds that we prove using Prover-Delayer games techniques in case $char(\mathbb{F}) = 0$ *do not* follow from lower bounds for Polynomial Calculus using size-width relations.

▶ **Theorem** (Theorem 38; Pigeonhole principle lower bounds). *Let $\mathbb{F}$ be any (possibly finite) field. Then every tree-like* $\mathrm{Res}_{sw}(\mathrm{lin}_{\mathbb{F}})$ *refutation of* $\neg\mathrm{PHP}_n^m$ *has size* $2^{\Omega\left(\frac{n-1}{2}\right)}$.

Together with the polynomial upper bounds for PHP_n^m refutations in dag-like $\mathrm{Res}(\mathrm{lin}_{\mathbb{F}})$ for fields $\mathbb{F}$ of characteristic zero demonstrated by Raz and Tzameret [26], Theorem 38 establishes a *separation between dag-like* $\mathrm{Res}(\mathrm{lin}_{\mathbb{F}})$ *and tree-like* $\mathrm{Res}_{sw}(\mathrm{lin}_{\mathbb{F}})$ for characteristic zero fields, for the language of unsatisfiable formulas in CNF:

▶ **Corollary.** *Over fields of characteristic zero $\mathbb{F}$,* $\mathrm{Res}(\mathrm{lin}_{\mathbb{F}})$ *has an exponential speed-up over tree-like* $\mathrm{Res}(\mathrm{lin}_{\mathbb{F}})$ *as refutation systems for unsatisfiable formulas in CNF.*

To prove Theorem 38 we need to prove that Delayer's strategy from [17] is successful over any field. This argument is new, and uses a result of Alon-Füredi [4] about the hyperplane coverings of the hypercube.

We prove another separation between dag-like $\mathrm{Res}(\mathrm{lin}_{\mathbb{Q}})$ and tree-like $\mathrm{Res}_{sw}(\mathrm{lin}_{\mathbb{Q}})$, as follows. For any ring R we define the *image avoidance principle* to be:

$$\mathsf{ImAv}\left(x_1+\cdots+x_n\right) := \left\{\langle x_1+\cdots+x_n \neq k\rangle\right\}_{k\in\{0,\ldots,n\}},$$

where $\langle x_1+\cdots+x_n \neq k\rangle := \bigvee_{k'\in\{0,\ldots,n\},\ k\neq k'} x_1+\cdots+x_n = k'$. In words, the image avoidance principle expresses the contradictory statement that for every $0 \leq i \leq n$, $x_1+\cdots+x_n$ equals some element in $\{0,\ldots,n\}\setminus i$. In more generality, let f be a linear form over $\mathbb{Q}$ and let $im_2(f)$ be the image of f under 0-1 assignments to its variables. Define $\langle f \neq A\rangle := \bigvee_{A\neq B\in im_2(f)}(f=B)$, where $A\in\mathbb{Q}$. We define

$$\mathsf{ImAv}\left(f\right) := \left\{\langle f\neq A\rangle : A\in im_2(f)\right\}. \tag{3}$$

▶ **Corollary** (Corollary 13). *For every ring R and every linear form f the contradiction* $\mathsf{ImAv}\left(f\right)$ *admits polynomial-size* $\mathrm{Res}(\mathrm{lin}_R)$ *refutations.*

▶ **Theorem** (Theorem 37). *We work over $\mathbb{Q}$. Let $f=\epsilon_1x_1+\cdots+\epsilon_nx_n$, where $\epsilon_i\in\{-1,1\}$. Then any tree-like* $\mathrm{Res}_{sw}(\mathrm{lin}_{\mathbb{Q}})$ *refutation of* $\mathsf{ImAv}\left(f\right)$ *is of size at least* $2^{\frac{n}{4}}$.

The lower bound in Theorem 37 is one more novel application of the Prover-Delayer game argument, combined with the notion of immunity from Alekhnovich and Razborov [2], as we now briefly explain.

Let f be a linear form as in Theorem 37. We consider an instance of the Prover-Delayer game for $\mathsf{ImAv}\left(f\right)$. A position in the game is determined by a *set Φ of linear non-equalities* of the form $g\neq 0$, which we think of as the set of non-equalities learned up to this point by Prover. In the beginning Φ is empty. We define Delayer's strategy in such a way that for Φ an end-game position, there is a satisfiable subset $\Phi'=\{g_1\neq 0,\ldots,g_m\neq 0\}\subseteq\Phi$ such that $\Phi'\models f=A$ for some $A\in\mathbb{F}$, and Delayer earns at least $|\Phi'|=m$ coins. Because $\mathbb{F}$ is of characteristic zero, it follows that $f\equiv A+1 \pmod 2$ $\models f\neq A\models g_1\cdot\ldots\cdot g_m=0$ and thus the $\frac{n}{4}$-immunity of $f\equiv A+1 (\mathrm{mod}\ 2)$ ([2]) implies $m\geq\frac{n}{4}$. To conclude, by a standard argument if Delayer always earns $\frac{n}{4}$ coins, then the shortest proof is of size at least $2^{\frac{n}{4}}$.

Table 1 sums up our knowledge up to this point with respect to $\mathbb{Q}$ (and for some cases any characteristic 0 field):

1.1.3 Finite Fields Lower Bounds

We now turn to resolution over linear equations in *finite fields.* We obtain many new tree-like lower bounds (see Table 2).

Table 1 Lower and upper bounds for $\mathbb{Q}$. The notation t-l Res(lin_R) stands for tree-like Res(lin_R). The rightmost column describes bounds on *derivations*, in contrast to refutations. All results except the upper bound on PHP are from the current work.

	$\sum_{i=1}^{n} 2x_i = 1$	$\sum_{i=1}^{n} 2^i x_i = -1$	$\mathsf{ImAv}\left(\sum_{i=1}^{n} x_i\right)$	PHP_n^m (CNF)	$\mathsf{Im}\left(\sum_{i=1}^{n} x_i\right)$
t-l Res($\text{lin}_{\mathbb{Q}}$)	$2^{\Omega(\sqrt{n})}$	$2^{\Omega(n)}$	$2^{\Omega(n)}$	$2^{\Omega(n)}$	$2^{\Omega(\sqrt{n})}$
t-l Res$_{sw}$($\text{lin}_{\mathbb{Q}}$)	poly	poly	$2^{\Omega(n)}$	$2^{\Omega(n)}$	poly
Res($\text{lin}_{\mathbb{Q}}$)	poly	$2^{\Omega(n)}$	poly	poly [26]	poly

We already discussed above lower bounds for the pigeonhole principle which hold both for positive and zero characteristic. We furthermore prove a separation between tree-like Res($\text{lin}_{\mathbb{F}_{p^k}}$) (resp. tree-like Res$_{sw}$($\text{lin}_{\mathbb{F}_{p^k}}$)) and tree-like Res($\text{lin}_{\mathbb{F}_{q^l}}$) (resp. tree-like Res$_{sw}$($\text{lin}_{\mathbb{F}_{q^l}}$)) for every pair of distinct primes $p \neq q$ and every $k, l \in \mathbb{N} \setminus \{0\}$. The separating instances are mod p Tseitin formulas $\text{TS}^{(p)}_{G,\sigma}$ (written as CNFs), which are reformulations of the standard Tseitin graph formulas TS_G for counting mod p. Furthermore, we establish an exponential lower bound for tree-like Res$_{sw}$($\text{lin}_{\mathbb{F}_{p^c}}$) on random k-CNFs.[4]

The lower bounds for tree-like Res($\text{lin}_{\mathbb{F}}$) for finite fields $\mathbb{F}$ are obtained via a variant of the size-width relation for tree-like Res($\text{lin}_{\mathbb{F}}$) together with a translation to polynomial calculus over the field $\mathbb{F}$, denoted $PC_{\mathbb{F}}$ [10], such that Res($\text{lin}_{\mathbb{F}}$) proofs of width ω are translated to $PC_{\mathbb{F}}$ proofs of degree ω (the *width* ω of a clause is defined to be the total number of disjuncts in a clause). This establishes the lower bounds for the size of tree-like Res($\text{lin}_{\mathbb{F}}$) proofs via known lower bounds on $PC_{\mathbb{F}}$ degrees ([2]).

We show that

$$\omega_0(\phi \vdash \bot) = O\left(\omega_0(\phi) + \log S_{\text{t-l Res}(\text{lin}_R)}(\phi \vdash \bot)\right),$$

where ω_0 is what we call the *principal width*, which counts the number of linear equations in clauses when we treat as identical those defining parallel hyperplanes, and $S_{\text{t-l Res}(\text{lin}_R)}(\phi \vdash \bot)$ denotes the minimal size of a tree-like Res(lin_R) refutation of ϕ.

Specifically, over finite fields the following upper and lower bounds provide exponential separations:

▶ **Theorem** (Theorem 44; Size-width relation). *Let ϕ be an unsatisfiable set of linear clauses over a field $\mathbb{F}$. The following relation between principal width and size holds for both tree-like* Res($\text{lin}_{\mathbb{F}}$) *and tree-like* Res$_{sw}$($\text{lin}_{\mathbb{F}}$)*: $S(\phi \vdash \bot) = 2^{\Omega(\omega_0(\phi \vdash \bot) - \omega_0(\phi))}$. If $\mathbb{F}$ is a finite field, then the same relation holds for the (standard) width of a clause ω.*

This extends to every field a result by Garlik-Kołodziejczyk [13, Theorem 14] who showed a size-width relation for a system denoted tree-like $\text{PK}^{\text{id}}_{O(1)}(\oplus)$, which is a system extending tree-like Res($\text{lin}_{\mathbb{F}_2}$) by allowing arbitrary constant-depth De Morgan formulas as inputs to $\oplus$ (XOR gates) (though note that our result does not deal with *arbitrary* constant-depth formulas).

▶ **Theorem** (Theorem 45). *Let $\mathbb{F}$ be a field and π be a* Res($\text{lin}_{\mathbb{F}}$) *refutation of an unsatisfiable CNF formula ϕ. Then, there exists a $PC_{\mathbb{F}}$ refutation π' of (the arithmetization of) ϕ of degree $\omega(\pi)$.*

[4] We thank Dmitry Itsykson for telling us about the lower bound for random k-CNF for the case of tree-like Res($\text{lin}_{\mathbb{F}_2}$), that was proved by Garlik and Kołodziejczyk using size-width relations (unpublished note). Our result extends Garlik and Kołodziejczyk's result to all finite fields. Similar to their result, we use a size-width argument and simulation by the polynomial calculus to establish the lower bound.

▶ **Corollary** (Corollary 46; Tseitin mod p lower bounds). *For any fixed prime p there exists a constant $d_0 = d_0(p)$ such that the following holds. If $d \geq d_0$, G is a d-regular directed graph satisfying certain expansion properties, and $\mathbb{F}$ is a finite field such that $char(\mathbb{F}) \neq p$, then every tree-like* $\mathrm{Res}(\mathrm{lin}_{\mathbb{F}})$ *refutation of the Tseitin mod p formula $\neg \mathrm{TS}^{(p)}_{G,\sigma}$ has size $2^{\Omega(dn)}$.*

▶ **Corollary** (Corollary 47; Random k-CNF formulas lower bounds). *Let ϕ be a randomly generated k-CNF with clause-variable ratio Δ, and where $\Delta = \Delta(n)$ is such that $\Delta = o\left(n^{\frac{k-2}{2}}\right)$, and let $\mathbb{F}$ be a finite field. Then, every tree-like* $\mathrm{Res}(\mathrm{lin}_{\mathbb{F}})$ *refutation of ϕ has size $2^{\Omega\left(\frac{n}{\Delta^{2/(k-2)} \cdot \log \Delta}\right)}$ with probability $1 - o(1)$.*

▶ Remark 1. We stress that the size-width relation of Theorem 44 **cannot** be used for transferring $PC_{\mathbb{F}}$ degree lower bounds to tree-like $\mathrm{Res}(\mathrm{lin}_{\mathbb{F}})$ size lower bounds in case $char(\mathbb{F}) = 0$. This is due to the essential difference between principal width and width in this case. Thus, all the lower bounds that we prove using Prover-Delayer games techniques in case $char(\mathbb{F}) = 0$ **do not** follow from lower bounds for $PC_{\mathbb{F}}$.

Table 2 shows the results for $\mathrm{Res}(\mathrm{lin}_R)$ over finite fields.

Table 2 Lower bounds over finite fields. Here G is d-regular graph and Δ is the clause density (number of clauses divided by the number of variables), $A\overline{x} = \overline{b}$ stands for a linear system over $\mathbb{F}_{p^k}$ that has no 0-1 solutions in the first and the third rows, and in the second row the linear system $A\overline{x} = \overline{b}$ is over $\mathbb{F}_2$. The notation $\mathrm{TS}^{(-)}_{G,\sigma}$ stands for $\mathrm{TS}^{(p)}_{G,\sigma}$ in the first and the third rows and for $\mathrm{TS}^{(2)}_{G,\sigma}$ in the second row. t-l $\mathrm{Res}(\mathrm{lin}_R)$ stands for tree-like $\mathrm{Res}(\mathrm{lin}_R)$, and $p \neq q$ are primes (in the second row and third column we assume $q \neq 2$). Circled "?" denotes an open problem. The results marked with [17, 13] were proved in the respective papers. All other results are from the current work.

	$A\overline{x} = \overline{b}$	$\mathrm{TS}^{(-)}_{G,\sigma}$	$\mathrm{TS}^{(q)}_{G,\sigma}$	random k-CNF	PHP^m_n
t-l $\mathrm{Res}(\mathrm{lin}_{\mathbb{F}_{p^k}})$	$2^{\Omega(n)}$	poly	$2^{\Omega(dn)}$	$2^{\Omega\left(\frac{n}{\Delta^{2/(k-2)} \cdot \log \Delta}\right)}$	$2^{\Omega(n)}$
t-l $\mathrm{Res}(\oplus)$	poly [17]	poly [17]	$2^{\Omega(dn)}$	$2^{\Omega\left(\frac{n}{\Delta^{2/(k-2)} \cdot \log \Delta}\right)}$ [13]	$2^{\Omega(n)}$ [17]
t-l $\mathrm{Res}_{sw}(\mathrm{lin}_{\mathbb{F}_{p^k}})$	poly	poly	?	?	$2^{\Omega(n)}$

1.1.4 Complexity of Linear Systems

The tree-like $\mathrm{Res}(\mathrm{lin}_{\mathbb{F}})$ upper bounds for mod p Tseitin formulas in the case $char(\mathbb{F}) = p$ stem from the following proposition:

▶ **Proposition** (Proposition 14; Upper bounds on unsatisfiable linear systems). *Let $\mathbb{F}$ be a field and assume that the linear system $A\,\overline{x} = \overline{b}$, where A is a $k \times n$ matrix over $\mathbb{F}$, has no solutions (over $\mathbb{F}$). Let ϕ be a CNF formula encoding the linear system $A\,\overline{x} = \overline{b}$. Then, there exist tree-like* $\mathrm{Res}(\mathrm{lin}_{\mathbb{F}})$ *refutations of ϕ of size polynomial in the sum of sizes of encodings of all coefficients in A.*

The upper bound in Proposition 14 applies only to linear systems that are unsatisfiable over the *whole* field $\mathbb{F}$. But does any system $A\,\overline{x} = \overline{b}$ over $\mathbb{F}$ that has a satisfying assignment over $\mathbb{F}$, but *not* over 0-1 assignments, admit polynomial-size $\mathrm{Res}(\mathrm{lin}_{\mathbb{F}})$ refutations?

For fields $\mathbb{F}$ with $char(\mathbb{F}) \geq 5$ or $char(\mathbb{F}) = 0$ it is known that 0-1 satisfiability of $A\,\overline{x} = \overline{b}$ is NP-complete. This means that unless coNP = NP there exist 0-1 unsatisfiable linear systems that require superpolynomial dag-like Res($\text{lin}_{\mathbb{F}}$) refutations.

If $char(\mathbb{F}) \geq k+1$ or $char(\mathbb{F}) = 0$, the canonical reduction R from the language k-UNSAT of unsatisfiable k-CNFs maps every $\phi(\overline{x}) \in k$-UNSAT to the system $R_\phi(\overline{x}, \overline{y})$ by encoding every clause in $\phi(\overline{x})$ as a linear equality with extra variables. This simple reduction allows to establish tight connections between proof complexity of CNF formulas and linear systems.

Firstly, lower bounds on $R_\phi(\overline{x}, \overline{y})$ imply lower bounds on $\phi(\overline{x})$: by implicational completeness there are polynomial-size derivations of $\phi(\overline{x})$ from $R_\phi(\overline{x}, \overline{y})$ in Res($\text{lin}_{\mathbb{F}}$).

Secondly, if $\mathbb{F}$ is a finite field, tree-like Res($\text{lin}_{\mathbb{F}}$) lower bounds on $\phi(\overline{x})$ imply tree-like Res($\text{lin}_{\mathbb{F}}$) lower bounds on $R_\phi(\overline{x}, \overline{y})$. Each linear equation $l(\overline{x}, \overline{y}) = 0$ in $R_\phi(\overline{x}, \overline{y})$ is equivalent to a polynomial equation $l(\overline{x}, \overline{p}(\overline{x})) = 0$, where $\overline{p}$ are polynomials of constant degree. Therefore, there is a constant degree $PC_{\mathbb{F}}$ derivation π_ϕ of $R_\phi(\overline{x}, \overline{p}(\overline{x}))$ from $\phi(\overline{x})$ and vice versa. As any $PC_{\mathbb{F}}$ refutation of $R_\phi(\overline{x}, \overline{y})$ can be turned into a refutation of $R_\phi(\overline{x}, \overline{p}(\overline{x}))$ by substitution without much loss in degree, it is easy to see that $PC_{\mathbb{F}}$ refutes $R_\phi(\overline{x}, \overline{y})$ in degree d iff $PC_{\mathbb{F}}$ refutes $\phi(\overline{x})$ in degree $\Theta(d)$. By size-width relation for finite fields (Theorem 44), we obtain that for any formula $\phi(\overline{x})$ that is hard for $PC_{\mathbb{F}}$, $R_\phi(\overline{x}, \overline{y})$ is hard for tree-like Res($\text{lin}_{\mathbb{F}}$).

1.1.5 Nondeterministic Linear Decision Trees

There is a well-known size preserving (up to a constant factor) correspondence between tree-like resolution refutations for unsatisfiable formulas ϕ and decision trees, which solve the following problem: given an assignment ρ for the variables of ϕ, determine which clause $C \in \phi$ is falsified by querying values of the variables under the assignment ρ. In Itsykson-Sokolov [17] this correspondence was generalized to tree-like Res($\oplus$) refutations and parity decision trees. In the paper by Beame et al. [5] an analogous correspondence was shown for tree-like R(CP) refutations [5] and decision trees that brunch on linear inequalities. In the current work we initiate the study of linear decision trees and their properties over different characteristics, extending the correspondence of [17] to a correspondence between tree-like Res(lin_R) (and tree-like Res$_{sw}$(lin_R)) derivations to what we call *nondeterministic linear decision trees* (NLDT).

NLDTs for an unsatisfiable set of linear clauses ϕ are binary rooted trees, where every edge is labeled with a non-equality $f \neq 0$ for a linear form f and every leaf is labeled with a linear clause $C \in \phi$, which is violated by the non-equalities on the path from the root to the leaf. (Note that in the same manner that in a (boolean) decision tree (which corresponds to a tree-like resolution refutation) we go along a path from the root to a leaf, choosing those edges that violate a literal x_i or $\neg x_i$, in an NLDT we branch along a path that violates equalities $f = 0$, or equivalently, certifies non-equalities of the form $f \neq 0$.)

2 Preliminaries

2.1 Notation

Denote by $[n]$ the set $\{1, \ldots, n\}$. We use $x_1, x_2, \ldots$ to denote variables, both propositional and algebraic. Let f be a linear polynomial (equivalently, an affine function) over a ring R, that is, a function of the form $\sum_{i=1}^{n} a_i x_i + a_0$ with $a_i \in R$. We sometimes refer to a linear

[5] R(CP) is a system operating with disjunctions of integer linear inequalities $f \geq 0$

form as a *hyperplane*, since a linear form determines a hyperplane. We denote by $im_2(f)$ the image of f under 0-1 assignments to its variables; $\langle f \neq A\rangle := \bigvee_{A\neq B\in im_2(f)}(f = B)$, where $A \in R$.

A *linear clause* is a formula of the form $(\sum_{i=1}^{n} a_{1i}x_i + b_1 = 0)\vee\cdots\vee(\sum_{i=1}^{n} a_{ki}x_i + b_k = 0)$ with $x_1,\ldots,x_n$ variables, and a_{ij}, b_i's ring elements (when the ring is specified in advanced). We sometimes abuse notation by writing a linear equation as $\sum_{i=0}^{n} a_{1i}x_i = -b_1$ instead of $\sum_{i=0}^{n} a_{1i}x_i + b_1 = 0$. We assume that all the disjuncts in a linear clause are distinct.

For ϕ a set of clauses or linear clauses, $vars(\phi)$ denotes the set of variables occurring in ϕ and let Vars denote the set of *all* variables.

Let A be a matrix over a ring. We introduce the notation $Ax \doteqdot b$ for a system of linear non-equalities, where a ***non-equality*** means $\neq$ (note the difference between $Ax \doteqdot b$, which stands for $A_i \cdot x \neq b_i$, for *all* rows A_i in A, and $Ax \neq b$, which stands for $A_i \cdot x \neq b_i$, for *some* row A_i in A).

If f is a linear polynomial over R and A is a matrix over R, denote by $|f|$ the sum of sizes of encodings of coefficients in f and by $|A|$ the sum of sizes of encodings of elements in A.

If $C = (\bigvee_{i\in[m]} f_i = 0)$ is a linear clause, denote by $\neg C$ the *set* of non-equalities $\{f_i \neq 0\}_{i\in[m]}$. Conversely, if $\Phi = \{f_i \neq 0\}_{i\in[n]}$ is a set of non-equalities, denote $\neg\Phi := \bigvee_{i\in[m]} f_i = 0$.

If ϕ is a set of linear clauses over a ring R and D is a linear clause over R, denote by $\bigwedge_{C\in\phi} C \models D$ and $\bigwedge_{C\in\phi} C \models_R D$ semantic entailment over 0-1 and R-valued assignments respectively.

Let l be a linear polynomial not containing the variable x. If C is a linear clause, denote by $C\restriction_{x\leftarrow l}$ the linear clause, which is obtained from C by substituting l for x everywhere in C. If $\phi = \{C_i\}_{i\in I}$ is a set of clauses, denote $\phi\restriction_{x\leftarrow l} := \{C_i\restriction_{x\leftarrow l}\}_{i\in I}$. We define a *linear substitution* ρ to be a sequence $(x_1 \leftarrow l_1,\ldots,x_n \leftarrow l_n)$ such that each linear polynomial l_i does not depend on x_i. For a clause or a set of clauses ϕ we define $\phi\restriction_\rho := (\ldots((\phi\restriction_{x_1\leftarrow l_1})\restriction_{x_2\leftarrow l_2})\ldots)\restriction_{x_n\leftarrow l_n}$.

Denote UNSAT $\subset \{0,1\}^*$ (resp. k-UNSAT $\subset \{0,1\}^*$) the language of unsatisfiable propositional CNF (resp. k-CNF) formulas. Denote by $S(\pi)$, and alternatively by $|\pi|$, the size of the binary encoding of a proof π in a proof system Π. For $\phi \in$ UNSAT and a refutation system Π denote by $S_\Pi(\phi \vdash \bot)$ (we sometimes omit the subscript Π when it is clear from the context) the minimal size of a Π-refutation of ϕ.

2.2 Propositional Proof Systems

The *resolution* system (which we denote also by Res) is a refutation system, based on the following rule, allowing to derive new clauses from given ones:

$$\frac{C \vee x \qquad D \vee \neg x}{C \vee D} \quad \text{(Resolution rule)}.$$

A *resolution derivation* of a clause D from a set of clauses ϕ is a sequence of clauses $(D_1,\ldots,D_s \equiv D)$ such that for every $1 \leq i \leq s$ either $D_i \in \phi$ or D_i is obtained from previous clauses by applying the resolution rule. A *resolution refutation* of $\phi \in$ UNSAT is a resolution derivation of the empty clause from ϕ, which stands for the truth value False.

A resolution derivation is *tree-like* if every clause in it is used at most once as a premise of a rule. Accordingly, *tree-like resolution* is the resolution system allowing only tree-like refutations.

Let $\mathbb{F}$ be a field. A *polynomial calculus* [10] derivation of a polynomial $q \in \mathbb{F}[x_1,\ldots,x_n]$ from a set of polynomials $\mathcal{P} \subseteq \mathbb{F}[x_1,\ldots,x_n]$ is a sequence $(p_1,\ldots,p_s), p_i \in \mathbb{F}[x_1,\ldots,x_n]$ such that for every $1 \leq i \leq s$ either $p_i = x_j^2 - x_j$, $p_i \in \mathcal{P}$ or p_i is obtained from previous polynomials by applying one of the following rules:

$$\frac{f \qquad g}{\alpha f + \beta g} \quad (\alpha, \beta \in \mathbb{F}, f, g \in \mathbb{F}[x_1, \ldots, x_n]) \qquad \frac{f}{x \cdot f} \quad (f \in \mathbb{F}[x_1, \ldots, x_n]) \,.$$

A polynomial calculus refutation of $\mathcal{P} \subseteq \mathbb{F}[x_1, \ldots, x_n]$ is a derivation of 1. The degree $d(\pi)$ of a polynomial calculus derivation π is the maximal total degree of a polynomial appearing in it. This defines the proof system $PC_{\mathbb{F}}$ for the language of unsatisfiable systems of polynomial equations over $\mathbb{F}$. It can be turned into a proof system for k-UNSAT via *arithmetization of clauses* as follows: $(x_1 \vee \ldots \vee x_k \vee \neg y_1 \vee \ldots \vee \neg y_l)$ is represented as $(1 - x_1) \cdot \ldots \cdot (1 - x_k) \cdot y_1 \cdot \ldots \cdot y_l = 0$.

2.3 Hard Instances

2.3.1 Pigeonhole Principle

The *pigeonhole principle* states that there is no injective mapping from the set $[m]$ to the set $[n]$, for $m > n$. Elements of the former and the latter sets are referred to as *pigeons* and *holes*, respectively. The CNF formula, denoted PHP_n^m, encoding the negation of this principle is defined as follows. Let the set of propositional variables $\{x_{i,j}\}_{i \in [m], j \in [n]}$ correspond to the mapping from $[m]$ to $[n]$, that is, $x_{i,j} = 1$ iff the i^{th} pigeon is mapped to the j^{th} hole. Then $\neg\mathrm{PHP}_n^m := \mathsf{Pigeons}_n^m \cup \mathsf{Holes}_n^m \in \mathrm{UNSAT}$, where $\mathsf{Pigeons}_n^m = \{\bigvee_{j \in [n]} x_{i,j}\}_{i \in [m]}$ are axioms for pigeons and $\mathsf{Holes}_n^m = \{\neg x_{i,j} \vee \neg x_{i',j}\}_{i \neq i' \in [m], j \in [n]}$ are axioms for holes.

2.3.2 Mod p Tseitin Formulas

We use the version given in [2] (which is different from the one in [9, 26]). Let $G = (V, E)$ be a directed d-regular graph. We assign to every edge $(u, v) \in E$ a corresponding variable $x_{(u,v)}$. Let $\sigma : V \to \mathbb{F}_p$. The *Tseitin mod p formulas* $\neg\mathrm{TS}_{G,\sigma}^{(p)}$ are the CNF encoding of the following equations for all $u \in V$:

$$\sum_{(u,v) \in E} x_{(u,v)} - \sum_{(v,u) \in E} x_{(v,u)} \equiv \sigma(u) \mod p \,. \tag{4}$$

Note that we use the standard encoding of boolean functions as CNF formulas and the number of clauses, required to encode these equations is $O(2^d|V|)$. $\neg\mathrm{TS}_{G,\sigma}^{(p)}$ is unsatisfiable if $\sum_{u \in V} \sigma(u) \not\equiv 0 \mod p$. To see this, note that if we sum (4) over all nodes $u \in V$ we obtain precisely $\sum_{u \in V} \sigma(u)$ which is different from $0 \mod p$; but on the other hand, in this sum over all nodes $u \in V$ each edge $(u, v) \in E$ appears once with a positive sign as an outgoing edge from u and with a negative sign as an incoming edge to v, meaning the the total sum is 0, which is a contradiction.

In particular, $\neg\mathrm{TS}_{G,\sigma}^{(2)}$ are the classical Tseitin formulas [28] and $\mathrm{TS}_{G,1}^{(2)}$, where 1 is the constant function $v \mapsto 1$ (for all $v \in V$), expresses the fact that the sum of total degrees (incoming + outgoing) of the vertices is even.

The proof complexity of Tseitin tautologies depends on the properties of the graph G. For example, if G is just a union of K_{d+1} (the complete graphs on $d + 1$ vertices), then they are easy to prove. On the other hand, they are known to be hard for some proof systems if G satisfies certain expansion properties.

Let $G = (V, E)$ be an *undirected* graph. For $U, U' \subseteq V$ define $e(U, U') := \{(u, u') \in E \mid u \in U, u' \in U'\}$. Consider the following measure of expansion for $r \geq 1$:

$$c_E(r, G) := \min_{|U| \leq r} \frac{e(U, V \backslash U)}{|U|}$$

G is (r, d, c)-expander if G is d-regular and $c_E(r, G) \geq c$. There are explicit constructions of good expanders. For example:

▶ **Proposition 2** (Lubotzky *et. al* [22]). *For any d, there exists an explicit construction of d-regular graph G, called Ramanujan graph, which is $(r, d, d(1 - \frac{r}{n}) - 2\sqrt{d-1})$-expander for any $r \geq 1$.*

▶ **Proposition 3** (Alekhnovich-Razborov [2]). *For any fixed prime p there exists a constant $d_0 = d_0(p)$ such that the following holds. If $d \geq d_0$, G is a d-regular Ramanujan graph on n vertices (augmented with arbitrary orientation of its edges) and $char(\mathbb{F}) \neq p$, then for every function σ such that $\neg TS^{(p)}_{G,\sigma} \in UNSAT$ every $PC_{\mathbb{F}}$ refutation of $\neg TS^{(p)}_{G,\sigma}$ has degree $\Omega(dn)$.*

2.3.3 Random k-CNFs

A random k-CNF is a formula $\phi \sim \mathcal{F}_k^{n,\Delta}$ with n variables that is generated by picking randomly and independently $\Delta \cdot n$ clauses from the set of all $\binom{n}{k} \cdot 2^k$ clauses.

▶ **Proposition 4** (Alekhnovich-Razborov [2]). *Let $\phi \sim \mathcal{F}_k^{n,\Delta}, k \geq 3$ and $\Delta = \Delta(n)$ is such that $\Delta = o\left(n^{\frac{k-2}{2}}\right)$. Then every $PC_{\mathbb{F}}$ refutation of ϕ has degree $\Omega\left(\frac{n}{\Delta^{2/(k-2)} \cdot \log \Delta}\right)$ with probability $1 - o(1)$ for any field $\mathbb{F}$.*

3 Resolution over Linear Equations for General Rings

In this section we define and outline some basic properties of systems that are extensions of resolution, where clauses are disjunctions of linear equations over a ring R: $(\sum_{i=0}^{n} a_{1i}x_i + b_1 = 0) \vee \cdots \vee (\sum_{i=0}^{n} a_{ki}x_i + b_k = 0)$. Recall that disjunctions of this form are called *linear clauses*, and that we assume that all disjuncts are distinct, hence contract duplicate linear equations. We sometimes abuse notation by writing a linear equation as $(\sum_{i=0}^{n} a_{1i}x_i = -b_1)$ instead of $(\sum_{i=0}^{n} a_{1i}x_i + b_1 = 0)$.

The rules of Res(lin_R) are as follows (cf. [26]):

$$\text{(Resolution)} \quad \frac{C \vee f(\overline{x}) = 0 \qquad D \vee g(\overline{x}) = 0}{C \vee D \vee (\alpha f(\overline{x}) + \beta g(\overline{x})) = 0} \quad (\alpha, \beta \in R)$$

$$\text{(Simplification)} \ \frac{C \vee a = 0}{C} \ (0 \neq a \in R) \qquad \text{(Weakening)} \ \frac{C}{C \vee f(\overline{x}) = 0}$$

where $f(\overline{x}), g(\overline{x})$ are linear forms over R and C, D are linear clauses. Note that contraction of duplicates disjuncts is done automatically when applying the resolution rule. The *boolean axioms* are defined as follows:

$$x_i = 0 \vee x_i = 1, \ \text{ for } x_i \text{ a variable}$$

A Res(lin_R) *derivation* of a linear clause D from a set of linear clauses ϕ is a sequence of linear clauses $(D_1, \ldots, D_s \equiv D)$ such that for every $1 \leq i \leq s$ either $D_i \in \phi$ or is a boolean axiom or D_i is obtained from previous clauses by applying one of the rules above. A Res(lin_R)

refutation of an unsatisfiable set of linear clauses ϕ is a Res(lin_R) derivation of the empty clause (which stands for false) from ϕ. The ***size*** of a Res(lin_R) derivation is the total size of all the clauses in the derivation, where the size of a clause is defined to be the total number of occurrences of variables in it plus the total size of all the coefficient occurring in the clause. The size of a coefficient when using integers (or integers embedded in characteristic zero rings) will be the standard size of the binary representation of integers.

In this definition we assume that R is a non-trivial ($R \neq \mathbf{0}$) ring such that there are polynomial-time algorithms for addition, multiplication and taking additive inverses.

Along with size, we will be dealing with two complexity measures of derivations: *width* and *principal width*.

▶ **Definition 5.** *A clause $C = (f_1 = 0 \vee \cdots \vee f_m = 0)$ has **width** $\omega(C) = m$ and **principal width** $\omega_0(C) = \left|\{f_i\}_{i\in[m]}/_\sim\right|$ where $\sim$ identifies R-linear forms $f_i = 0$ and $f_j = 0$ if they define parallel hyperplanes, that is, if $f_i = Af_j + B$ or $f_j = Af_i + B$ for some $A, B \in R$. For $\mu \in \{\omega, \omega_0\}$, the measure μ associated with a* Res(lin_R) *derivation $\pi = (D_1, \ldots, D_s)$ is $\mu(\pi) := \max_{1\leq i\leq s} \mu(D_i)$. For $\phi \in$ UNSAT, denote by $\mu(\phi \vdash \bot)$ the minimal value of $\mu(\pi)$ over all* Res(lin_R) *refutations π.*

▶ **Proposition 6.** Res(lin_R) *is sound and complete. It is also implicationally complete, that is if ϕ is a set of linear clauses and C is a linear clause such that $\phi \models C$, then there exists a* Res(lin_R) *derivation of C from ϕ.*

Proof. The soundness can be checked by inspecting that each rule of Res(lin_R) is sound. Implicational completeness (and thus completeness) follows from Proposition 28. ◀

We now define two systems of resolution with linear equations over a ring, where some of the rules are semantic: Res$_{sw}$(lin_R) and Sem-Res(lin_R). Res$_{sw}$(lin_R) is obtained from Res(lin_R) by replacing the boolean axioms with $0 = 0$, discarding simplification rule and replacing the weakening rule with the following *semantic weakening rule*:

$$\text{(Semantic weakening)}\ \frac{C}{D}\ (C \models D)$$

The system Sem-Res(lin_R) has no axioms except for $0 = 0$, and has only the following *semantic resolution rule*:

$$\text{(Semantic resolution)}\ \frac{C \qquad C'}{D}\ (C \wedge C' \models D)$$

It is easy to see that Res(lin_R) $\leq_p$ Res$_{sw}$(lin_R) $\leq_p$ Sem-Res(lin_R), where $P \leq_p Q$ denotes that Q polynomially simulates P.

In contrast to the case $R = \mathbb{F}_2$ (see [17]), for rings R with $char(R) \notin \{1, 2, 3\}$ both Res$_{sw}$(lin_R) and Sem-Res(lin_R) are not Cook-Reckhow proof systems, unless P = NP:

▶ **Proposition 7.** *The following decision problem is* coNP*-complete: given a linear clause over a ring R with $char(R) \notin \{1, 2, 3\}$ decide whether it is a tautology under 0-1 assignments.*

Proof. Consider a 3-DNF ϕ and encode every conjunct $(x_{i_1}^{\sigma_1} \wedge \cdots \wedge x_{i_k}^{\sigma_k}) \in \phi, 1 \leq k \leq 3, \sigma_i \in \{0, 1\}$ as the equation $(1 - 2\sigma_1)x_1 + \cdots + (1 - 2\sigma_k)x_k = k - (\sigma_1 + \cdots + \sigma_k)$, where $x^0 := x, x^1 := \neg x$. Then ϕ is tautological if and only if the disjunction of these linear equations is tautological (that is, for every 0-1 assignment to the variables at least one of the equations hold, when the equations are computed over a ring with characteristic zero or finite characteristic bigger than 3). ◀

We leave it as an open question to determine the complexity of verifying a correct application of the semantic weakening in case $char(R) = 3$ or in case $char(R) = 2$ and $R \neq \mathbb{F}_2$. In the case $R = \mathbb{F}_2$ the negation of a clause is a system of linear equations and thus the existence of solutions for it can be checked in polynomial time. Therefore $\text{Res}_{sw}(\text{lin}_{\mathbb{F}_2})$ is a Cook-Reckhow propositional proof system. The definitions of $\text{Res}(\text{lin}_{\mathbb{F}_2})$, $\text{Res}_{sw}(\text{lin}_{\mathbb{F}_2})$ and Sem-Res$(\text{lin}_{\mathbb{F}_2})$ coincide with the definitions of syntactic Res$(\oplus)$, Res$(\oplus)$ and $\text{Res}_{\text{sem}}(\oplus)$ from [17], respectively[6]. As showed in [17], $\text{Res}(\text{lin}_{\mathbb{F}_2})$, $\text{Res}_{sw}(\text{lin}_{\mathbb{F}_2})$ and Sem-Res$(\text{lin}_{\mathbb{F}_2})$ are polynomially equivalent.

We now show that if $char(R) \notin \{1, 2, 3\}$, then $\text{Res}_{sw}(\text{lin}_R)$ is polynomially bounded as a proof system for 3-UNSAT (that is, admits polynomial-size refutation for every instance):

▶ **Proposition 8.** *If* $char(R) \notin \{1, 2, 3\}$*, then dag-like* $\text{Res}_{sw}(\text{lin}_R)$ *and tree-like* Sem-Res(lin_R) are polynomially bounded (not necessarily Cook-Reckhow) propositionally proof systems for 3-UNSAT.

Proof. Let $\phi(x_1, \ldots, x_n) = \{C_i\}_{i\in[m]} \in$ 3-UNSAT. Given $C = (x_{j_1}^{\sigma_1} \vee \ldots \vee x_{j_k}^{\sigma_k})$ define $lin(\neg C) := ((2\sigma_1 - 1)x_{j_1} + \ldots + (2\sigma_k - 1)x_{j_k} - (\sigma_1 + \ldots + \sigma_k))$ where $\sigma_i \in \{0, 1\}, j_l \in [n], x^0 := x, x^1 := \neg x$. The linear clause $lin(\neg\phi) := \bigvee_{i\in[m]} lin(\neg C_i) = 0$ is a tautology (under 0-1 assignments) and thus can be derived in $\text{Res}_{sw}(\text{lin}_R)$ in a single step as a weakening of $0 = 0$ or resolving $0 = 0$ with $0 = 0$ in tree-like Sem-Res(lin_R).

In tree-like Sem-Res(lin_R) the disjunct $lin(\neg C_i) = 0$ can be eliminated from $lin(\neg\phi)$ by a single resolution with C_i, thus the empty clause is derived by a sequence of m resolutions of $lin(\neg\phi)$ with $C_1, \ldots, C_m$.

Similarly, the disjuncts $lin(\neg C_i) = 0$ are eliminated from $lin(\neg\phi)$ in $\text{Res}_{sw}(\text{lin}_R)$, but with a few more steps. Let D_0 be the empty clause and $D_{s+1} := D_s \vee lin(\neg C_{s+1}) = 0, 0 \leq s < m$. Assume D_{s+1} is derived and assume without loss of generality, that $C_{s+1} = (x_1 = 1 \vee \ldots \vee x_k = 1)$ and thus $lin(\neg C_{s+1}) = (-x_1 - \ldots - x_k)$. Derive D_s as follows. Resolve D_{s+1} with C_{s+1} on $lin(\neg C_{s+1}) + (x_k - 1)$ to get the clause $E_1 := D_s \vee (-x_1 - \ldots - x_{k-1} - 1) = 0 \vee x_1 = 1 \vee \ldots \vee x_{k-1} = 1$ and apply semantic weakening to get $E_1' := D_s \vee x_1 = 1 \vee \ldots \vee x_{k-1} = 1$. Resolve D_{s+1} with E_1' on $lin(\neg C_{s+1}) + (x_{k-1} - 1)$ and apply semantic weakening to get the clause $E_2' := D_s \vee x_1 = 1 \vee \ldots \vee x_{k-2} = 1$. After k steps the clause $D_s = E_k'$ can be derived. ◀

The following proposition is straightforward, but useful as it allows, for example, to transfer results about $\text{Res}(\text{lin}_{\mathbb{Q}})$ to $\text{Res}(\text{lin}_{\mathbb{Z}})$.

▶ **Proposition 9.** *If* R *is an integral domain and* $Frac(R)$ *is its field of fractions, then* $\text{Res}(\text{lin}_R)$ *is equivalent to* $\text{Res}(\text{lin}_{Frac(R)})$ *and tree-like* $\text{Res}(\text{lin}_R)$ *is equivalent to tree-like* $\text{Res}(\text{lin}_{Frac(R)})$.

Proof. Every proof in $\text{Res}(\text{lin}_R)$ is also a proof in $\text{Res}(\text{lin}_{Frac(R)})$. To get the converse, just multiply every line by the least common multiple (lcm) of all the coefficients in the $\text{Res}(\text{lin}_{Frac(R)})$ proof. If $a_1, \ldots, a_N \in R$ is the list of denominators of all the coefficients in a $\text{Res}(\text{lin}_{Frac(R)})$ proof π, then under a reasonable encoding of R: $|lcm(a_1, \ldots, a_N)| \leq |a_1| + \cdots + |a_N| \leq |\pi|$. Therefore the corresponding $\text{Res}(\text{lin}_R)$ proof is of size at most $O(|\pi|^2)$. ◀

[6] There is, however, one minor difference in the formulation of syntactic Res$(\oplus)$ and $\text{Res}(\text{lin}_{\mathbb{F}_2})$: the former does not have the boolean axioms, but has an extra rule (*addition rule*).

3.1 Basic Counting in $\mathrm{Res}(\mathrm{lin}_{\boldsymbol{R}})$ and $\mathrm{Res}_{\boldsymbol{sw}}(\mathrm{lin}_{\boldsymbol{R}})$

Here we introduce several unsatisfiable sets of linear clauses that express some counting principles, and serve to exemplify the ability of dag-like $\mathrm{Res}(\mathrm{lin}_R)$, tree-like $\mathrm{Res}(\mathrm{lin}_R)$ and tree-like $\mathrm{Res}_{sw}(\mathrm{lin}_R)$ to reason about counting, for a ring R. We then summarize what we know about refutations of these instances in our different systems, proving along the way some upper bounds and stating some lower bounds proved in the sequel.

Our unsatisfiable instances are the following:

Linear systems: If $A = (B|b)$ is an $m \times (n+1)$ matrix over R, where the B sub-matrix consists of the first n columns, such that $B\overline{x} = b$ has no 0-1 solutions, then (B_i is the ith row in B):

$$\mathsf{LinSys}(A) := \{B_i \cdot \overline{x} = b_i\}_{i\in[m]}\,. \tag{5}$$

Subset Sum: Let f be a linear form over R such that $0 \notin im_2(f)$. Then,

$$\mathsf{SubSum}(f) := \{f = 0\}\,. \tag{6}$$

Image avoidance: Let f be a linear form over R and recall the notation $\langle f \neq A\rangle$ from Sec. 2.1. We define

$$\mathsf{ImAv}\,(f) := \{\langle f \neq A\rangle : A \in im_2(f)\}\,. \tag{7}$$

We also consider the following (tautological) generalization of the boolean axiom $x = 0 \vee x = 1$.

Image axiom: For f a linear form, define

$$\mathsf{Im}(f) := \bigvee_{A\in im_2(f)} f = A\,. \tag{8}$$

Dag-Like $\mathrm{Res}(\mathrm{lin}_{\boldsymbol{R}})$

Upper bounds. For any given linear polynomial f, $\mathsf{Im}(f)$ has a $\mathrm{Res}(\mathrm{lin}_R)$-derivation of polynomial-size (in the size of $\mathsf{Im}(f)$):

▶ **Proposition 10.** *Let $f = \sum_{i=1}^{n} a_i x_i + b$ be a linear polynomial over R. There exists a $\mathrm{Res}(\mathrm{lin}_R)$ derivation of $\mathsf{Im}(f)$ of size polynomial in $|\mathsf{Im}(f)|$ and of principal width at most 3.*

Proof. We construct derivations of $\mathsf{Im}\left(\sum_{i=1}^{k} a_i x_i + b\right)$, $0 \le k \le n$, inductively on k.

Base case: $k = 0$. In this case $\mathsf{Im}(b)$ is just the axiom $b = b$ and thus derived in one step.

Induction step: Let $f_k := \sum_{i=1}^{k} a_i x_i + b$ and assume $\mathsf{Im}(f_k)$ was already derived. Derive $C_0 := \left(\bigvee_{A\in im_2(f_k)} f_k + a_{k+1}x_{k+1} = A\right) \vee x_{k+1} = 1$ from $\mathsf{Im}(f_k)$ by $|im_2(f_k)|$ many resolution applications with $x_{k+1} = 0 \vee x_{k+1} = 1$. Similarly derive $C_1 := \left(\bigvee_{A\in im_2(f_k)} f_k + a_{k+1}x_{k+1} = A + a_{k+1}\right) \vee x_{k+1} = 0$ and obtain $\mathsf{Im}(f_{k+1})$ by resolving C_0 with C_1 on x_{k+1}. The size of the derivation is $n \cdot |\mathsf{Im}(f)|$, and as there is no clause with more than 3 equations that determines non-parallel hyperplanes, hence the principal width of the derivation is at most 3. ◀

▶ **Proposition 11.** *For every linear polynomial f such that $0 \notin im_2(f)$, the contradiction $\mathsf{SubSum}(f)$ admits $\mathrm{Res}(\mathrm{lin}_R)$ refutation of size polynomial in $|\mathsf{Im}(f)|$.*

Proof. First construct the shortest derivation of $\mathsf{Im}(f)$, and then by a sequence of $|im_2(f)|$ many application of the resolution rule with $f = 0$ derive the empty clause. By Proposition 10 the resulting refutation is of polynomial in $|\mathsf{Im}(f)|$ size. ◀

▶ **Proposition 12.** *Let f be a linear polynomial over R, $a \in im_2(f)$ and $\phi = \{\langle f \neq b\rangle\}_{b \in im_2(f), b \neq a}$. Then there exists* $\mathrm{Res}(\mathrm{lin}_R)$ *derivation π of $f = a$ from ϕ, such that $S(\pi) = poly(|\phi|)$ and $\omega_0(\pi) \leq 3$.*

Proof. Let $A_1, \ldots, A_N = a$ be an enumeration of all the elements in $im_2(f)$. By Proposition 10 there exists a derivation of $\left(\bigvee_{i \geq 1} f = A_i\right)$ of principal width at most 3. For $1 < k < N$, we derive $C := \left(\bigvee_{i \geq k+1} f = A_i\right)$ from $\left(\bigvee_{i \geq k} f = A_i\right) = (C \vee f = A_k)$ and $\langle f \neq A_k\rangle = (C \vee f = A_1 \vee \cdots \vee f = A_{k-1})$ in $k-1$ steps as follows: at the sth step we get $(C \vee f - f = A_s - A_k \vee f = A_{s+1} \vee \cdots \vee f = A_{k-1}) = (C \vee f = A_{s+1} \vee \cdots \vee f = A_{k-1})$ by resolving $C \vee f = A_s \vee \cdots \vee f = A_{k-1}$ with $C \vee f = A_k$. We thus obtain a derivation of principal width $\omega_0 \leq 3$ and of size $(1 + \cdots + (N-2))|f| = \frac{(N-1)(N-2)}{2}|f|$. ◀

▶ **Corollary 13.** *For every ring R and every linear polynomial f the contradiction* $\mathsf{ImAv}(f)$ *admits polynomial-size* $\mathrm{Res}(\mathrm{lin}_R)$ *refutations.*

Proof. Pick some $a \in im_2(f)$. By Proposition 12 there is a derivation of $f = a$ from $\mathsf{ImAv}(f)$ of polynomial size. This derivation can be extended to a refutation of $\mathsf{ImAv}(f)$ by a sequence of resolution rule applications of $f = a$ with $\langle f \neq a\rangle \in \mathsf{ImAv}(f)$. ◀

All $\mathrm{Res}(\mathrm{lin}_R)$ upper bounds for $\mathsf{LinSys}(A)$ are tree-like. So for more $\mathsf{LinSys}(A)$ upper bounds we refer the reader to the tree-like $\mathrm{Res}(\mathrm{lin}_R)$ upper bounds further in this section.

Lower bounds. In Sec. 4 we prove an exponential lower bound for $\mathsf{SubSum}(f)$ in case f is a linear polynomial with large coefficients (Theorem 21).

Tree-Like $\mathrm{Res}(\mathrm{lin}_R)$

Upper bounds. In case R is a finite ring, in Sec. 5.1 we prove that the clauses in $\mathsf{Im}(f)$ admit derivations of polynomial size (Theorem 29). Obviously, in that case (R is finite) any unsatisfiable R-linear equation $f = 0$ has at most $|R|$ variables and $\mathsf{SubSum}(f)$ are always refutable in constant size. In contrast, in case $R = \mathbb{Q}$ we prove a lower bound for $\mathsf{Im}(f)$, $\mathsf{SubSum}(f)$ and $\mathsf{ImAv}(f)$ for a specific f with small coefficients (see the lower bounds below).

In case a matrix $A = (B|b)$ with entries in a field $\mathbb{F}$ defines a system of equations $B\overline{x} = b$, that is unsatisfiable under arbitrary $\mathbb{F}$-valued assignments (not just under 0-1 assignments), we prove a polynomial upper bound for tree-like $\mathrm{Res}(\mathrm{lin}_{\mathbb{F}})$ refutations of $\mathsf{LinSys}(A)$.

▶ **Proposition 14.** *If a $m \times (n+1)$ matrix $A = (B|b)$ with entries in a field $\mathbb{F}$ is such that $B\overline{x} = b$ has no $\mathbb{F}$-valued solutions, then there exists tree-like* $\mathrm{Res}(\mathrm{lin}_{\mathbb{F}})$ *refutation of* $\mathsf{LinSys}(A)$ *of linear size.*

Proof. It is a well-known fact from linear algebra that $B\overline{x} = b$ has no $\mathbb{F}$-valued solutions iff there exists $\alpha \in \mathbb{F}^m$ such that $\alpha^T B = 0$ and $\alpha^T b = 1$. Therefore, by $m-1$ resolutions of $B_1\overline{x} - b_1 = 0, \ldots, B_m\overline{x} - b_m = 0$ we can derive $-\alpha_1(B_1\overline{x} - b_1) - \ldots - \alpha_m(B_m\overline{x} - b_m) = 0$, which is $1 = 0$. ◀

Lower bounds. In Sec. 4 we prove tree-like $\mathrm{Res}(\mathrm{lin}_{\mathbb{Q}})$ exponential-size lower bounds for derivations of $\mathsf{Im}(f)$ and refutations of $\mathsf{SubSum}(f)$ for any f (Corollary 34 and Theorem 35). For $\mathsf{ImAv}(f)$ whenever f is of the form $f = \epsilon_1 x_1 + \ldots + \epsilon_n x_n - A$ for some $\epsilon_i \in \{-1, 1\}, A \in \mathbb{F}$ the lower bound holds even for the stronger system tree-like $\mathrm{Res}_{sw}(\mathrm{lin}_{\mathbb{F}})$ (see below).

Tree-Like $\mathrm{Res}_{sw}(\mathrm{lin}_R)$

Upper bounds. Most of the instances above admit short derivations/refutations in tree-like $\mathrm{Res}_{sw}(\mathrm{lin}_R)$: $\mathsf{Im}(f)$ is semantic weakening of $0 = 0$ and thus derivable in one step; The empty clause is a semantic weakening of $\mathsf{SubSum}(f)$ and $\mathsf{LinSys}(A)$ and thus can be refuted via deriving $\bigvee_{i\in[m]} \langle A_i\overline{x} - b_i \neq 0\rangle$ as a semantic weakening of $0 = 0$ and resolving it with equalities in $\mathsf{LinSys}(A) = \{A_i\overline{x} - b_i = 0\}_{i\in[m]}$.

Lower bounds. In case $\mathbb{F}$ is a field of characteristic zero, $\mathsf{ImAv}(f)$ are hard even for tree-like $\mathrm{Res}_{sw}(\mathrm{lin}_R)$ whenever f is of the form $f = \epsilon_1 x_1 + \ldots + \epsilon_n x_n - A$ for some $\epsilon_i \in \{-1, 1\}, A \in \mathbb{F}$ (Theorem 37).

3.2 CNF Upper Bounds for $\mathrm{Res}(\mathrm{lin}_R)$

In this section we outline two basic polynomial upper bounds, which we use to establish our separations in subsequent sections: short tree-like $\mathrm{Res}(\mathrm{lin}_R)$ refutations for CNF encodings of linear systems over a ring R, and short $\mathrm{Res}(\mathrm{lin}_R)$ refutations for $\neg\mathrm{PHP}^m_n$. Together with our lower bounds, these imply the separation between tree-like $\mathrm{Res}(\mathrm{lin}_{\mathbb{F}})$ and tree-like $\mathrm{Res}(\mathrm{lin}_{\mathbb{F}'})$, where $\mathbb{F}, \mathbb{F}'$ are fields of positive characteristic such that $char(\mathbb{F}) \neq char(\mathbb{F}')$. The short refutation of the pigeonhole principle will imply a separation between dag-like and tree-like $\mathrm{Res}(\mathrm{lin}_{\mathbb{F}})$ for fields $\mathbb{F}$ of characteristic 0.

In what follows we consider standard CNF encodings of linear equations $f = 0$ where the linear equations are considered as boolean functions (i.e., functions from 0-1 assignments to $\{0, 1\}$); we do not use extension variable in these encodings.

▶ **Proposition 15.** *Let $\mathbb{F}$ be a field and $A\overline{x} = b$ be a system of linear equations that has no solution over $\mathbb{F}$, where A is $k \times n$ matrix with entries in $\mathbb{F}$, and A_i denotes the ith row in A. Assume that ϕ_i is a CNF encoding of $A_i \cdot \overline{x} - b_i = 0$, for $i \in [k]$. Then, there exists a tree-like* $\mathrm{Res}(\mathrm{lin}_{\mathbb{F}})$ *refutation of $\phi = \{\phi_i\}_{i\in[k]}$ of size polynomial in $|\phi| + \sum_{i\in[k]} \left|A_i \cdot \overline{x} - b_i = 0\right|$.*

Proof. The idea is to derive the actual linear system of equations from their CNF encoding, and then refute the linear system using a previous upper bound (Proposition 14).

If n_i is the number of variables in $A_i \cdot \overline{x} - b_i = 0$, then $|\phi_i| = \Theta(2^{n_i})$. By Proposition 28 proved in the sequel there exists a tree-like $\mathrm{Res}(\mathrm{lin}_{\mathbb{F}})$ derivation of $A_i \cdot \overline{x} - b_i = 0$ from ϕ_i of size $O(2^{n_i}|A_i \cdot \overline{x} - b_i = 0|) = O(|\phi_i| \cdot \left|A_i \cdot \overline{x} - b_i = 0\right|)$.

By Proposition 14 there exists a tree-like $\mathrm{Res}(\mathrm{lin}_{\mathbb{F}})$ refutation of $\{A_i \cdot \overline{x} - b_i = 0\}_{i\in[k]}$ of size $O\left(\sum_{i\in[k]} |A_i \cdot \overline{x} - b_i = 0|\right)$. The total size of the resulting refutation of ϕ is $O\left(\sum_{i\in[k]} \left|\phi_i\right| \cdot |A_i \cdot \overline{x} - b_i = 0|\right)$ and thus is $O\left(\left(\sum_{i\in[k]} |\phi_i| + \sum_{i\in[k]} |A_i \cdot \overline{x} - b_i = 0|\right)^2\right) = O\left(\left(|\phi| + \sum_{i\in[k]} |A_i \cdot \overline{x} - b_i = 0|\right)^2\right)$. ◀

As a corollary we get the polynomial upper bound for the Tseitin formulas (see Sec. 2.3.2 for the definition):

▶ **Theorem 16.** *Let $G = (V, E)$ be a d-regular directed graph, p a prime number, $\sigma : V \to \mathbb{F}_p$ such that $\sum_{u \in V} \sigma(u) \not\equiv 0 \pmod p$, then $\neg\mathrm{TS}^{(p)}_{G,\sigma}$ admit tree-like $\mathrm{Res}(\mathrm{lin}_{\mathbb{F}_p})$ refutations of polynomial size.*

Proof. $\neg\mathrm{TS}^{(p)}_{G,\sigma}$ is an unsatisfiable system of linear equations over $\mathbb{F}_p$ (note that no assignment of $\mathbb{F}$-elements to the variables in $\neg\mathrm{TS}^{(p)}_{G,\sigma}$ is satisfying, and so we do not need to use the (non-linear) boolean axioms to get the unsatisfiability of the system of equations). Therefore, by Proposition 15 there exists a tree-like $\mathrm{Res}(\mathrm{lin}_{\mathbb{F}_p})$ refutation of $\neg\mathrm{TS}^{(p)}_{G,\sigma}$ of polynomial size. ◀

▶ **Theorem 17** (Raz and Tzameret [26])**.** *Let R be a ring such that $char(R) = 0$. There exists a $\mathrm{Res}(\mathrm{lin}_R)$ refutation of $\neg PHP^m_n$ of polynomial size.*

Proof. This follows from the upper bound of [26] for $\mathrm{Res}(\mathrm{lin}_{\mathbb{Z}})$ and the fact that any $\mathrm{Res}(\mathrm{lin}_{\mathbb{Z}})$ proof can be interpreted as $\mathrm{Res}(\mathrm{lin}_R)$ if R is of characteristic 0. ◀

4 Dag-Like Lower Bound

4.1 Lower Bound for Subset Sum with Large Coefficients

In this section we prove an exponential lower bound on the size of dag-like $\mathrm{Res}(\mathrm{lin}_{\mathbb{Q}})$ refutations of $\mathsf{SubSum}(f)$, where $f = 1 + x_1 + \cdots + 2^n x_n$.

The lower bound is obtained by defining a mapping, that sends every refutation π of $f = 0$ to a derivation π' from the boolean axioms of some clause C_π, in such a way that π' satisfies two properties:

1. π' is at most polynomially larger than π;
2. C_π is exponentially large.

We ensure that the second property holds by defining the construction of π' in such a way that every disjunct $g = 0$ in C_π has a sufficiently small number Z_g of 0-1 solutions, namely Z_g is at most 2^{cn}, for some constant $c < 1$. This, together with the observation that C_π must be a boolean tautology, because it is derivable from the boolean axioms only, implies that C_π must be of exponential size (since C_π has 2^n satisfying assignments and each disjunct contributes at most 2^{cn} satisfying disjunctions). Therefore, by the first property, π must be of exponential size.

The fact that f has exponentially large coefficients is essential in our proof that C_π is of exponential size. All contradictions of the form $f = 0$, where f has polynomially bounded coefficients, have polynomial dag-like $\mathrm{Res}(\mathrm{lin}_{\mathbb{Q}})$ refutations and, thus, there is no hope to prove strong bounds for dag-like refutations in this case. However, in Sec 5 we prove that any $f = 0$, as long as f depends on n variables, must have tree-like $\mathrm{Res}(\mathrm{lin}_{\mathbb{Q}})$ refutations of size at least $2^{\Omega(\sqrt{n})}$. The argument relies on a similar transformation from refutations π of $f = 0$ to derivations of some C_π and in this way reduces the problem to proving size lower bounds against tree-like $\mathrm{Res}(\mathrm{lin}_{\mathbb{Q}})$ derivations of C_π from the boolean axioms.

In order to deal with both tree-like and dag-like lower bounds we formulate and prove a generalised statement about the translation. For both dag-like and tree-like lower bounds we need that for all the disjuncts $g = 0$ in C_π a certain predicate $\mathcal{P}$ holds for g. In case of the dag-like bound, $\mathcal{P}(g) = 1$ iff $g = 0$ has at most 2^{cn} 0-1 solutions, while in case of the tree-like bound $\mathcal{P}(g) = 1$ iff g depends on at least $\frac{n}{2}$ variables. In Theorem 18 we prove that the translation can be achieved as long as $\mathcal{P}$ satisfies certain properties (in what follows $\mathbb{F}[x_1, \ldots, x_n]_{\leq 1}$ denotes the linear polynomials in $\mathbb{F}[x_1, \ldots, x_n]$).

▶ **Theorem 18.** *Let f be a linear polynomial over a field $\mathbb{F}$ with n variables and let $\mathcal{P} : \mathbb{P}(\mathbb{F}[x_1, \ldots, x_n]_{\leq 1}) \to \{0, 1\}$ be a predicate on the projective space[7] of linear polynomials over $\mathbb{F}$ satisfying the following properties:*

1. *for all linear polynomials g and for all but at most one $a \in \mathbb{F}$: $\mathcal{P}(g + af) = 1$;*
2. *for all $b \in \mathbb{F}$: $\mathcal{P}(b + f) = 1$.*

If there exists $\mathrm{Res}(\mathrm{lin}_{\mathbb{F}})$ *(resp. tree-like* $\mathrm{Res}(\mathrm{lin}_{\mathbb{F}})$*) refutation of $f = 0$ of size S, then there exists* $\mathrm{Res}(\mathrm{lin}_{\mathbb{F}})$ *(resp. tree-like* $\mathrm{Res}(\mathrm{lin}_{\mathbb{F}})$*) derivation of size $O(n \cdot S^3)$ of a linear clause $\bigvee_{j \in [N]} g_j = 0$ (for some positive N), where $\mathcal{P}(g_j) = 1$ for every $j \in [N]$.*

Proof. We now sketch the plan of the proof. Assume that π is a $\mathrm{Res}(\mathrm{lin}_{\mathbb{F}})$ refutation of $f = 0$. By taking out resolutions with $f = 0$ we transform π into a derivation π' of some clause C such that $\mathcal{P}(g) = 1$ for every disjunct $g = 0$ in C. We do this in such a way that π' is not much larger than π: $|\pi'| = O(n \cdot |\pi|^3)$.

Denote $\pi_{\leq k}$ the fragment of π, consisting of the first k lines of π. By induction on k we define the sequence π'_k of derivations of some clauses D_k from boolean axioms. The derivations π'_k are defined together with a surjective function τ_k from lines of $\pi_{\leq k}$ to lines of π'_k such that if $D = \left(\bigvee_{t \in [m]} g_t = 0 \right)$ is a line in $\pi_{\leq k}$, then

$$\tau_k(D) = \left(\bigvee_{t \in [m]} g_t + a_t f = 0 \right) \vee \bigvee_{s \in [m']} h_s = 0$$

is a line in π'_k, where $a_t \in \mathbb{F}$ and each h_s is a linear polynomial. Moreover, $\tau_k(D)$ satisfies the following properties:

1. For each $h_s = 0$: $\mathcal{P}(h_s) = 1$.
2. The sets H_D of disjuncts $h_s = 0$ in $\tau_k(D)$ are not too large: $\left| \bigcup_{D \in \pi_{\leq k}} H_D \right| \leq 2|\pi_{\leq k}|$.
3. The numbers a_t and coefficients of h_s are not too large: their bit-size does not exceed the maximal bit-size of coefficients in π.

Before we proceed to the inductive definition of π'_k, we finish the proof assuming that π'_k described above exists. If l is the length of π, then $\pi' := \pi'_l$ contains a derivation of $\tau_l(\emptyset)$, where $\emptyset$ denotes the empty clause.

We now turn to the inductive definition of π'_k.

Base case: Define π'_0 to be the empty derivation.
Induction step: Assume π'_k and τ_k satisfy the properties above and k is smaller than the length of π. If D is the last line of $\pi_{\leq k+1}$, then τ_{k+1} extends τ_k to D and π'_{k+1} either extends π'_k with $\tau_{k+1}(D)$ or coincides with π'_k. Consider the possible cases in which the last line D of $\pi_{\leq k+1}$ is derived:

Case 1: Boolean axiom: $D = (x_i = 0 \vee x_i = 1)$. Then π'_{k+1} extends π'_k with D and $\tau_{k+1}(D) = D$.

Case 2: $D = (f = 0)$. Then π'_{k+1} extends π'_k with the axiom $0 = 0$ and $\tau_{k+1}(D) = (f - f = 0)$.

Case 3: D is derived by resolution: $D = (C_1 \vee C_2 \vee \alpha G_1 + \beta G_2 = 0)$ for some lines $(C_1 \vee G_1 = 0)$ and $(C_2 \vee G_2 = 0)$ in $\pi_{\leq k}$.

---

[7] Here, a *projective space* $\mathbb{P}(\mathbb{F}[x_1, \ldots, x_n]_{\leq 1})$ means the set of linear polynomials quotient by the relation $f \sim \alpha f$ for nonzero scalars α.

If $C_i = \bigvee_{t\in[m_i]} g_t^{(i)} = 0$, by induction hypothesis $\tau_k(C_i \vee G_i = 0)$ is of the form ($i = 1, 2$; $A_i \in \mathbb{F}$):

$$\tau_k(C_i \vee G_i = 0) = \left(G_i + A_i f = 0 \vee \bigvee_{t\in[m_i]} g_t^{(i)} + a_t^{(i)} f = 0 \right) \vee \bigvee_{s\in[m'_i]} h_s^{(i)} = 0$$

Define $\tau_{k+1}(D)$ to be the following resolution of $\tau_k(C_1 \vee G_1 = 0) \in \pi'_k$ with $\tau_k(C_2 \vee G_2 = 0) \in \pi'_k$:

$$\tau_{k+1}(D) := \left(\alpha G_1 + \beta G_2 + (\alpha A_1 + \beta A_2) f = 0 \vee \bigvee_{i=1,2} \bigvee_{t\in[m_i]} g_t^{(i)} + a_t^{(i)} f = 0 \right) \vee \vee \bigvee_{i=1,2} \bigvee_{s\in[m'_i]} h_s^{(i)} = 0$$

The derivation π'_{k+1} extends π'_k with $\tau_{k+1}(D)$. It remains to be shown that $\tau_{k+1}(D)$ is of required form and that τ_{k+1} satisfies the required properties.

If we consider the clause $(\alpha G_1 + \beta G_2 = 0 \vee C_1 \vee C_2)$ as a *multiset* of disjuncts and C_1, C_2, as usual, as sets of disjuncts, there can be up to three identical copies of $g = 0$ (from C_1, from C_2 and from $\{\alpha G_1 + \beta G_2 = 0\}$), that are contracted to a single element in the set D. In $\tau_{k+1}(D)$ these copies can be different because of different $+af$ terms and, thus, can be non-contractible.

For every disjunct $g = 0$ in D, denote $\mathcal{F}_g$ the set of disjuncts in $\tau_{k+1}(D)$ that correspond to g, namely, $(g_j^{(i)} + a_j^{(i)} f = 0) \in \mathcal{F}_g$ iff $g_j^{(i)} = g$ and $(\alpha G_1 + \beta G_2 + (\alpha A_1 + \beta A_2) f = 0) \in \mathcal{F}_g$ iff $\alpha G_1 + \beta G_2 = g$. For every $g = 0 \in D$, pick one element $g + af = 0 \in \mathcal{F}_g$, which minimises $\mathcal{P}(g + af)$, and denote X the set of these elements. Denote $Y := \left(\bigcup_{g=0\in D} \mathcal{F}_g\right) \setminus X$. Write $\tau_{k+1}(D)$ as follows:

$$\tau_{k+1}(D) = \left(\bigvee_{g+af=0\in X} g + af = 0 \right) \vee \left(\bigvee_{i=1,2} \bigvee_{s\in[m'_i]} h_s^{(i)} = 0 \vee \bigvee_{g+af=0\in Y} g + af = 0 \right)$$

We now show that τ_{k+1} satisfies all the desired properties:

1. For every $h_s^{(i)} = 0$, $\mathcal{P}(h_s^{(i)}) = 1$ holds by induction hypothesis. For every $g + af = 0 \in Y$, $\mathcal{P}(g + af) = 1$ holds by definition of Y.
2. Note that $|H_D \setminus \{h_s^{(i)} = 0\}_{i,s}| \leq 2|D|$. By induction hypothesis $|\bigcup_{\tilde{D}\in\pi_{\leq k}} H_{\tilde{D}}| \leq 2|\pi_{\leq k}|$. It follows that $|\bigcup_{\tilde{D}\in\pi_{\leq k}} H_{\tilde{D}} \cup H_D| = |\bigcup_{\tilde{D}\in\pi_{\leq k}} H_{\tilde{D}} \cup (H_D \setminus \{h_s^{(i)} = 0\}_{i,s})| \leq |\bigcup_{\tilde{D}\in\pi_{\leq k}} H_{\tilde{D}}| + |H_D \setminus \{h_s^{(i)} = 0\}_{i,s}| \leq 2|\pi_{\leq k}| + 2|D| \leq 2|\pi_{\leq k+1}|$.
3. The absolute values of coefficients in π'_{k+1} do not exceed the maximal absolute value of coefficients in π.

Case 4: D is derived by simplification from a line $D \vee b = 0$ in $\pi_{\leq k}$. If $D = \left(\bigvee_{t\in[m]} g_t = 0 \right)$, then $\tau_k(D \vee b = 0)$ has the form: $\tau_k(D \vee b = 0) = \left(\bigvee_{t\in[m]} g_t + a_t f = 0 \right) \vee b + af = 0$.

If $a = 0$, we apply simplification to $\tau_k(D \vee b = 0)$ to derive $\tau_{k+1}(D) := \left(\bigvee_{t\in[m]} g_t + a_t f = 0 \right)$ and let π'_{k+1} extend π'_k .

Otherwise, if $a \neq 0$, we define $\tau_{k+1}(D)$ to be $\tau_{k+1}(D) := \tau_k(D \vee b = 0)$ and $\pi'_{k+1} := \pi'_k$.

Case 5: D is derived by weakening from a line C of $\pi_{\leq k}$: $D = (C \vee g = 0)$ for some g. Define $\tau_{k+1}(D) := (\tau_k(C) \vee g = 0)$ and let π'_{k+1} extend π'_k with $\tau_{k+1}(D)$. ◀

▶ **Lemma 19.** *Let $g : \mathbb{Z}^n \to \mathbb{Z}$ be a linear function. For the sets $I(g) := im_2(g)$ and $K(g) := g^{-1}(0) \cap \{0,1\}^n$ it holds that $|I(g)| \cdot |K(g)| \leq 3^n$.*

Proof. For every element $a \in I(g)$ choose some $v_a \in \{0,1\}^n$ such that $g(v_a) = a$. Consider the set $X := \{v_a + u\}_{a \in I(g), u \in K(g)} \subset \{0,1,2\}^n$.

It is easy to see that $|X| = |I(g)| \cdot |K(g)|$. Indeed, if $v_a + u = v_{a'} + u'$, then $g(v_a) + g(u) - g(0) = g(v_a + u) = g(v_{a'} + u') = g(v_{a'}) + g(u') - g(0)$ and therefore $a = a', v_a = v_{a'}, u = u'$.

On the other hand, $|X| \leq 3^n$. ◀

▶ **Lemma 20.** *Let $f = 1 + 2x_1 + \cdots + 2^n x_n$ and $g : \mathbb{Z}^n \to \mathbb{Z}$ be a linear function. For any $a \in \mathbb{Z} \backslash \{0\}$ one of the following holds:*

1. *$g = 0$ has at most $3^{\frac{n}{2}}$ 0-1 solutions.*
2. *$g + af = 0$ has at most $3^{\frac{n}{2}}$ 0-1 solutions.*

Proof. For every $b \in \mathbb{Z}$, there exists at most one boolean assignment that satisfies both $g = b$ and $b + af = 0$. Therefore the number of 0-1 solutions of $g + af = 0$ is at most the size of the boolean image $im_2(g)$ of g. By Lemma 19 either $|im_2(g)| \leq 3^{\frac{n}{2}}$ or $|g^{-1}(0) \cap \{0,1\}^n| \leq 3^{\frac{n}{2}}$. ◀

▶ **Theorem 21.** *Let $f = 1 + 2x_1 + \cdots + 2^n x_n$. Any $\mathrm{Res}(\mathrm{lin}_{\mathbb{Q}})$ refutation of $f = 0$ is of size $2^{\Omega(n)}$.*

Proof. Define the predicate $\mathcal{P}(g)$ on linear polynomials over $\mathbb{Q}$ as follows: $\mathcal{P}(g) = 1$ iff $g = 0$ has at most $2^{(0.5 \cdot \log 3)n}$ 0-1 solutions. By Lemma 20, $\mathcal{P}$ satisfies the properties in Theorem 18. Therefore, by Theorem 18, if π is a refutation of $f = 0$, then there exists a derivation π' of some clause $C = \bigvee_{j \in [N]} g_j = 0$ from the boolean axioms, where each $g_j = 0$ has at most $2^{(0.5 \cdot \log 3)n}$ 0-1 solutions. Moreover $|\pi'| = O(n \cdot |\pi|^3)$. As C must be a boolean tautology, that satisfied by 2^n assignments, it must contain at least $2^{(1-0.5 \cdot \log 3)n}$ disjuncts (because every disjunct contributes at most $2^{(0.5 \cdot \log 3)n}$ satisfying assignments). Therefore $|\pi| = 2^{\Omega(n)}$. ◀

5 Tree-Like Lower Bounds

5.1 Nondeterministic Linear Decision Trees

In this section we extend the classical correspondence between tree-like resolution refutations and decision trees (cf. [6]) to tree-like $\mathrm{Res}(\mathrm{lin}_R)$ and tree-like $\mathrm{Res}_{sw}(\mathrm{lin}_R)$. We define *nondeterministic linear decision trees* (NLDT), which generalize parity decision trees, proposed in [17] for $R = \mathbb{F}_2$, to arbitrary rings. We shall use these trees in the sequel to establish some of our upper and lower bounds (though not for our dag-like lower bounds).

Let ϕ be a set of linear clauses (that we wish to refute) and Φ a set of linear non-equalities over R (that we take as assumptions). Consider the following two decision problems:

DP1. Assume $\Phi \models \neg\phi$. Given a satisfying boolean assignment ρ to Φ, determine which clause $C \in \phi$ is violated by ρ by making queries of the form: which of $f|_\rho \neq 0$ or $g|_\rho \neq 0$ hold for linear forms f, g in case $f|_\rho + g|_\rho \neq 0$.

DP2. Similar to DP1, only that we assume $\Phi \models_R \neg\phi$, and given R-valued assignment ρ, satisfying Φ, we ask to find a clause $C \in \phi$ falsified by ρ.

Below we define NLDTs of types $\mathrm{DT}_{sw}(R)$ and $\mathrm{DT}(R)$, which provide solutions to DP1 and DP2, respectively. The root of a tree is labeled with a system Φ, the edges in a tree are labeled with linear non-equalities of the form $f \neq 0$ and the leaves are labeled with clauses $C \in \phi$. Informally, at every node v there is a set Φ_v of all *learned* non-equalities, which is the union of Φ and the set of non-equalities along the path from the root to the node. If v is an internal node, two outgoing edges $f \neq 0$ and $g \neq 0$ define a query to be made at v, where $f + g \neq 0$ is a consequence of Φ_v. If v is a leaf, then $\Phi_v \cup \Phi$ contradicts a clause $C \in \phi$.

Starting from the root, based on the assignment ρ, we go along a path, from the root to a leaf, by choosing in each node to go along the left edge $f \neq 0$ or the right edge $g \neq 0$, depending on whether $f|_\rho \neq 0$ or $g|_\rho \neq 0$. Note that $f|_\rho \neq 0$ and $g|_\rho \neq 0$ may not be mutually exclusive, and this is why the decision made in each node may be *nondeterministic*.

▶ **Definition 22** (Nondeterministic linear decision tree NLDT; $\mathrm{DT}(R)$, $\mathrm{DT}_{sw}(R)$)**.** *Let ϕ be a set of linear clauses and Φ be a set of linear non-equalities over a ring R. A nondeterministic linear decision tree T of type $DT(R)$ and of type $DT_{sw}(R)$ for (ϕ, Φ) is a binary rooted tree, where every edge is labeled with some linear non-equality $f \neq 0$, in such a way that the conditions below hold. In what follows, for a node v, we denote by $\Phi_{r \rightsquigarrow v}$ the set of non-equalities along the path from the root r to v and by Φ_v the set $\Phi_{r \rightsquigarrow v} \cup \Phi$. We say that Φ_v is the set of* learned *non-equalities at v.*

1. *Let v be an internal node. Then v has two outgoing edges labeled by linear non-equalities $f_v \neq 0$ and $g_v \neq 0$, such that:*
   - *If $T \in DT(R)$, then $\alpha f_v + \beta g_v \neq 0 \in \Phi_v \cup \{a \neq 0 \mid a \in R \setminus 0\}$ for some $\alpha, \beta \in R$.*
   - *If $T \in DT_{sw}(R)$, then $\Phi_v \models \alpha f_v + \beta g_v \neq 0$ for some $\alpha, \beta \in R$.*
2. *A node v is a leaf if there is a linear clause $C \in \phi \cup \{0 = 0\}$ which is violated by Φ_v in the following sense:*
   - *If $T \in DT(R)$, then $\neg C \subseteq \Phi_v \cup \{a \neq 0 \mid a \in R \setminus 0\}$.*
   - *If $T \in DT_{sw}(R)$, then $\Phi_v \models \neg C$.*

In case Φ is empty, we sometimes simply write that the NLDT is for ϕ instead of $(\phi, \emptyset)$.

Assume $\Phi \models \neg\phi$. Then an NLDT for $(\phi \cup \{x = 0 \vee x = 1 \,|\, x \in vars(\phi)\}, \Phi)$ of type $\mathrm{DT}(R)$ can be converted into an NLDT of type $\mathrm{DT}_{sw}(R)$ for (ϕ, Φ) by truncating all maximal subtrees with all leaves from $\{x = 0 \vee x = 1 \,|\, x \in vars(\phi)\}$ and marking their roots with arbitrary clauses from ϕ.

Below we give several examples (and basic properties) of NLDTs.

▶ **Example 23.** Let ϕ be a set of clauses, representing unsatisfiable CNF. Then any standard decision tree on boolean variables is an NLDT for $\phi \cup \{x = 0 \vee x = 1 \,|\, x \in vars(\phi)\}$ of type $\mathrm{DT}(R)$, where a branching on the value of a variable x is realized by branching on $(1 - x) + x \neq 0$ to either $1 - x \neq 0$ or $x \neq 0$.

This is illustrated by (the proof of) the following proposition:

▶ **Proposition 24.** *If Φ is a set of linear non-equalities and ϕ is a set of linear clauses over R such that $\Phi \models \neg\phi$, then there exists a $DT(R)$ tree for $(\phi \cup \{x = 0 \vee x = 1 \,|\, x \in vars(\phi \cup \{\neg\Phi\})\}, \Phi)$ of size $O(2^n|\Phi|)$, where $n = |vars(\phi \cup \{\neg\Phi\})|$.*

Proof. Let $vars(\phi \cup \{\neg\Phi\}) = \{x_1, \ldots, x_n\}$ and fix an ordering on these variables. Construct a tree T_0 with 2^n nodes, that branches on $x_1, \ldots, x_n$, in this order. Thus, in every leaf v of T_0 a total assignment to the variables is determined (i.e., $\Phi_v = \{x_i \neq \nu_i\}_{i \in [n]} \cup \Phi$ for some $\nu_i \in \{0, 1\}$). Since $\Phi \models \neg\phi$, this assignment violates either some clause $C = (f_1 = 0 \vee \cdots \vee f_m = 0)$ in ϕ or some non-equality $g \neq 0$ in Φ. We augment T_0 to T by attaching a subtree to every leaf v of T_0 depending on whether the former or latter condition holds for v, as follows:

Case 1: $\{x_i \neq \nu_i\}_{i\in[n]} \models \neg C$. We attach a subtree to v that makes m sequences of branches as follows. If $f_i = a_1x_1 + \ldots + a_nx_n + b$ then $a_1(1-\nu_1)+\ldots+a_n(1-\nu_n)+b \neq 0$ holds and the ith sequence is the following sequence of "substitutions": $(a_1x_1 + a_2(1-\nu_2) + \ldots + a_n(1-\nu_n)+b)+(a_1(1-\nu_1)-a_1x_1) \neq 0$ to $a_1x_1+a_2(1-\nu_2)+\ldots+a_n(1-\nu_n)+b \neq 0$ and $a_1(1-\nu_1)-a_1x_1 \neq 0, \ldots, (a_1x_1+\ldots+a_{n-1}x_{n-1}+a_n(1-\nu_n)+b)+(a_n(1-\nu_n)-a_nx_n) \neq 0$ to $f_i \neq 0$ and $a_n(1-\nu_n) - a_nx_n \neq 0$. All the right branches lead to nodes u such that $\{x_i \neq 0, x_i \neq 1\} \subseteq \Phi_u$ for some $i \in [n]$ and thus they satisfy the DT(R) leaf condition in Definition 22. Such a sequence indeed performs substitutions: the edge to the leftmost node is $f_i \neq 0$ and as we go upwards, we apply the substitutions $x_n \leftarrow 1-\nu_n$, $\ldots$, $x_1 \leftarrow 1-\nu_1$ to this non-equality.
In the leftmost node w in the end of the mth sequence, $\{f_1 \neq 0, \ldots, f_m \neq 0\} \subseteq \Phi_w$ holds and thus again C is violated at w in the sense of Definition 22 and therefore w is a legal DT(R)-leaf.

Case 2: $\{x_i \neq \nu_i\}_{i\in[n]} \models g = 0$, where $g \neq 0 \in \Phi_v$. Let $g = a_1x_1 + \ldots + a_nx_n + b$. Attach to v a subtree that makes the following branches: $(a_1(1-\nu_1)+a_2x_2+\ldots+a_nx_n+b)-(a_1(1-\nu_1)-a_1x_1) \neq 0$ to $(a_1(1-\nu_1)+a_2x_2+\ldots+a_nx_n+b) \neq 0$ and $a_1(1-\nu_1)-a_1x_1 \neq 0,\ldots,$ $(a_1(1-\nu_1)+\ldots+a_{n-1}(1-\nu_{n-1})+a_n(1-\nu_n)+b)-(a_n(1-\nu_n)-a_nx_n) \neq 0$ to $1 \neq 0$ and $a_1(1-\nu_1)-a_1x_1 \neq 0$. All leaves of the subtree satisfy the condition for DT(R) leaves in Definition 22.

The tree T is a DT(R) tree for (ϕ, Φ). ◀

▶ **Example 25.** Let ϕ be as in Example 23. *Parity decision trees*, as defined in [17], are NLDTs for ϕ of type $\mathrm{DT}_{sw}(\mathbb{F}_2)$: branching on the value of an $\mathbb{F}_2$-linear form f is realized by branching from $(1-f)+f \neq 0$ to $1-f \neq 0$ and $f \neq 0$. And the converse also holds: a branching of $f+g \neq 0$ to $f \neq 0$ and $g \neq 0$, where, say, f is a non-constant $\mathbb{F}_2$-linear form, is equivalent to branching on the value of f.

▶ **Example 26.** Let $\phi = \{f_1 = 0, \ldots, f_m = 0\}$, where $f_1, \ldots, f_m$ are R-linear forms such that $f_1 + \ldots + f_m = 1$. Then a polynomial-size NLDT of type DT(R) for ϕ makes the following branchings, where all right edges lead to a leaf: $(f_1 + \ldots + f_{m-1}) + f_m \neq 0$ (this is just $1 \neq 0$) to $f_1 + \ldots + f_{m-1} \neq 0$ and $f_m \neq 0$, $\ldots$, $f_1 + f_2 \neq 0$ to $f_1 \neq 0$ and $f_2 \neq 0$.

We now show the equivalence between NLDTs and tree-like $\mathrm{Res}(\mathrm{lin}_R)$ proofs.

▶ **Theorem 27.** *Let ϕ be a set of linear clauses over a ring R and Φ be a set of linear non-equalities over R. Then, there exist decision trees DT(R) (resp. $\mathrm{DT}_{sw}(R)$) for $(\phi \cup \{x = 0 \vee x = 1 \mid x \in vars(\phi)\}, \Phi)$ (resp. (ϕ, Φ)) of size s iff there exist tree-like $\mathrm{Res}(\mathrm{lin}_R)$ (resp. tree-like $\mathrm{Res}_{sw}(\mathrm{lin}_R)$) derivations of the clause $\neg\Phi = \bigvee_{f\neq 0\in\Phi} f = 0$ from ϕ of size $O(s)$.*

Proof. ($\Rightarrow$) Let T_ϕ be an NLDT of type DT(R) or $\mathrm{DT}_{sw}(R)$ for ϕ. We construct a tree-like $\mathrm{Res}(\mathrm{lin}_R)$ or tree-like $\mathrm{Res}_{sw}(\mathrm{lin}_R)$ derivation from T_ϕ, respectively, as follows. Consider the tree of clauses π_0, obtained from T_ϕ by replacing every vertex u with the clause $\neg\Phi_u$. This tree is not a valid tree-like derivation yet. We augment it to a valid derivation π by appropriate insertions of applications of weakening and simplification rules.

Case 1: If $\neg\Phi_u \in \pi_0$ is a leaf, then Φ_u violates a clause $D \in \phi \cup \{0 = 0\}$. By condition 2 in Definition 22, $\neg\Phi_u$ must be a weakening of D (syntactic for $T_\phi \in \mathrm{DT}(R)$ and semantic for $T_\phi \in \mathrm{DT}_{sw}(R)$) and we add D as the only child of this node.

Case 2: Let $\neg\Phi_u \in \pi_0$ be an internal node with two outgoing edges labeled with $f_u \neq 0$ and $g_u \neq 0$.

If $T_\phi \in \mathrm{DT}(R)$, then $\alpha f_u + \beta g_u \neq 0 \in \Phi_u \cup \{a \neq 0 \,|\, a \in R \setminus 0\}$. Apply resolution to $\neg\Phi_{l(u)} = (\neg\Phi_u \vee f_u = 0)$ and $\neg\Phi_{r(u)} = (\neg\Phi_u \vee g_u = 0)$ to derive $\neg\Phi_u \vee \alpha f_u + \beta g_u = 0$. In case $\alpha f_u + \beta g_u \neq 0 \in \Phi_u$ this clause coincides with $\neg\Phi_u$ and no additional steps are required. In case $\alpha f_u + \beta g_u \neq 0 \in \{a \neq 0 \,|\, a \in R \setminus 0\}$ insert an application of the simplification rule to get a derivation of $\neg\Phi_u$.

If $T_\phi \in \mathrm{DT}_{sw}(R)$, $\Phi_u \models \alpha f_u + \beta g_u \neq 0$, we derive $\neg\Phi_u \vee \alpha f_u + \beta g_u = 0$ from $\neg\Phi_{l(u)} = (\neg\Phi_u \vee f_u = 0)$ and $\neg\Phi_{r(u)} = (\neg\Phi_u \vee g_u = 0)$ by an application of the resolution rule and then deriving $\neg\Phi_u$ by an application of the semantic weakening rule.

($\Leftarrow$) Conversely, assume π is a tree-like $\mathrm{Res}(\mathrm{lin}_R)$ or a tree-like $\mathrm{Res}_{sw}(\mathrm{lin}_R)$ derivation of a (possibly empty) clause $\mathcal{C}$ from ϕ. In what follows, when we say weakening we mean syntactic or semantic weakening depending on π being a tree-like $\mathrm{Res}(\mathrm{lin}_R)$ or a tree-like $\mathrm{Res}_{sw}(\mathrm{lin}_R)$ derivation, respectively.

Let the edges in the proof-tree of π be directed from conclusion to premises. We turn this proof-tree into a decision tree T_π for $(\phi, \neg\mathcal{C})$ as follows. Every node of outgoing degree 2 in the proof-tree π is a clause obtained from its children by a resolution rule. For each such node $C \vee D \vee (\alpha f + \beta g = 0)$ we label its outgoing edges to $C \vee f = 0$ and $D \vee g = 0$ with $f \neq 0$ and $g \neq 0$, respectively. We contract all unlabeled edges, which are precisely those corresponding to applications of weakening and simplification rules. If $C_1, \ldots, C_k$ is a maximal (with respect to inclusion) sequence of weakening and simplification rule applications (the latter occur only in $\mathrm{Res}(\mathrm{lin}_R)$ derivations), then we contract it to C_k. In this way we obtain the tree T_π, where every edge is labeled with linear non-equality and every node u is labeled with a clause C_u such that if $f \neq 0$ and $g \neq 0$ are labels of edges to the left $l(u)$ and to the right $r(u)$ children respectively, then C_u is a weakening and a simplification (the latter again in case of $\mathrm{Res}(\mathrm{lin}_R)$) of the clause $C \vee D \vee \alpha f + \beta g = 0$ for some $\alpha, \beta \in R$, such that $C_{l(u)} = (C \vee f = 0)$, $C_{r(u)} = (D \vee g = 0)$.

We now prove that T_π is a valid decision tree of type $\mathrm{DT}(R)$ (respectively, $\mathrm{DT}_{sw}(R)$) if π is a tree-like $\mathrm{Res}(\mathrm{lin}_R)$ derivation (respectively, tree-like $\mathrm{Res}_{sw}(\mathrm{lin}_R)$ derivation).

Case 1: Assume π is tree-like $\mathrm{Res}(\mathrm{lin}_R)$ derivation. We prove inductively that for every node u in T_π we have $\neg C_u \subseteq \Phi_u$.

Base case: u is the root r. We have $\Phi_r = \neg\mathcal{C} = \neg C_r$.

Induction step: For any other node u assume $\neg C_p \subseteq \Phi_p \cup \{a \neq 0 \,|\, a \in R \setminus 0\}$ holds for its parent node p. Let $f \neq 0$ be the label on the edge from p to u. Then $C_u = (C \vee f = 0)$ for some clause C and C_p must be of the form $(C \vee D)$ for some clause D, and hence $\neg C_u \subseteq \neg C \cup \{f \neq 0\} \subseteq \neg C_p \cup \{f \neq 0\} \subseteq \Phi_p \cup \{f \neq 0\} = \Phi_u$.

Now we show that T_π satisfies the conditions of Definition 22 for $\mathrm{DT}(R)$ trees.

- (Internal nodes) Let u be an internal node of T_π with outgoing edges labeled with $f \neq 0$ and $g \neq 0$. C_u must be both a weakening and a simplification of $(C \vee \alpha f + \beta g = 0)$ for some $\alpha, \beta \in R$ and a linear clause C. If $\alpha f + \beta g \neq 0 \in \{a \neq 0 \,|\, a \in R \setminus 0\}$, then the condition trivially holds, otherwise $\alpha f + \beta g = 0$ cannot be eliminated via simplification and thus $\alpha f + \beta g \neq 0 \in \neg C_u$ and $\neg C_u \subseteq \Phi_u$ imply $\alpha f + \beta g \neq 0 \in \Phi_u$ and the condition for internal nodes in Definition 22 is satisfied.
- (Leaves) Let u be a leaf of T_π. Then C_u must be both a weakening and a simplification of some clause C in $\phi \cup \{x = 0 \vee x = 1 \,|\, x \in vars(\phi)\} \cup \{0 = 0\}$, that is $C_u = (C \vee D)$ for some clause D. Therefore $\neg C_u \subseteq \Phi_u$ implies that C is falsified by Φ_u.

Case 2: Assume π is a tree-like $\mathrm{Res}_{sw}(\mathrm{lin}_R)$ derivation. We prove inductively that for every node u in T_π, $C_u \models \neg\Phi_u$ holds.

Base case: u is the root r and we have $\neg\Phi_r = \mathcal{C} = C_r$.

Induction step: u is a node which is not the root. If $C_p \models \neg\Phi_p$ holds for its parent p and $f \neq 0$ is the label on the edge from p to u, then $(C \vee D \vee \alpha f + \beta g = 0) \models C_p$, $C_u = (C \vee f = 0)$ for some $\alpha, \beta \in R$ a linear form g and some linear clauses C, D. Therefore, $C_u = (C \vee f = 0) \models (C_p \vee f = 0) \models (\neg\Phi_p \vee f = 0) = \neg\Phi_u$.

We now show that T_π satisfies the conditions of Definition 22 for $\mathrm{DT}_{sw}(R)$ trees.

- (Internal nodes) Let u be an internal node of T_π with outgoing edges labeled with $f \neq 0$ and $g \neq 0$. Then $(C \vee \alpha f + \beta g = 0) \models C_u$ for some $\alpha, \beta \in R$ and a linear clause C. Therefore $C_u \models \neg\Phi_u$ implies $\Phi_u \models \alpha f + \beta g \neq 0$.
- (Leaves) Let u be a leaf of T_π. Then C_u must be a weakening of some clause C in $\phi \cup \{0 = 0\}$, that is, $C_u = (C \vee D)$ for some clause D. Therefore $C_u \models \neg\Phi_u$ implies that C is falsified by Φ_u. ◀

An immediate corollary is the following:

▶ **Proposition 28.** *If $\phi \cup \{C\}$ is a set of linear clauses over a ring R such that $\phi \models C$, then there exists a tree-like* $\mathrm{Res}(\mathrm{lin}_R)$ *derivation of C from ϕ of size $O(2^n|C|)$, where $n = \big|vars(\phi \cup \{C\})\big|$.*

Proof. By Proposition 24 there exists a $\mathrm{DT}(R)$ tree for $(\phi \cup \{x = 0 \vee x = 1 \,|\, x \in vars(\phi \cup \{C\})\}, \neg C)$ of size $O(2^n|C|)$ and, thus, by Theorem 27 there exists a tree-like $\mathrm{Res}(\mathrm{lin}_R)$ derivation of C from ϕ of size $O(2^n|C|)$. ◀

We construct an NLDT to prove the following upper bound:

▶ **Proposition 29.** *Let R be a finite ring, $f = a_1x_1 + \cdots + a_nx_n$ a linear form over R, s_f the size of* $\mathsf{lm}(f)$ *(i.e., the size of its encoding) and $d_f = |im_2(f)|$. Then, there exists a tree-like* $\mathrm{Res}(\mathrm{lin}_R)$ *derivation of* $\mathsf{lm}(f)$ *of size $O(s_f n^{2d_f})$.*

Proof. We construct a decision tree of type $\mathrm{DT}(R)$ of size $O(s_f n^{2d_f})$ with the system $\Phi_r = \{f \neq A\}_{A \in im_2(f)}$ at its root r. By Theorem 27 this implies the existence of a tree-like $\mathrm{Res}(\mathrm{lin}_R)$ proof of $\mathsf{lm}(f)$ of the same size.

Let $f^{(1)} := a_1x_1 + \cdots + a_{\lfloor \frac{n}{2} \rfloor}x_{\lfloor \frac{n}{2} \rfloor}$ and $f^{(2)} := a_{\lfloor \frac{n}{2} \rfloor + 1}x_{\lfloor \frac{n}{2} \rfloor + 1} + \cdots + a_nx_n$. The decision tree for $\mathsf{lm}(f)$ is constructed recursively as a tree of height $2d_f$, where a subtree for $\mathsf{lm}\big(f^{(1)}\big)$ or for $\mathsf{lm}\big(f^{(2)}\big)$ is hanged from each leaf. At every node u of depth d the system of non-equalities is of the form: $\Phi_u = \Phi_r \cup \Phi_u^{(1)} \cup \Phi_u^{(2)}$, where $\Phi_u^{(i)} \subseteq \{f^{(i)} \neq A\}_{A \in im_2(f^{(i)})}$, $i \in \{1, 2\}$ and $|\Phi_u^{(1)}| + |\Phi_u^{(2)}| = d$. A node u is a leaf if and only if $\Phi_u^{(i)} = \{f^{(i)} \neq A\}_{A \in im_2(f^{(i)})}$ for some $i \in \{1, 2\}$. The branching at an internal node u is made by the non-equality $f^{(1)} - A_1 + f^{(2)} - A_2 \neq 0$, for some $A_i \in im_2(f^{(i)})$ where $f^{(i)} - A_i \notin \Phi_u^{(i)}, i \in \{1, 2\}$. The size s_n of this tree can be upper bounded as follows: $s_n \leq 2^{2d_f} s_{\lfloor \frac{n}{2} \rfloor + 1} + s_f 2^{2d_f} = O(s_f n^{2d_f})$. ◀

5.2 Prover-Delayer Games

The *Prover-Delayer game* is an approach to obtain lower bounds on resolution refutations introduced by Pudlák and Impagliazzo [25]. The idea is that the non-existence of small decision trees, and hence small tree-like resolution refutations, for an unsatisfiable formula, can be phrased in terms of the existence of a certain strategy for Delayer in a game against Prover, associated to the unsatisfiable formula. We define such games G^R and G^R_{sw} for decision trees $\mathrm{DT}(R)$ and $\mathrm{DT}_{sw}(R)$, respectively. Below we show (Lemma 30) that the existence of certain strategies for the Delayer in G^R and G^R_{sw} imply lower bounds on the size of $\mathrm{DT}(R)$ and $\mathrm{DT}_{sw}(R)$ trees, respectively.

The game

Let ϕ be a set of linear clauses and Φ_s be a set of linear non-equalities. Consider the following game between two parties called Prover and Delayer. The game goes in rounds, consisting of one move of Prover followed by one move of Delayer. The position in the game is determined by a system of linear non-equalities Φ, which is extended by one non-equality after every round. The starting position is Φ_s.

In each round, Prover presents to Delayer a possible branching $f \neq 0$ and $g \neq 0$ over a linear non-equality $f + g \neq 0$, such that $f + g \neq 0 \in \Phi \cup \{a \neq 0 \,|\, a \in R \setminus 0\}$ or $\Phi \models f + g \neq 0$ in G^R and G^R_{sw}, respectively. After that, Delayer chooses either $f \neq 0$ or $g \neq 0$ to be added to Φ, or leaves the choice to the Prover and thus earns a coin. The game G^R finishes, when $\neg C \subseteq \Phi$ for some $C \in \phi \cup \{0 = 0\}$, and G^R_{sw} finishes, when $\Phi \models \neg C$ for some clause $C \in \phi \cup \{0 = 0\}$.

▶ **Lemma 30.** *If there exists a strategy with a starting position Φ_s for Delayer in the game G^R (respectively, G^R_{sw}) that guarantees at least c coins on a set of linear clauses ϕ, then the size of a $DT(R)$ (respectively $DT_{sw}(R)$) tree for ϕ, with the system Φ_s in the root, must be at least 2^c.*

Proof. Assume that T is a tree of type $\mathrm{DT}(R)$ (respectively, $\mathrm{DT}_{sw}(R)$) for ϕ. We define an embedding of the full binary tree B_c of height c to T inductively as follows. We simulate Prover in the game G^R (respectively, G^R_{sw}) by choosing branchings from T and following to a subtree chosen by the Delayer until Delayer decides to earn a coin and leaves the choice to the Prover or until the game finishes. In case we are at a position where Delayer earns a coin, and which corresponds to a vertex u in T, we map the root of B_c to u and proceed inductively by embedding two trees B_{c-1} to the left and right subtrees of u, corresponding to two choices of the Prover. ◀

5.3 Lower Bounds for the Subset Sum with Small Coefficients

We now turn to tree-like lower bounds. In this section we prove tree-like $\mathrm{Res}(\mathrm{lin}_{\mathbb{Q}})$ lower bound for $\mathsf{SubSum}(f)$ including instances, where coefficients of f are small, and tree-like $\mathrm{Res}_{sw}(\mathrm{lin}_{\mathbb{F}})$ lower bound for $\mathsf{ImAv}(\pm x_1 \pm \cdots \pm x_n)$.

The proof of tree-like $\mathrm{Res}(\mathrm{lin}_{\mathbb{Q}})$ lower bound for $\mathsf{SubSum}(f)$ goes in two stages. Assume f depends on n variables. First, as in the proof of dag-like lower bound in Sec. 4 we use Theorem 18 to transform refutations π of $f = 0$ to derivations π' of a clause C_π from only the boolean axioms. We ensure that π' is not much larger than π and C_π possesses the following property, which makes it hard to derive: for every disjunct $g = 0$ in C_π the linear polynomial g depends on at least $\frac{n}{2}$ variables. Second, we use Prover-Delayer games to prove the lower bound for derivations of any clause with this property. The proof that Delayer's strategy succeeds to earn sufficiently many coins is guaranteed by a bound on size of essential coverings of hypercubes.

▶ **Definition 31.** *Let $\mathcal{H}$ be a set of hyperplanes in $\mathbb{Q}^n$. We say that $\mathcal{F}$ forms **essential cover** of the cube $B_n = \{0,1\}^n$ if:*

- *Every point of B_n is covered by some hyperplane in $\mathcal{H}$.*
- *No proper subset $\mathcal{H}' \subsetneq \mathcal{H}$ covers B_n.*
- *No axis in $\mathbb{Q}^n$ is parallel to all hyperplanes in $\mathcal{H}$. In other words, if $\mathcal{H} = \{H_1, \ldots, H_m\}$ and $f_i = 0$ is the linear equation defining H_i, $i \in [m]$, then every variable x_j, $j \in [n]$, occurs with nonzero coefficient in some f_i.*

▶ **Theorem 32** ([21]). *Any essential cover of the cube B_n in $\mathbb{Q}^n$ must contain at least $\frac{1}{2}(\sqrt{4n+1}+1)$ hyperplanes.*

We use Prover-Delayer games to prove the lower bound below.

▶ **Theorem 33.** *Any tree-like* $\mathrm{Res}(\mathrm{lin}_{\mathbb{Q}})$ *derivation of any tautology of the form $\bigvee_{j\in[N]} g_j = 0$, for some positive N, where each g_j is linear over $\mathbb{Q}$ and depends on at least $\frac{n}{2}$ variables, is of size $2^{\Omega(\sqrt{n})}$.*

Proof. According to the definitions in Sec. 5.2 the corresponding Prover-Delayer game is on $0 = 0$ and starts with the position

$$\Phi_r = \{g_j \neq 0 \mid j \in [N]\}.$$

The game finishes at a position Φ, where $\{x_i \neq 0,\ x_i \neq 1\} \subseteq \Phi$ for some $i \in [n]$ or $0 \neq 0 \in \Phi$.

We now define a Delayer's strategy that guarantees $\Omega(\sqrt{n})$ coins and by Lemma 30 obtain the lower bound.

If Φ is a position in the game, denote by $\Phi_c \subset \Phi$ the subset of so-called "coin" non-equalities, that is, non-equalities that were chosen by Prover when Delayer decided to leave the choice to Prover and earn a coin. The number $|\Phi_c|$ is then precisely the number of coins earned by Delayer at Φ. Throughout the game Delayer constructs a partial assignment ρ_I for variables in $I \subseteq [n]$ and a set of non-equalities $\Phi_I \subseteq \Phi_c$, such that:

1. $|\Phi_I| = \Omega(\sqrt{|I|})$;
2. for all $g \neq 0 \in (\Phi \restriction_{\rho_I}) \setminus (\Phi_c \restriction_{\rho_I})$, the function g depends on at least $\frac{n}{2} - |I|$ variables;
3. Φ_I contains variables only from I; and
4. $\Phi_c \restriction_{\rho_I}$ is 0-1 satisfiable.

In the beginning both ρ_I and Φ_I are empty.

Let the position in the game be defined by a system Φ and let the branching chosen by the Prover be $g_1 \neq 0$ and $g_2 \neq 0$, where $g_1 + g_2 \neq 0 \in \Phi$. Delayer does the following. Before making any decision Delayer checks if there exists some nonconstant linear g with variables in $[n] \setminus I$ such that $(\Phi_c \restriction_{\rho_I}) \cup \{g \neq 0\}$ is unsatisfiable over 0-1.

In case it holds, $\Psi := (\Phi_c \setminus \Phi_I) \restriction_{\rho_I} \cup \{g \neq 0\}$ must be 0-1 unsatisfiable. Consider a minimal subset $\Psi' \subseteq \Psi$ such that Ψ' is 0-1 unsatisfiable and denote $I' \subseteq [n]$ the set of variables that occur in Ψ'. As $\Psi'' := \Psi' \setminus \{g \neq 0\}$ is 0-1 satisfiable, there exists an assignment $\rho_{I'}$ for variables in I', that satisfies Ψ''. Delayer extends the assignment ρ_I with $\rho_{I'}$ to $\rho_{I\cup I'}$ and defines $\Phi_{I\cup I'} := \Phi_I \cup \Psi''$.

If $\Psi' = \{g_1 \neq 0, \ldots, g_k \neq 0\}$, then the hyperplanes $H_1, \ldots, H_k$ defined by the equations $g_1 = 0, \ldots, g_k = 0$ form an essential cover of the cube $B_{|I'|}$. Therefore, by Theorem 32, $|\Psi''| = |\Psi'| - 1 \geq \frac{1}{2} \cdot \sqrt{|I'|}$ and thus $|\Phi_{I\cup I'}| \geq \frac{1}{2} \cdot \sqrt{|I|} + \frac{1}{2} \cdot \sqrt{|I'|} \geq \frac{1}{2} \cdot \sqrt{|I \cup I'|}$.

If necessary, Delayer repeats the above procedure constructing extensions $\rho_{I_1} \subset \cdots \subset \rho_{I_L}$ and $\Phi_{I_1} \subset \cdots \subset \Phi_{I_L}$, where $I_1 = I \subset \ldots \subset I_L$, until there is no $g \neq 0$ inconsistent with $\Phi_c \restriction_{\rho_{I_L}}$ as described above. The new value of I is set to I_L. After that Delayer does the following:

1. if $g_1 \restriction_{\rho_I} = 0$, then choose $g_2 \neq 0$;
2. otherwise, if $g_2 \restriction_{\rho_I} = 0$, then choose $g_1 \neq 0$;
3. if none of the above cases hold, leave the choice to Prover and earn a coin.

Denote by Φ' and $\Phi'_c \subseteq \Phi'$ the new position and the subset of "coin" non-equalities, respectively, after the choice is made. It is easy to see that the property that any $g \neq 0 \in (\Phi' \restriction_{\rho_I}) \setminus (\Phi'_c \restriction_{\rho_I})$ depends on at least $\frac{n}{2} - |I|$ variables still holds.

It follows from the definition of Delayer's strategy that Φ_c is always 0-1 satisfiable. Therefore if Φ is the endgame position, that is if $0 \neq 0 \in \Phi$ or $\{x_i \neq 0, x_i \neq 1\} \subset \Phi$ for some $i \in [n]$, then $0 \neq 0 \in (\Phi \restriction_{\rho_I}) \setminus (\Phi_c \restriction_{\rho_I})$ or $\{x_i \neq 0, x_i \neq 1\} \subset (\Phi \restriction_{\rho_I}) \setminus (\Phi_c \restriction_{\rho_I})$ respectively. This implies that $|I| \geq \frac{n}{2} - 1$ and therefore $|\Phi_c| \geq |\Phi_I| \geq \frac{1}{2} \cdot \sqrt{|I|} = \Omega(\sqrt{n})$. Thus the number of coins earned by Delayer is $\Omega(\sqrt{n})$. ◀

▶ **Corollary 34.** *Let f be any linear polynomial over $\mathbb{Q}$ that depends on n variables. Then tree-like* $\mathrm{Res}(\mathrm{lin}_{\mathbb{Q}})$ *derivations of* $\mathsf{Im}(f)$ *are of size* $2^{\Omega(\sqrt{n})}$.

▶ **Theorem 35.** *If f is a linear polynomial over $\mathbb{Q}$, which depends on n variables and $0 \notin im_2(f)$, then every tree-like* $\mathrm{Res}(\mathrm{lin}_{\mathbb{Q}})$ *refutation of $f = 0$ is of size* $2^{\Omega(\sqrt{n})}$.

Proof. Consider the following predicate $\mathcal{P}$ on linear polynomials: $\mathcal{P}(g) = 1$ iff g depends on at least $\frac{n}{2}$ variables. It is easy to see that $\mathcal{P}$ satisfies the conditions in Theorem 18 with respect to f. Therefore by Theorem 18 for every refutation π of $f = 0$ there exists a derivation π' of a clause C_π from the boolean axioms such that $|\pi'| = O(n \cdot |\pi|^3)$ and $\mathcal{P}(g)$ for every $g = 0$ in C_π. Thus, by Theorem 33 $|\pi'| = 2^{\Omega(\sqrt{n})}$ and $|\pi| = 2^{\Omega(\sqrt{n})}$. ◀

▶ **Lemma 36.** *Let Φ be a satisfiable system of m non-equalities over $\mathbb{F}$. If $\Phi \models \epsilon_1 x_1 + \cdots + \epsilon_n x_n = A$ for some $\epsilon_i \in \{-1, 1\} \subset \mathbb{F}, A \in \mathbb{F}$, then $m \geq \frac{n}{4}$.*

Note that A must be an integer (inside $\mathbb{F}$), since the coefficients of variables are all $-1, 1$, and the variables themselves are boolean (since $\models$ stands for semantic implication over 0-1 assignments only).

Proof. Let $\Phi = \{\overline{a}_1 \cdot \overline{x} + b_1 \neq 0, \ldots, \overline{a}_m \cdot \overline{x} + b_m \neq 0\}$ and put $\sigma = A \mod 2$, $f = \epsilon_1 x_1 + \cdots + \epsilon_n x_n$. Then

$$f \equiv 1 - \sigma \pmod 2 \models f \neq A$$
$$\models (\overline{a}_1 \cdot \overline{x} + b_1) \cdot \ldots \cdot (\overline{a}_m \cdot \overline{x} + b_m) = 0.$$

By Theorem 4.4 in Alekhnovich-Razborov [2], the function $f \equiv 1 - \sigma \pmod 2$ is $\frac{n}{4}$-*immune*, that is, the degree of any non-zero polynomial g such that $f \equiv 1 - \sigma \pmod 2 \models g = 0$ must be at least $\frac{n}{4}$. Therefore $m \geq \frac{n}{4}$. ◀

▶ **Theorem 37.** *We work over $\mathbb{Q}$. Let $f = \epsilon_1 x_1 + \cdots + \epsilon_n x_n$, where $\epsilon_i \in \{-1, 1\}$. Then any tree-like* $\mathrm{Res}_{sw}(\mathrm{lin}_{\mathbb{Q}})$ *refutation of* $\mathsf{ImAv}(f)$ *is of size at least* $2^{\frac{n}{4}}$.

Proof. According to the definitions in Sec. 5.2 the corresponding Prover-Delayer game is on $\mathsf{ImAv}(f)$ and starts with the empty position. The game finishes at a position Φ, where $\Phi \models f - A = 0$ for some $A \in im_2(f)$.

We now define a Delayer's strategy that guarantees $\frac{n}{4}$ coins and by Lemma 30 obtain the lower bound.

The strategy is as follows. Let the position in the game be defined by a system Φ and let the branching chosen by the Prover be $g_1 \neq 0$ and $g_2 \neq 0$, where $\Phi \models g_1 + g_2 \neq 0$. Delayer does the following:

1. if $g_2 \neq 0$ is inconsistent with Φ, but $g_1 \neq 0$ is consistent with Φ, then choose $g_1 \neq 0$;
2. if $g_1 \neq 0$ is inconsistent with Φ, but $g_2 \neq 0$ is consistent with Φ, then choose $g_2 \neq 0$;
3. if none of the above holds, then leave the choice to the Prover and earn a coin.

We now prove that this strategy guarantees the required number of coins.

Suppose that the game has finished at a position Φ. The strategy of Delayer guarantees that Φ is satisfiable and Φ contradicts a clause $\langle f \neq A \rangle$ of $\mathsf{ImAv}(f)$, that is $\Phi \models f - A = 0$ for some $A \in im_2(f)$. Let $\zeta_1, \ldots, \zeta_\ell$ be the set of non-equalities in Φ, in the order they were added to Φ. Let $\Psi \subseteq \Phi$ be the set of all ζ_i, $i \in [\ell]$, such that ζ_i is not implied by previous non-equalities ζ_j, for $j < i$. Then, Delayer earns at least $|\Psi|$ coins, $\Psi \models f = A$, and by Lemma 36 we conclude that $|\Psi| \geq \frac{n}{4}$. ◀

5.4 Lower Bounds for the Pigeonhole Principle

Here we prove that every tree-like $\text{Res}_{sw}(\text{lin}_\mathbb{F})$ refutations of $\neg\text{PHP}_n^m$ must have size at least $2^{\Omega(\frac{n-1}{2})}$ (see Sec. 2.3.1 for the definition of $\neg\text{PHP}_n^m$). Together with the upper bound for dag-like $\text{Res}(\text{lin}_\mathbb{F})$ (Theorem 17) this provides a separation between tree-like and dag-like $\text{Res}_{sw}(\text{lin}_\mathbb{F})$ in the case $char(\mathbb{F}) = 0$, for formulas in CNF. The lower bound argument is comprised of exhibiting a strategy for Delayer in the Prover-Delayer game. Delayer's strategy is similar to that in [17]. However, the proof that Delayer's strategy guarantees sufficiently many coins relies on Lemma 39, which is a generalization of Lemma 3.3 in [17] for arbitrary fields. Since the proof of Lemma 3.3 in [17] for the $\mathbb{F}_2$ case does not apply to arbitrary fields, our proof is different, and uses a result from Alon-Füredi [4] on the hyperplane coverings of the hypercube.

▶ **Theorem 38.** *For every field $\mathbb{F}$, the shortest tree-like $\text{Res}_{sw}(\text{lin}_\mathbb{F})$ refutation of $\neg PHP_n^m$ has size $2^{\Omega(\frac{n-1}{2})}$.*

Proof. We prove that there exists a strategy for Delayer in the $\neg\text{PHP}_n^m$ game, which guarantees Delayer to earn $\frac{n-1}{2}$ coins. Following the terminology in [17], we call an assignment $x_{i,j} \mapsto \alpha_{ij}$, for $\alpha \in \{0,1\}^{mn}$, *proper* if it does not violate $\mathsf{Pigeons}_n^m$, namely, if it does not send two distinct pigeons to the same hole. We need to prove several lemmas before concluding the theorem.

▶ **Lemma 39.** *Let $A\overline{x} \doteq \overline{b}$ be a system of k linear non-equalities over a field $\mathbb{F}$ with n variables and where $\overline{x} = 0$ is a solution, that is, $0 \doteq \overline{b}$. If $k < n$, then there exists a non-zero boolean solution to this system.*

Proof. Let $\overline{a}_1, \ldots, \overline{a}_k$ be the rows of the matrix A. The boolean solutions to the system $A\overline{x} \doteq \overline{b}$ are all the points of the n-dimensional boolean hypercube $B_n := \{0,1\}^n \subset \mathbb{F}^n$, that are not covered by the hyperplanes $H := \{\overline{a}_1\overline{x} - b_1 = 0, \ldots, \overline{a}_k\overline{x} - b_k = 0\}$. We need to show that if $k < n$ and $0 \in B_n$ is not covered by H, then some other point in B_n is not covered by H as well. This follows from [4]:

▶ **Corollary from Alon-Füredi [4, Theorem 4].** *Let $Y(l) := \{(y_1, \ldots, y_n) \in \mathbb{F}^n \mid \forall i \in [n], 0 < y_i \leq 2, \text{ and } \sum_{i=1}^n y_i \geq l\}$. For any field $\mathbb{F}$, if k hyperplanes in $\mathbb{F}^n$ do not cover B_n completely, then they do not cover at least $M(2n-k)$ points from B_n, where*

$$M(l) := \min_{(y_1,\ldots,y_n)\in Y(l)} \prod_{1\leq i\leq n} y_i \,.$$

Thus, if $k < n$ hyperplanes do not cover B_n completely, then they do not cover at least $M(n+1)$ points. The set $Y(n+1)$ in the Corollary above consists of all tuples $(y_1, \ldots, y_n)$, where $y_i = 2$ for some $i \in [n]$ and $y_j = 1$ for $j \in [n], j \neq i$. Therefore $M(n+1) = 2$. ◀

For two boolean assignments $\alpha, \beta \in \{0,1\}^n$, denote by $\alpha \oplus \beta$ the bitwise XOR of the two assignments.

▶ **Lemma 40.** *Let $A\overline{x} \doteq \overline{b}$ be a system of k linear non-equalities over a field $\mathbb{F}$ with $n > k$ variables and let $\alpha \in \{0,1\}^n$ be a solution to the system. Then, for every choice I of $k+1$ bits in α, there exists at least one $i \in I$ so that flipping the ith bit in α results in a new solution to $A\overline{x} \doteq \overline{b}$. In other words, if $I \subseteq [n]$ is such that $|I| = k+1$, then there exists a boolean assignment $\beta \neq 0$ such that $\{i \mid \beta_i = 1\} \subseteq I$ and $A(\alpha \oplus \beta) \doteq \overline{b}$.*

Proof. Let $I \subseteq \{0,1\}^n$. Denote by $A^\star_I$ the matrix with columns $\{(1-2\alpha_i)\overline{a}_i \mid i \in I\}$, where $\overline{a}_i$ is the ith *column* of A. That is, $A^\star_I$ is the matrix A restricted to columns i with $i \in I$ and where column i flips its sign iff α_i is 1.

Assume that $\beta \in \{0,1\}^n$ is nonzero and all its 1's must appear in the indices in I, that is, $\{i \mid \beta_i = 1\} \subseteq I$. Given a set of indices $J \subseteq [n]$, denote by β_J the restriction of β to the indices in J. Similarly, for a vector $v \in \mathbb{F}^n$, v_J denotes the restriction of v to the indices in J.

▷ Claim. $A(\alpha \oplus \beta) \doteq \overline{b}$ iff $A^\star_I \beta_I \doteq \overline{b} - A\alpha$.

Proof. We prove that $A(\alpha \oplus \beta) = A^\star_I \beta_I + A\alpha$. Consider any row $\mathbf{v}$ in A, and the corresponding row $\mathbf{v}^\star_I$ in $A^\star_I$. Notice that $\mathbf{v} \cdot (\alpha \oplus \beta)$ (for "$\cdot$" the dot product) equals the dot product of $\mathbf{v}$ and $\alpha \oplus \beta$, where both vectors are restricted only to those entries in which α and β differ. Considering entries outside I, by assumption we have $\beta_{[n]\setminus I} = 0$, which implies that

$$\mathbf{v}_{[n]\setminus I} \cdot (\alpha \oplus \beta)_{[n]\setminus I} = \mathbf{v}_{[n]\setminus I} \cdot \alpha_{[n]\setminus I} \,. \tag{9}$$

On the other hand, considering entries inside I, we have

$$\mathbf{v}_I \cdot (\alpha \oplus \beta)_I = \mathbf{v}_I \cdot \alpha_I + \mathbf{v}^\star_I \cdot \beta_I \,. \tag{10}$$

Equation (10) can be verified by inspecting all four cases for the ith bits in α, β, for $i \in I$, as follows: for those indices $i \in I$, such that $\alpha_i = 1$ and $\beta_i = 0$, only $\mathbf{v}_I \cdot \alpha$ contributes to the right hand side in (10). If $\alpha_i = 1$ and $\beta_i = 1$, then by the definition of $A^\star_I$, the two summands in the right hand side in (10) cancel out. The cases $\alpha_i = 0, \beta_i = 1$ and $\alpha_i = \beta_i = 0$, can also be inspected to contribute the same values to both sides of (10).

The two equations (9) and (10) concludes the claim. ◁

We know that $A\alpha \doteq \overline{b}$, and we wish to show that for some nonzero $\beta \in \{0,1\}^n$ where $\{i \mid \beta_i = 1\} \subseteq I$, it holds that $A(\alpha \oplus \beta) \doteq \overline{b}$. By the claim above it remains to show the existence of such β where $A^\star_I \beta_I \doteq \overline{b} - A\alpha$. But notice that $\overline{b} - A\alpha \doteq 0$, since $A\alpha \doteq \overline{b}$, and that $A^\star_I \beta_I$ is a matrix of dimension $k \times (k+1)$. Therefore, by Lemma 39, the system $A^\star_I \beta_I \doteq \overline{b} - A\alpha$ has a nonzero solution, that is, there exists a $\beta \neq 0$ for which all ones are in the I entries, such that $A^\star_I \beta_I \doteq \overline{b} - A\alpha$. ◀

▶ **Lemma 41.** *Assume that a system $A\overline{x} \doteq \overline{b}$ of $k \leq \frac{n-1}{2}$ non-equalities over $\mathbb{F}$ with variables $\{x_{i,j}\}_{(i,j)\in[m]\times[n]}$ has a proper solution. Then, for every $i \in [m]$ there exists a proper solution to the system, that satisfies the clause $\bigvee_{j\in[n]} x_{i,j}$. In other words, for every pigeon, there exists a proper solution that sends the pigeon to some hole.*

Proof. We first show that if there exists a proper solution of $A\overline{x} \doteq \overline{b}$, then there exists a proper solution of this system with at most k ones. Let α be a proper solution with at least $k+1$ ones. If I is a subset of $k+1$ ones in α, then Lemma 40 assures us that some other

proper solution can be obtained from α by flipping some of these ones (note that flipping one to zero preserves the properness of assignments). Thus the number of ones can always be reduced until it is at most k.

Let α be a proper solution with at most k ones. The condition $k \leq \frac{n-1}{2}$ implies that there are $n - k \geq k + 1$ free holes. Let J be a subset of size $k + 1$ of the set of indices of free holes. Then for any $i \in [m]$ some of the bits in $I = \{(i, j) \mid j \in J\}$ can be flipped and still satisfy $A\overline{x} \doteqdot \overline{b}$, by Lemma 40. (As before, flipping from one to zero maintains the properness of the solution.) Hence, the resulting proper solution must satisfy the clause $\bigvee_{j\in[n]} x_{i,j}$. ◀

We now describe the desired strategy for Delayer.

Delayer's Strategy. Let a position in the game be defined by the system of non-equalities Φ and assume that the branching chosen by Prover is $f_0 \neq 0$ or $f_1 \neq 0$, where $\Phi \models f_0 + f_1 \neq 0$. The only objective of Delayer is to ensure that the system Φ has proper solutions. Delayer uses the opportunity to earn a coin whenever both $\Phi \cup \{f_0 \neq 0\}$ and $\Phi \cup \{f_1 \neq 0\}$ have proper solutions by leaving the choice to Prover. Otherwise, in case $\Phi \wedge \mathsf{Pigeons}_n^m \models f_i = 0$, for some $i \in \{0, 1\}$, Delayer chooses $f_{1-i} \neq 0$, which must satisfy $\Phi \wedge \mathsf{Pigeons}_n^m \models f_{1-i} \neq 0$, and so the sets of proper solutions of Φ and $\Phi \cup \{f_{1-i} \neq 0\}$ are identical.

This strategy ensures, that for every end-game position Φ, Φ has proper solutions and $\Phi \models \neg\mathsf{Holes}_n^m$. Note that Φ has the same proper solutions as Φ', obtained by throwing away from Φ all non-equalities that were added by Delayer when making a choice. Therefore, if $\Phi \models \neg\mathsf{Holes}_n^m$, then $\Phi' \wedge \mathsf{Pigeons}_n^m \models \neg\mathsf{Holes}_n^m$ and thus $|\Phi'| > \frac{n-1}{2}$ by Lemma 41.

Since $|\Phi'|$ is precisely the number of coins earned by Delayer, this gives the desired lower bound. ◀

6 Size-Width Relation and Simulation by Polynomial Calculus

In this section we prove a size-width relation for tree-like $\mathrm{Res}(\mathrm{lin}_R)$ (Theorem 44), which then implies an exponential lower bound on the size of tree-like $\mathrm{Res}_{sw}(\mathrm{lin}_R)$ refutations in terms of the principal width of refutations (Definition 5). The connection between the principal width and the degree of PC refutations for finite fields $\mathbb{F}$, together with lower bounds on degree of PC refutations from [2] on Tseitin mod p formulas and random CNFs, imply exponential lower bounds for the size of tree-like $\mathrm{Res}_{sw}(\mathrm{lin}_{\mathbb{F}})$ for these instances (Corollaries 46 and 47).

▶ **Proposition 42.** *Let $\phi = \{C_i\}_{1\leq i\leq m}$ be a set of linear clauses and $x \in vars(\phi)$. Assume that l is a linear form in the variables $vars(\phi) \setminus \{x\}$. Then, there is a $\mathrm{Res}(\mathrm{lin}_R)$ derivation π of $\{C_i \restriction_{x\leftarrow l} \vee \langle x - l \neq 0\rangle\}_{1\leq i\leq m}$ from ϕ of size polynomial in $|\phi| + |\mathsf{lm}(l)|$ and such that $\omega_0(\pi) \leq \omega_0(\phi) + 2$.*

Proof. The clause $x - l = 0 \vee \langle x - l \neq 0\rangle$ is derivable in $\mathrm{Res}(\mathrm{lin}_R)$ in polynomial in $|\mathsf{lm}(l)|$ size by Proposition 10. Assume

$$C = \left(\bigvee_{j\in[k]} f_j + a_j x + b_j^{(1)} = 0 \vee \cdots \vee f_j + a_j x + b_j^{(N_j)} = 0\right),$$

where $x \notin vars(f_i)$ and we have grouped disjuncts so that $\omega_0(C) = k$. Then we resolve these groups one by one with $x - l = 0 \vee \langle x - l \neq 0\rangle$ and after $N_1 + \ldots + N_k$ steps yield $\left(\bigvee_{j\in[k]} f_j + a_j l + b_j^{(1)} = 0 \vee \cdots \vee f_j + a_j l + b_j^{(N_j)} = 0 \vee \langle x - l \neq 0\rangle\right)$. It is easy to see that the principal width never exceeds $k + 2$ along the way. Therefore $\omega_0(\pi) \leq \omega_0(\phi) + 2$. ◀

▶ **Corollary 43.** *Let $\phi = \{C_i\}_{1\leq i\leq m}$ be a set of linear clauses and $x \in vars(\phi)$. Suppose that l is a linear form with variables $vars(\phi)\setminus\{x\}$ and that π is a $\mathrm{Res}(\mathrm{lin}_R)$ refutation of $\phi\restriction_{x\leftarrow l}\cup\{l=0\vee l=1\}$. Then, there exists a $\mathrm{Res}(\mathrm{lin}_R)$ derivation $\widehat{\pi}$ of $\langle x-l\neq 0\rangle$ from ϕ, such that $S(\widehat{\pi}) = O(S(\pi)+|\mathsf{lm}(l)|)$ and $\omega_0(\widehat{\pi})\leq \max(\omega_0(\pi)+1,\omega_0(\phi)+2)$. Additionally, there is a refutation $\widehat{\pi}'$ of $\phi\cup\{x-l=0\}$ where $\omega_0(\widehat{\pi}')\leq\max(\omega_0(\pi),\omega_0(\phi)+2)$.*

Proof. By Proposition 42 there exists a derivation π_s of

$$\{C_i\restriction_{x\leftarrow l}\vee\langle x-l\neq 0\rangle\}_{1\leq i\leq m}\cup\{l=0\vee l=1\vee\langle x-l\neq 0\rangle\}$$

from ϕ of width at most $\omega_0(\phi)+2$. Composing π_s with $\pi\vee\langle x-l\neq 0\rangle$ yields the derivation $\widehat{\pi}$ of $\langle x-l\neq 0\rangle$ from ϕ.

Moreover, by taking the derivation π_s and adding to it the axiom $x-l=0$, and then using a sequence of resolutions of π_s with $x-l=0$, we obtain a derivation of $\phi\restriction_{x\leftarrow l}\cup\{l=0\vee l=1\}$ from $\phi\cup\{x-l=0\}$. The latter derivation composed with π yields the refutation $\widehat{\pi}'$ of $\phi\cup\{x-l=0\}$ of width at most $\max(\omega_0(\pi),\omega_0(\phi)+2)$. ◀

▶ **Theorem 44.** *Let ϕ be an unsatisfiable set of linear clauses over a field $\mathbb{F}$. The following size-width relation holds for both tree-like $\mathrm{Res}(\mathrm{lin}_{\mathbb{F}})$ and tree-like $\mathrm{Res}_{sw}(\mathrm{lin}_{\mathbb{F}})$:*

$$S(\phi\vdash\bot) = 2^{\Omega(\omega_0(\phi\vdash\bot)-\omega_0(\phi))}.$$

Proof. We prove by induction on n, the number of variables in ϕ, the following:

$$\omega_0(\phi\vdash\bot)\leq\lceil\log_2 S(\phi\vdash\bot)\rceil+\omega_0(\phi)+2\,.$$

Base case: $n=0$. Thus ϕ must contain only linear clauses $a=0$, for $a\in\mathbb{F}$, and the principal width for refuting ϕ is therefore 1.

Induction step: Let π be a tree-like refutation of $\phi=\{C_1,\ldots,C_m\}$ such that $S(\pi)=S(\phi\vdash\bot)$ (i.e., π is of minimal size). Without loss of generality, we assume that the resolution rule in π is only applied to simplified clauses, that is clauses not containing disjuncts $1=0$ in case of tree-like $\mathrm{Res}(\mathrm{lin}_{\mathbb{F}})$ and not containing unsatisfiable $f=0$, $0\notin im_2(f)$ in case of tree-like $\mathrm{Res}_{sw}(\mathrm{lin}_{\mathbb{F}})$. The former can be eliminated by the simplification rule and the latter by the semantic weakening rule. By this assumption, the empty clause at the root of π is derived in tree-like $\mathrm{Res}(\mathrm{lin}_{\mathbb{F}})$ (resp. tree-like $\mathrm{Res}_{sw}(\mathrm{lin}_{\mathbb{F}})$) as a simplification (resp. weakening) of an unsatisfiable $h=0$ ($1=0$ in case of tree-like $\mathrm{Res}(\mathrm{lin}_{\mathbb{F}})$) equation, which is derived by application of the resolution rule. Denote the left and right subtrees, corresponding to the premises of $h=0$, by π_1 and π_2, respectively.

The roots of π_1 and π_2 must be of the form $f_1=0$ and $f_2=0$, respectively, where $f_1-f_2=h$. Therefore,

$$f_1=l(x_1,\ldots,x_{n-1})+a_nx_n\quad\text{and}\quad f_2=l(x_1,\ldots,x_{n-1})+a_nx_n-h\,,$$

for some $l(x_1,\ldots,x_{n-1})=\sum_{i=1}^{n-1}a_ix_i+B$, where $a_i,B\in\mathbb{F}$.

Assume without loss of generality that $a_n\neq 0$ and $S(\pi_1)\leq S(\pi_2)$. We now use the induction hypothesis to construct a narrow derivation $\pi_1^\bullet$ of $f_1=0$ such that

$$\begin{aligned}\omega_0(\pi_1^\bullet)&\leq\lceil\log_2 S(\pi_1)\rceil+1+\omega_0(\phi)+2\\&\leq\lceil\log_2 S(\pi)\rceil+\omega_0(\phi)+2\,.\end{aligned}$$

For every nonzero $A \in im_2(f_1)$ define the partial linear substitution ρ_A as $x_n \leftarrow (A - l(x_1, \ldots, x_{n-1}))a_n^{-1}$. Thus, $f_1 \restriction \rho_A = A$. The set of linear clauses

$$\phi \restriction_{\rho_A} \cup \left\{(A-l)a_n^{-1} = 0 \vee (A-l)a_n^{-1} = 1\right\} \tag{11}$$

is unsatisfiable and has $n-1$ variables, and is refuted by $\pi_1 \restriction_{\rho_A}$.

By induction hypothesis there exists a (narrow) refutation π_1^A of (11) with

$$\begin{aligned}\omega_0(\pi_1^A) &\leq \lceil \log_2 S(\pi_1 \restriction_{\rho_A}) \rceil + \omega_0(\phi) + 2 \\ &\leq \lceil \log_2 S(\pi_1) \rceil + \omega_0(\phi) + 2\,.\end{aligned}$$

By Corollary 43 there exists a derivation $\widehat{\pi}_1^A$ of $\langle l + a_n x_n \neq A \rangle$ from ϕ such that $\omega_0(\widehat{\pi}_1^A) \leq \max(\omega_0(\pi_1^A) + 1, \omega_0(\phi) + 2) \leq \lceil \log_2 S(\pi_1) \rceil + \omega_0(\phi) + 3$. By Proposition 12 there exists a derivation $\pi_1^\bullet$ of $f_1 = 0$ such that $\omega_0(\pi_1^\bullet) \leq \lceil \log_2 S(\pi_1) \rceil + \omega_0(\phi) + 3 \leq \lceil \log_2 S(\pi) \rceil + \omega_0(\phi) + 2$.

Consider the following substitution ρ: $x_n \leftarrow -l \cdot a_n^{-1}$. Then, $\pi_2|_\rho$ is a derivation of $h = 0$ from $\phi|_\rho \cup \{-l \cdot a_n^{-1} = 0 \vee -l \cdot a_n^{-1} = 1\}$, which we augment to refutation π_2' by taking composition with simplification (resp. weakening) in case of tree-like $\mathrm{Res}(\mathrm{lin}_\mathbb{F})$ (resp. tree-like $\mathrm{Res}_{sw}(\mathrm{lin}_\mathbb{F})$). By induction hypothesis there exists a refutation $\pi_2^\bullet$ of width

$$\begin{aligned}\omega_0(\pi_2^\bullet) &\leq \lceil \log_2(S(\pi_2') + 1) \rceil + \omega_0(\phi) + 2 \\ &\leq \lceil \log_2 S(\pi) \rceil + \omega_0(\phi) + 2\,,\end{aligned}$$

and thus by Corollary 43 there exists a refutation $\widehat{\pi}_2^\bullet$ of $\phi \cup \{f_1 = 0\}$ of width $\omega_0(\widehat{\pi}_2^\bullet) \leq \lceil \log_2 S(\pi) \rceil + \omega_0(\phi) + 2$. The combination of $\widehat{\pi}_2^\bullet$ and $\pi_1^\bullet$ gives a refutation of ϕ of the desired width. ◀

▶ **Theorem 45.** *Let $\mathbb{F}$ be a field and π be a* $\mathrm{Res}(\mathrm{lin}_\mathbb{F})$ *refutation of an unsatisfiable set of linear clauses ϕ. Then, there exists a $PC_\mathbb{F}$ refutation π' of (the arithmetization of) ϕ of degree $\omega(\pi)$.*

Proof. The idea is to replace every clause $C = (f_1 = 0 \vee \ldots \vee f_m = 0)$ in π by its arithmetization $a(C) := f_1 \cdot \ldots \cdot f_m$, and then augment this sequence to a valid $PC_\mathbb{F}$ derivation by simulating all the rule applications in π by several $PC_\mathbb{F}$ rule applications.

Case 1: If $D = (C \vee g_1 = 0 \vee \ldots \vee g_m = 0)$ is a weakening of C, then apply the product and the addition rules to derive $a(D) = a(C) \cdot g_1 \cdot \ldots \cdot g_m$ from $a(C)$.

Case 2: If D is a simplification of $D \vee 1 = 0$, then $a(D) = a(D \vee 1 = 0)$.

Case 3: If $D = (x = 0 \vee x = 1)$ is a a boolean axiom, then $a(D) = x^2 - x$ is an axiom of $PC_\mathbb{F}$.

Case 4: If $D = (C \vee C' \vee E \vee \alpha f + \beta g = 0)$ is a result of resolution of $(C \vee E \vee f = 0)$ and $(C' \vee E \vee g = 0)$, where C and C' do not contain the same disjuncts, then by the product and addition rules of PC we derive $a(C) \cdot a(C') \cdot a(E) \cdot f$ from $a(C \vee E \vee f = 0) = a(C) \cdot a(E) \cdot f$, and also derive $a(C) \cdot a(C') \cdot a(E) \cdot g$ from $a(C' \vee E \vee f = 0) = a(C') \cdot a(E) \cdot f$, and then apply the addition rule to derive $a(C) \cdot a(C') \cdot a(E) \cdot (\alpha f + \beta g) = a(D)$.

It is easy to see that the degree of the resulting $PC_\mathbb{F}$ refutation is at most $\omega(\pi)$. ◀

As a consequence of Theorems 44 and 45, and the relation $\omega_0 \geq \frac{1}{|\mathbb{F}|}\omega$ as well as the results from [2], we have the following:

▶ **Corollary 46.** *For every prime p there exists a constant $d_0 = d_0(p)$ such that the following holds. If $d \geq d_0$, G is a d-regular Ramanujan graph on n vertices (augmented with arbitrary orientation to its edges) and $\mathbb{F}$ is a finite field with $char(\mathbb{F}) \neq p$, then for every function σ such that $\neg TS_{G,\sigma}^{(p)} \in UNSAT$, every tree-like* $\mathrm{Res}(\mathrm{lin}_\mathbb{F})$ *refutation of $\neg TS_{G,\sigma}^{(p)}$ has size $2^{\Omega(dn)}$.*

Proof. Corollary 4.5 from [2] states that the degree of $PC_{\mathbb{F}}$ refutations of $\neg\mathrm{TS}^{(p)}_{G,\sigma}$ is $\Omega(dn)$. Theorem 45 implies that the principal width of $\mathrm{Res}(\mathrm{lin}_{\mathbb{F}})$ refutations of $\neg\mathrm{TS}^{(p)}_{G,\sigma}$ is $\Omega(\frac{1}{|\mathbb{F}|}dn) = \Omega(dn)$ and thus by Theorem 44 the size is $2^{\Omega(dn)}$. ◀

▶ **Corollary 47.** *Let $\phi \sim \mathcal{F}^{n,\Delta}_k, k \geq 3$ and $\Delta = \Delta(n)$ be such that $\Delta = o(n^{\frac{k-2}{2}})$ and let $\mathbb{F}$ be any finite field. Then every tree-like* $\mathrm{Res}(\mathrm{lin}_{\mathbb{F}})$ *refutation of ϕ has size $2^{\Omega\left(\frac{n}{\Delta^{2/(k-2)}\cdot\log\Delta}\right)}$ with probability $1-o(1)$.*

Proof. Corollary 4.7 from [2] states that the degree of $PC_{\mathbb{F}}$ refutations of $\phi \sim \mathcal{F}^{n,\Delta}_k$, where $k \geq 3$, is $\Omega(dn)$ with probability $1-o(1)$. Theorem 45 implies that the principal width of $\mathrm{Res}(\mathrm{lin}_{\mathbb{F}})$ refutations of $\phi \sim \mathcal{F}^{n,\Delta}_k$ is $\Omega(\frac{1}{|\mathbb{F}|}dn) = \Omega(dn)$ and thus by Theorem 44 the size of the refutations is $2^{\Omega(dn)}$ with probability $1-o(1)$. ◀

7 Conclusion

By the discussion in Sec. 1.1.4, for finite fileds we can take any CNF $\phi(\overline{x})$ known to be hard for $PC_{\mathbb{F}}$ (e.g. Tseitin formulas, random CNFs etc) and turn it into the linear system $R_{\phi}(\overline{x},\overline{y})$, which we can prove is hard for tree-like $\mathrm{Res}(\mathrm{lin}_{\mathbb{F}})$. It is reasonable to conjecture that these linear systems are also hard for dag-like $\mathrm{Res}(\mathrm{lin}_{\mathbb{F}})$ and to try to prove a lower bound for them. However, this would require dealing with particular systems, arising from these CNFs and, therefore, having a specific structure. Alternatively, we may turn our attention to fields $char(\mathbb{F}) = 0$: the hard instance in this case can be chosen freely among systems $L(\overline{x})$, where all coefficients are bounded by a constant: every equation in $L(\overline{x})$ can be coded as a short CNF formula, which admits short $\mathrm{Res}(\mathrm{lin}_{\mathbb{F}})$ derivations from $L(\overline{x})$. $\mathrm{Res}(\mathrm{lin}_{\mathbb{F}})$ lower bound for such a linear system would imply $\mathrm{Res}(\mathrm{lin}_{\mathbb{F}})$ CNF lower bound if $char(\mathbb{F}) = 0$ and by generalization of the proof of simulation of $\mathrm{Res}(\mathrm{lin}_{\mathbb{F}_2})$ by $\mathrm{Res}(\mathrm{lin}_{\mathbb{Q}})$ in [17] to arbitrary finite fields, this would imply CNF $\mathrm{Res}(\mathrm{lin}_{\mathbb{F}'})$ bounds for any finite field $\mathbb{F}'$.

Thus, for any field $\mathbb{F}$, the general dag-like $\mathrm{Res}(\mathrm{lin}_{\mathbb{F}})$ lower bound problem for CNF can be reduced to the following problem: find a 0-1 unsatisfiable linear system $L(\overline{x})$ over $\mathbb{Z}$ with coefficients bounded by a constant such that any $\mathrm{Res}(\mathrm{lin}_{\mathbb{Q}})$ refutation of $L(\overline{x})$ is of superpolynomial size.

Note that $\mathrm{Res}(\mathrm{lin}_{\mathbb{Q}})$ is pretty strong proof system: classical tautologies such as Pigeonhole Principle, Clique-Coclique Principle or (mod p)-Tseitin Tautologies are all easy for $\mathrm{Res}(\mathrm{lin}_{\mathbb{Q}})$[26]. Therefore, even indentifying explicit hard candidate for $\mathrm{Res}(\mathrm{lin}_{\mathbb{Q}})$ is a non-trivial problem. Linear systems in many respects are more handy to work with while indentifying hardness conditions as well as analysing structure of $\mathrm{Res}(\mathrm{lin}_{\mathbb{Q}})$ proofs.

References

1. Miklós Ajtai. The complexity of the pigeonhole principle. In *Proceedings of the IEEE 29th Annual Symposium on Foundations of Computer Science*, pages 346–355, 1988.
2. Michael Alekhnovich and Alexander A. Razborov. Lower bounds for polynomial calculus: non-binomial case. In *Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science (Las Vegas, NV, 2001)*, pages 190–199. IEEE Computer Soc., Los Alamitos, CA, 2001.
3. Yaroslav Alekseev, Dima Grigoriev, Edward A. Hirsch, and Iddo Tzameret. Semi-Algebraic Proofs, IPS Lower Bounds and the τ-Conjecture: Can a Natural Number be Negative? *Electronic Colloquium on Computational Complexity TR19-142*, 2019.
4. Noga Alon and Zoltán Füredi. Covering the Cube by Affine Hyperplanes. *Eur. J. Comb.*, 14(2):79–83, March 1993. doi:10.1006/eujc.1993.1011.

5 Paul Beame, Noah Fleming, Russell Impagliazzo, Antonina Kolokolova, Denis Pankratov, Toniann Pitassi, and Robert Robere. Stabbing Planes. In Anna R. Karlin, editor, *9th Innovations in Theoretical Computer Science Conference (ITCS 2018)*, volume 94 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 10:1–10:20, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.ITCS.2018.10.

6 Paul Beame, Henry A. Kautz, and Ashish Sabharwal. Towards Understanding and Harnessing the Potential of Clause Learning. *J. Artif. Intell. Res.*, 22:319–351, 2004. doi:10.1613/jair.1410.

7 Eli Ben-Sasson. Hard examples for the bounded depth Frege proof system. *Comput. Complexity*, 11(3-4):109–136, 2002.

8 Eli Ben-Sasson and Avi Wigderson. Short proofs are narrow—resolution made simple. *J. ACM*, 48(2):149–169, 2001.

9 Samuel R. Buss, Dima Grigoriev, Russell Impagliazzo, and Toniann Pitassi. Linear gaps between degrees for the polynomial calculus modulo distinct primes. *J. Comput. System Sci.*, 62(2):267–289, 2001. Special issue on the 14th Annual IEEE Conference on Computational Complexity (Atlanta, GA, 1999).

10 Matthew Clegg, Jeffery Edmonds, and Russell Impagliazzo. Using the Groebner basis algorithm to find proofs of unsatisfiability. In *Proceedings of the 28th Annual ACM Symposium on the Theory of Computing (Philadelphia, PA, 1996)*, pages 174–183, New York, 1996. ACM.

11 Stephen A. Cook and Robert A. Reckhow. The Relative Efficiency of Propositional Proof Systems. *J. Symb. Log.*, 44(1):36–50, 1979. This is a journal-version of Reckhow [27]. doi:10.2307/2273702.

12 Michael A. Forbes, Amir Shpilka, Iddo Tzameret, and Avi Wigderson. Proof Complexity Lower Bounds from Algebraic Circuit Complexity. In *31st Conference on Computational Complexity, CCC 2016, May 29 to June 1, 2016, Tokyo, Japan*, pages 32:1–32:17, 2016. doi:10.4230/LIPIcs.CCC.2016.32.

13 Michal Garlik and Lezsek Kołodziejczyk. Some Subsystems of Constant-Depth Frege with Parity. *ACM Transactions on Computational Logic*, 19(4), 2018. URL: https://www.mimuw.edu.pl/~lak/jansparity.pdf.

14 Joshua A. Grochow and Toniann Pitassi. Circuit Complexity, Proof Complexity, and Polynomial Identity Testing: The Ideal Proof System. *J. ACM*, 65(6):37:1–37:59, 2018. doi:10.1145/3230742.

15 Armin Haken. The intractability of resolution. *Theoret. Comput. Sci.*, 39(2-3):297–308, 1985.

16 J. Hastad. On Small-Depth Frege Proofs for Tseitin for Grids. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 97–108, Los Alamitos, CA, USA, October 2017. IEEE Computer Society. doi:10.1109/FOCS.2017.18.

17 Dmitry Itsykson and Dmitry Sokolov. Lower Bounds for Splittings by Linear Combinations. In *Mathematical Foundations of Computer Science 2014 - 39th International Symposium, MFCS 2014, Budapest, Hungary, August 25-29, 2014. Proceedings, Part II*, pages 372–383, 2014. doi:10.1007/978-3-662-44465-8_32.

18 Jan Krajícek. A feasible interpolation for random resolution. *Logical Methods in Computer Science*, 13(1), 2017. doi:10.23638/LMCS-13(1:5)2017.

19 Jan Krajícek and Igor Carboni Oliveira. On monotone circuits with local oracles and clique lower bounds. *Chicago J. Theor. Comput. Sci.*, 2018, 2018. URL: http://cjtcs.cs.uchicago.edu/articles/2018/1/contents.html.

20 Jan Krajíček, Pavel Pudlák, and Alan Woods. An exponential lower bound to the size of bounded depth Frege proofs of the pigeonhole principle. *Random Structures Algorithms*, 7(1):15–39, 1995.

21 Nathan Linial and Jaikumar Radhakrishnan. Essential covers of the cube by hyperplanes. *Journal of Combinatorial Theory, Series A*, 109:331–338, 2005.

22 A. Lubotzky, R. Phillips, and P. Sarnak. Ramanujan graphs. *Combinatorica*, 8(3):261–277, September 1988. doi:10.1007/BF02126799.

23 Jakob Nordström. On the Interplay Between Proof Complexity and SAT Solving. *ACM SIGLOG News*, 2(3):19–44, August 2015. doi:10.1145/2815493.2815497.

24 Toniann Pitassi, Paul Beame, and Russell Impagliazzo. Exponential lower bounds for the pigeonhole principle. *Comput. Complexity*, 3(2):97–140, 1993.

25 Pavel Pudlák and Russell Impagliazzo. A lower bound for DLL algorithms for k-SAT (preliminary version). In *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms, January 9-11, 2000, San Francisco, CA, USA.*, pages 128–136, 2000. URL: http://dl.acm.org/citation.cfm?id=338219.338244.

26 Ran Raz and Iddo Tzameret. Resolution over linear equations and multilinear proofs. *Ann. Pure Appl. Logic*, 155(3):194–224, 2008. doi:10.1016/j.apal.2008.04.001.

27 Robert Reckhow. *On the lengths of proofs in the propositional calculus.* PhD thesis, University of Toronto, 1976. Technical Report No . 87.

28 Grigori Tseitin. *On the complexity of derivations in propositional calculus*, pages 466–483. Studies in constructive mathematics and mathematical logic Part II. Consultants Bureau, New-York-London, 1968.

The Random-Query Model and the Memory-Bounded Coupon Collector

Ran Raz
Department of Computer Science, Princeton University, NJ, United States
ranr@cs.princeton.edu

Wei Zhan
Department of Computer Science, Princeton University, NJ, United States
weizhan@cs.princeton.edu

Abstract

We study a new model of space-bounded computation, the *random-query* model. The model is based on a branching-program over input variables $x_1, \ldots, x_n$. In each time step, the branching program gets as an input a random index $i \in \{1, \ldots, n\}$, together with the input variable x_i (rather than querying an input variable of its choice, as in the case of a standard (oblivious) branching program). We motivate the new model in various ways and study time-space tradeoff lower bounds in this model.

Our main technical result is a quadratic time-space lower bound for zero-error computations in the random-query model, for XOR, Majority and many other functions. More precisely, a zero-error computation is a computation that stops with high probability and such that conditioning on the event that the computation stopped, the output is correct with probability 1. We prove that for any Boolean function $f : \{0,1\}^n \to \{0,1\}$, with sensitivity k, any zero-error computation with time T and space S, satisfies $T \cdot (S + \log n) \geq \Omega(n \cdot k)$. We note that the best time-space lower bounds for standard oblivious branching programs are only slightly super linear and improving these bounds is an important long-standing open problem.

To prove our results, we study a memory-bounded variant of the coupon-collector problem that seems to us of independent interest and to the best of our knowledge has not been studied before. We consider a zero-error version of the coupon-collector problem. In this problem, the coupon-collector could explicitly choose to stop when he/she is sure with zero-error that all coupons have already been collected. We prove that any zero-error coupon-collector that stops with high probability in time T, and uses space S, satisfies $T \cdot (S + \log n) \geq \Omega(n^2)$, where n is the number of different coupons.

2012 ACM Subject Classification Theory of computation → Models of computation

Keywords and phrases random-query model, time-space trade-offs, branching programs

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.20

Funding *Ran Raz*: Research supported by the Simons Collaboration on Algorithms and Geometry, by a Simons Investigator Award and by the National Science Foundation grant No. CCF-1714779. *Wei Zhan*: Research supported by the Simons Collaboration on Algorithms and Geometry, by a Simons Investigator Award and by the National Science Foundation grant No. CCF-1714779.

1 Introduction

In this paper, we introduce a new model for studying time-space tradeoff lower bounds for computation, the *random-query* model. The model is based on a *branching program*. Roughly speaking, a branching program of length T and width 2^S, over input variables $x_1, \ldots, x_n$, is a directed (multi) graph with vertices arranged in $T + 1$ layers containing at most 2^S vertices each. Intuitively, each layer represents a time step and each vertex represents a memory state of the program. In layer-0 of the program, there is only one vertex, called the start vertex. Each leaf of the program is labelled by an element from $\{0, 1\}$ that we think of as the output of the program on that leaf.

In a standard branching program, every non-leaf vertex v in the program is labeled by an input variable x_v and has 2 outgoing edges, labeled by 0 and 1, going into vertices in the next layer. Intuitively, x_v is the input variable read by the vertex v. The program is called *oblivious* if all the vertices in the same layer read the same input variable. Given a branching program, the input $x_1, \ldots, x_n$ defines a computation-path, by starting from the start vertex and following in each step the edge labeled by the value of the corresponding input variable. The program outputs the label of the leaf reached by the computation path.

In the random-query model, every non-leaf vertex v in the program has $2n$ outgoing edges, labeled by each element of $\{1, \ldots, n\} \times \{0, 1\}$ exactly once. Given such a program and input $x_1, \ldots, x_n$, the computation-path starts from the start vertex and follows in each step the edge labeled by (i, x_i), where $i \in \{1, \ldots, n\}$ is random. (Intuitively, the program reads a random index $i \in \{1, \ldots, n\}$, together with the input variable x_i). As before, the program outputs the label of the leaf reached by the computation path.

1.1 Motivation

We have various motivations to study the new model. First, it seems to us an interesting model in its own right. The standard model of space-bounded computation is not always fully convincing in all settings, as it is not clear why would a machine be able to store for free the n input variables, while at the same time have a very restricted (typically, of size much smaller than n) additional memory. Moreover, in various situations the random-query model seems to be the natural one to use. Consider for example the following situation: you are at a party and you want to know if the majority of the participants prefer coffee or tea. (Assume that you know the number of participants in the party, that that number is odd and that you know all participants (or they are labeled $1, \ldots, n$)). Assume that at each time step you meet a random participant and she/he tells you their preference. How long would it take to figure out if the majority prefers coffee or tea if your memory is bounded? Another example may be a distributed setting where n players have one input variable each and they keep sending these input variables to a central player who needs to compute a Boolean function of all of them. However, the input variables arrive to the central player in an arbitrary order.

Second, we study time-space lower bounds for the random-query model in order to make progress in proving time-space lower bounds for standard (oblivious) branching programs. Time-space lower bounds for branching programs have been studied in numerous works (see for example [3, 2, 1, 5, 6]). Currently, the best time-space lower bounds for any explicit function are only slightly super linear and improving these lower bounds has been a very important and long standing open problem in computational complexity. In section 5, we show that various extensions of our results would imply such strong time-space lower bounds. Roughly speaking, our time-space lower bounds for the random-query model are proved for the case where the indices $i_1, i_2, i_3, \ldots$ of the input variables read by the program at time steps $1, 2, 3, \ldots$ are mutually independent random variables, while in order to extend these lower bounds to standard branching programs one needs to generalize the proofs to the case where some of these indices are known to be the same. Interestingly, the key component of our proof, Theorem 2, does apply to the more general case where some of the indices are known to be the same. However, the main results do not.

Third, the new model is related to several other problems that have been studied recently. First, it is related to the recent line of works on proving time-space lower bounds for learning (see for example [16, 18, 14, 11, 12, 15, 13, 4, 8, 7, 17, 9]). Indeed, computing a function $f : \{0, 1\}^n \to \{0, 1\}$ in the random-query model is equivalent to the task of distinguishing

between the following two families of distributions (which is a learning task[1]): For $x \in \{0,1\}^n$, let $\mathcal{D}_x$ be the distribution of the random variable (i, x_i), where $i \in \{1, \ldots, n\}$ is uniformly distributed. The task is to distinguish between a distribution taken from $\{\mathcal{D}_x\}_{x:f(x)=0}$ and a distribution taken from $\{\mathcal{D}_x\}_{x:f(x)=1}$, from a stream of independent samples. Second, the random-query model is similar to a recently studied model of streaming complexity, where a source of i.i.d samples of edges of a graph is considered [10]. In particular, [10] studied approximation algorithms for the maximum matching problem in that model. The main difference from our model is that they studied the space needed for approximate computation in the case where the number of samples is smaller than the number of input variables, while we study time-space tradeoffs for exact computation in the case where the number of samples may be much larger than the number of input variables.

Finally, it turns out that in the zero-error case, the random-query model is closely related to a memory-bounded variant of the coupon-collector problem, a problem that seems to be of independent interest and to the best of our knowledge has not been studied before. In our variant of the problem, the coupon collector gets a stream of random elements from the set $\{1, \ldots, n\}$ and needs to stop when she is sure with zero-error that all elements of $\{1, \ldots, n\}$ have already passed. The question is what is the time T needed when the memory size of the coupon collector is bounded by S.

1.2 Our Results

In Theorem 5, we prove that any algorithm for the zero-error coupon-collector problem that runs in time T and space S satisfies $T \cdot (S + \log n) \geq \Omega(n^2)$. This result is essentially tight. In Theorem 6, we prove that in the random-query model, any zero-error computation of XOR or Majority (or any other function with sensitivity $\Omega(n)$) that runs in time T and space S satisfies $T \cdot (S + \log n) \geq \Omega(n^2)$. The results for XOR and Majority are essentially tight (See Remarks 8 and 9 for the discussions on tightness). More generally, in the random-query model, any zero-error computation of a function with sensitivity k that runs in time T and space S satisfies $T \cdot (S + \log n) \geq \Omega(n \cdot k)$.

A very interesting open problem is to prove similar time-space lower bounds for the random-query model in the bounded-error case, rather than the zero-error case (Conjecture 1).

In Theorem 2, we prove time-space lower bounds for a special type of branching programs called set-labeled branching programs, in the random-query model. Intuitively, a set-labeled branching program is a branching-program for the coupon-collector problem, such that each vertex in the program "remembers" a set of coupons that must have been collected if that vertex was reached.

1.3 Paper Organization

The paper is organized as follows. In section 3, we prove the tight time-space lower bound for set-labeled branching programs, in the random-query model. In section 4, we reduce zero-error computation tasks in the random-query model, including the coupon-collector problem and function evaluation, to set-labeled branching programs, and hence prove tight time-space lower bounds for both problems. In section 5, we illustrate how lower bounds in the random-query model with special input distribution imply lower bounds for oblivious branching programs.

[1] Technically it is a testing task, which is easier than learning.

2 Preliminaries

For an integer n, we use $[n]$ to denote $\{1, 2, \ldots, n\}$. For any set A and an n-tuple $x \in A^n$, we use x_i to denote the i-th element of x. For any $x \in \{0,1\}^n$, let $x^{(i)}$ be the vector that is the same as x but with the i-th coordinate flipped. Given a boolean function $f : \{0,1\}^n \to \{0,1\}$, let $s(f, x)$ be the sensitivity of f at x, that is the number of coordinates $i \in [n]$ such that $f(x^{(i)}) \neq f(x)$, and let $s(f) = \max_x s(f, x)$ be the sensitivity of f.

2.1 Coupon-Collector Problem

The classical coupon-collector problem asks how large T should be, so that a uniformly random T-tuple in $[n]^T$ contains every element of $[n]$ with high probability. Generalizing the goal to a subset $A \subseteq [n]$, we have the following answer:

▶ **Proposition 1.** *Given any subset $A \subseteq [n]$, for a uniformly random $i \in [n]^T$, the probability that $A \nsubseteq \{i_1, \ldots, i_T\}$ is at most $n(1 + \log |A|)T^{-1}$.*

The proof follows directly from the fact that the expected waiting time for every element in A to appear is $n \sum_{j=1}^{|A|} j^{-1} \geq n(1 + \log |A|)$, and Markov's inequality.

In this paper, we consider a zero-error version of the coupon-collector problem. In this problem, the coupon collector could explicitly choose to stop when she is sure with zero-error that every element in A has already been collected. The results in this paper show that with bounded memory, the zero-error coupon-collector cannot stop within few (say, $O(n \log |A|)$) turns with high probability, in contrast to the proposition above.

2.2 Random-Query Model

In the random-query model, at each step $t \in \mathbb{N}_+$ a uniformly random index $i_t \in [n]$ is provided. When the problem specifies an input $x \in \{0,1\}^n$, at each step t the value of the bit $x_{i_t} \in \{0,1\}$ is also given along with the random index i_t. In this paper, we consider two cases for the joint distribution of the indices:

Independent The indices $i_1, i_2, \ldots$ are mutually independent.

Recurring The only dependencies allowed among $i_1, i_2, \ldots$ are equalities. More formally, there is a partition $p : \mathbb{Z}_+ \to \mathbb{Z}_+$, such that $i_t = i'_{p(t)}$ for every $t \in \mathbb{Z}_+$, where $i'_1, i'_2, \ldots$ are mutually independent and uniformly random over $[n]$.

For the rest of the paper, we refer to the two cases as *independent distribution* and *recurring distributions*. Notice that the independent distribution is a special case of the recurring ones. The recurring distributions are closely related to oblivious branching programs; see Section 5 for a detailed discussion.

2.3 Computational Models

The computational models we consider are based on branching programs. A *branching program* of length T and width 2^S is a directed (multi) graph with vertices arranged in $T+1$ layers containing at most 2^S vertices each. Denote the set of vertices in the i-th layer by $\mathcal{L}_i$, for $i = 0, 1, \ldots, T$. In $\mathcal{L}_0$ there is only one vertex, called the *start vertex*. Every vertex in $\mathcal{L}_T$ has out-degree 0, and is called a *leaf*. The outgoing edges from every non-leaf vertex in $\mathcal{L}_i$ only go to vertices in $\mathcal{L}_{i+1}$, for every $i < T$.

A simple branching program is one such that every non-leaf vertex has n outgoing edges, labeled with each element in $[n]$ exactly once. We consider two types of simple branching programs:

- A set-labeled branching program is a simple branching program, where every vertex v is labeled with a set $H(v) \subseteq [n]$, satisfying the following *soundness* condition: if an edge from vertex u to vertex v is labeled with $i \in [n]$, it must hold that $H(v) \subseteq H(u) \cup \{i\}$. The start vertex must be labeled with $\varnothing$.
- A branching program for the coupon-collector problem is a simple branching program such that every leaf is labeled with either "accept" or "reject".

When the indices $i_1, \ldots, i_T \in [n]$ are given, the computation path in a simple branching program starts from the start vertex, and at step t follows the edge labeled with i_t until reaching a leaf v, and outputs the label of v.

Given a function $f : \{0,1\}^n \to \{0,1\}$, a branching program computing f is one such that every non-leaf vertex has $2n$ outgoing edges, labeled with each element in $[n] \times \{0,1\}$ exactly once. Every leaf v in the program is labeled with an output $\tilde{f}_v \in \{0,1,\square\}$. When an input $x \in \{0,1\}^n$ and the indices $i_1, \ldots, i_T \in [n]$ are given, the computation path in the branching program starts from the start vertex, and at step t follows the edge labeled with (i_t, x_{i_t}) until reaching a leaf v, and outputs $\tilde{f}_v$.

In the random-query model where the indices $i_1, \ldots, i_T$ are given according to a specified distribution, we define the success of every type of branching program as follows:

- We say that a set-labeled branching program *succeeds on* $A \subseteq [n]$, if the probability that the output of the branching program $H(v) \supseteq A$ is at least $1/2$.
- For the coupon-collector problem, we say the branching program *collects* $A \subseteq [n]$ *with zero-error*, if the probability that the branching program outputs "accept" is at least $1/2$, and conditioned on outputting "accept", the probability that $\{i_1, \ldots, i_T\} \supseteq A$ is 1.
- For computing a function f, we say that the branching program *computes* f *with error* ε, if for every $x \in \{0,1\}^n$, the probability that the output of the branching program $\tilde{f}_v = f(x)$ is at least $1 - \varepsilon$. We say that the branching program *computes* f *with zero-error*, if for every $x \in \{0,1\}^n$, the probability that the output of the branching program $\tilde{f}_v \in \{0,1\}$ is at least $1/2$, and the probability that $\tilde{f}_v = 1 - f(x)$ is zero.

3 Lower Bounds for Set-Labeled Branching Programs

In this section, we prove the following theorem:

▶ **Theorem 2.** *Under the random-query model with any recurring distribution, for any set $A \subseteq [n]$, any set-labeled branching program of width $2^S \geq |A|$ that succeeds on A must have length at least $\frac{n|A|}{8S}$ for sufficiently large n.*[2]

Fix such a set-labeled branching program. We first prove an upper bound on the probability of the computation path reaching two given vertices:

▶ **Lemma 3.** *For any two vertices u, v in a set-labeled branching program, where $u \in \mathcal{L}_i$, $v \in \mathcal{L}_j$ and $i < j$. Under the random-query model with any recurring distribution,*

$$\Pr[\textit{reaching } u \wedge \textit{reaching } v] \leq \left(\frac{j-i}{n}\right)^{|H(v) \setminus H(u)|}.$$

[2] Notice that by definition, a branching program of width $2^S < |A|$ is also a branching program of width $|A|$. Therefore for smaller widths, the theorem still holds, but with an additional $\log|A|$ overhead on S. Theorems 5 and 6 work similarly.

Proof. Let $p : \mathbb{Z}_+ \to \mathbb{Z}_+$ be the partition for the recurring distribution. Let $\ell = |\{p(k) \mid i < k \le j\}|$. The indices received from the random queries between layer i and layer j are uniformly distributed over $[n]^\ell$. Let G be the random variable that represents the set of indices received between layer i and layer j. By the soundness requirement of set-labeled branching programs, if the computation path reaches u and then v, the set G corresponding to this path must satisfy $H(v) \subseteq H(u) \cup G$. Therefore,

$$\Pr[\text{reaching } u \wedge \text{reaching } v] \le \Pr[H(v) \subseteq H(u) \cup G] = \Pr[H(v) \setminus H(u) \subseteq G].$$

If $\ell < |H(v) \setminus H(u)|$ then the above probability is zero. Otherwise by (over)counting the positions where the elements of $|H(v) \setminus H(u)|$ appear and the union bound we have

$$\begin{aligned}\Pr[H(v) \setminus H(u) \subseteq G] &\le \frac{\ell!}{(\ell - |H(v) \setminus H(u)|)!} \cdot n^{-|H(v)\setminus H(u)|} \\ &\le \left(\frac{\ell}{n}\right)^{|H(v)\setminus H(u)|} \\ &\le \left(\frac{j-i}{n}\right)^{|H(v)\setminus H(u)|}.\end{aligned}$$

◀

▶ Remark 4. For the independent distribution, the above argument yield:

$$\Pr[\text{reaching } v \mid \text{reaching } u] \le \left(\frac{j-i}{n}\right)^{|H(v)\setminus H(u)|}.$$

The weaker result in Lemma 3, however, holds more generally for any recurring distribution. It is also strong enough for proving Theorem 2.

Proof for Theorem 2. Suppose the length of the set-labeled branching program is T. Define the weight of a vertex v as $W(v) = \Pr[\text{reaching } v]$. For a set of vertices $\mathcal{A}$, let $W(\mathcal{A}) = \sum_{v\in\mathcal{A}} W(v)$. Since the leaves are all in $\mathcal{L}_T$, for every $0 \le i \le T$ we have $W(\mathcal{L}_i) = 1$. The fact that the branching program succeeds on $A \subseteq [n]$ translates to:

$$\sum_{\substack{v\in\mathcal{L}_T \\ A\subseteq H(v)}} W(v) \ge 1/2. \tag{1}$$

We divide the branching program into $\frac{|A|}{2S}$ stages, each consists of a consecutive part of the layers. For every $0 \le k \le \frac{|A|}{2S}$, let i_k be the smallest index of a layer $\mathcal{L}_i$ such that

$$\sum_{\substack{v\in\mathcal{L}_i \\ |H(v)|\ge 2kS}} W(v) \ge \frac{kS}{|A|}.$$

By (1) we know such a layer must exist. Now the k-th stage consists of the layers from $\mathcal{L}_{i_k}$ to $\mathcal{L}_{i_{k+1}-1}$. Let

$$\mathcal{A}_k = \{u \in \mathcal{L}_{i_k} \mid |H(u)| \ge 2kS\}, \qquad \mathcal{B}_k = \{u \in \mathcal{L}_{i_k-1} \mid |H(u)| < 2kS\}.$$

By the definitions of i_k, we know that $W(\mathcal{A}_k) \ge kS/|A|$, $W(\mathcal{B}_k) > 1 - kS/|A|$.

Now we show that every stage contains at least $(n/3-1)$ layers. Suppose for contradiction that for some k, it holds that $i_{k+1} - i_k < n/3 - 1$. For any two vertices $u \in \mathcal{B}_k$ and $v \in \mathcal{A}_{k+1}$, by Lemma 3 we have

$$\Pr[\text{reaching } u \wedge \text{reaching } v] \le \left(\frac{i_{k+1} - i_k + 1}{n}\right)^{|H(v)\setminus H(u)|} < 3^{-2S}.$$

Therefore, applying the union bound gives:

$$\begin{aligned}
&\Pr[\text{reaching } \mathcal{L}_{i_k-1} \wedge \text{reaching } \mathcal{L}_{i_{k+1}}] \\
\leq &\Pr[\text{reaching } \mathcal{L}_{i_k-1} \setminus \mathcal{B}_k] + \Pr[\text{reaching } \mathcal{L}_{i_{k+1}} \setminus \mathcal{A}_{k+1}] + \Pr[\text{reaching } \mathcal{B}_k \wedge \text{reaching } \mathcal{A}_{k+1}] \\
\leq &1 - W(\mathcal{B}_k) + 1 - W(\mathcal{A}_{k+1}) + \sum_{\substack{u\in\mathcal{B}_k \\ v\in\mathcal{A}_{k+1}}} \Pr[\text{reaching } u \wedge \text{reaching } v] \\
< &\frac{kS}{|A|} + 1 - \frac{(k+1)S}{|A|} + 2^S \cdot 2^S \cdot 3^{-2S} < 1.
\end{aligned}$$

The second last step is because there are at most 2^S vertices in each layer, and the last step is because $2^S \geq |A|$. However, since the computation path must pass through both $\mathcal{L}_{i_k-1}$ and $\mathcal{L}_{i_{k+1}}$, the probability above must be 1, which is a contradiction.

Thus we conclude that, for n large enough, $i_{k+1} - i_k \geq n/3 - 1 \geq n/4$. Therefore,

$$T \geq \sum_{0\leq k<|A|/2S} (i_{k+1} - i_k) \geq \frac{n|A|}{8S}$$ ◀

4 Lower Bounds for Zero-error Computations under Independent Distribution

▶ **Theorem 5.** *Under the random-query model with the independent distribution, for any set $A \subseteq [n]$, any branching program for the coupon-collector problem of width $2^S \geq |A|$ which collects A with zero-error must have length at least $\frac{n|A|}{8S}$ for sufficiently large n.*

Proof. We show that for such a branching program, we can assign each vertex v with a label $H(v) \subseteq [n]$ so that the branching program is set-labeled. Let $P(v)$ be the collection of directed paths from the starting vertex to v. For every directed path p let $h(p)$ be the collection of indices labeled on the edges of p. Then we define $H(v) = \cap_{p\in P(v)} h(p)$.

The starting vertex is clearly labeled with the empty set. To check the soundness, consider an edge e from vertex u to vertex v labeled with i. For every path $p \in P(u)$, the concatenation pe is a path in $P(v)$, and $h(pe) = h(p) \cup \{i\}$. Therefore $H(v) \subseteq \cap_{p\in P(u)} h(pe) = H(u) \cup \{i\}$.

Notice that every path from the starting vertex to a leaf corresponds to a collection of indices $i_1, \ldots, i_T$, that are given with probability $n^{-T} > 0$ under the independent distribution. Since the branching program collects A with zero-error, for every path to an "accept" leaf it must holds $A \subseteq \{i_1, \ldots, i_T\}$, so every "accept" leaf v is now labeled with $H(v) \supseteq A$. Therefore, as a set-labeled branching program it succeeds on A. By Theorem 2 we know the length of the branching program is at least $\frac{n|A|}{8S}$ for sufficiently large n. ◀

▶ **Theorem 6.** *Let $f : \{0,1\}^n \to \{0,1\}$ be a boolean function with sensitivity $s(f)$. Under the random-query model with the independent distribution, any branching program of width $2^S \geq n$ which computes f with zero-error must have length at least $\frac{n\cdot s(f)}{8S}$ for sufficiently large n.*

Proof. Suppose there is a branching program $\mathcal{P}$ of width 2^S and length T that computes f with zero-error. Let $x \in \{0,1\}^n$ be an input such that $s(f) = s(f,x)$, and let $A = \{i \in [n] \mid f(x) \neq f(x^{(i)})\}$. We show below that from $\mathcal{P}$, one can extract a simple branching program $\mathcal{P}'$ for the coupon-collector problem of width at most 2^S and length T, which collects A with zero-error. Since $|A| = s(f)$, by Theorem 5 we know $T \geq \frac{n\cdot s(f)}{8S}$ for sufficiently large n.

We construct $\mathcal{P}'$ inductively to simulate $\mathcal{P}$ on input x. For vertex v in $\mathcal{P}$ we use v' to denote its corresponding vertex in $\mathcal{P}'$. The start vertex v'_0 in $\mathcal{P}'$ corresponds to the start vertex v_0 in $\mathcal{P}$. If in $\mathcal{P}$ there exists an edge from u to v labeled with (i, x_i), and u' is in $\mathcal{P}'$, then add v' to $\mathcal{P}'$ (if v' is not already there), and add an edge from u' to v' labeled with i. Finally, for every leaf v' in $\mathcal{P}'$, label v' with "accept" if $\tilde{f}_v = f(x)$, otherwise label v' with "reject".

First notice that under the independent distribution, the probability of reaching a vertex v' in $\mathcal{P}'$ is exactly the same as the probability of reaching v in $\mathcal{P}$ with the input x. Since the probability that $\mathcal{P}$ outputs $f(x)$ on input x is at least $1/2$, the probability that $\mathcal{P}'$ outputs "accept" is also at least $1/2$.

We now show that conditioned on reaching a leaf v' in $\mathcal{P}'$ labeled with "accept", it must hold that $A \subseteq \{i_1, \ldots, i_T\}$. Suppose not, then for some index $i \in A$ there is a path p' from the start vertex to v' where no edge is labeled with i. Consider the corresponding path p in $\mathcal{P}$. On input $x^{(i)}$, the computation follows the path p with non-zero probability and outputs $\tilde{f}_v = f(x) \neq f(x^{(i)})$, which contradicts the zero-error property of $\mathcal{P}$. That concludes the proof that $\mathcal{P}'$ collects A with zero-error. ◀

For the large class of functions with sensitivity $\Omega(n)$, Theorem 6 provides the quadratic time-space lower bound:

▶ **Corollary 7.** *Let f be a boolean function on n-bits with sensitivity $\Omega(n)$ (For instance, AND, XOR, Majority, s-t connectivity, etc.). Under the random-query model with the independent distribution, any branching program of width $2^S \geq n$ which computes f with zero-error must have length $\Omega(n^2/S)$.*

▶ Remark 8. Theorem 6 is tight up to logarithmic factors, in the sense that for every $m \leq n$, the function $x_1 \oplus \cdots \oplus x_m$ can be computed with zero-error within S space and $O(nmS^{-1} \log n)$ steps. We briefly sketch the algorithm here: Equally partition $[m]$ into $O(mS^{-1})$ parts, each of size $O(S)$. For each part P, use $O(n \log n)$ steps to record the values x_i for all indices $i \in P$. If any $i \in P$ does not appear within these $O(n \log n)$ steps, output $\square$. Otherwise compute the partial parity $\bigoplus_{i \in P} x_i$, and accumulate the partial parities.

As the lower bound in Theorem 6 is derived directly from Theorem 5 and further from Theorem 2, variants of the above algorithm also imply that Theorems 2 and 5 are tight up to logarithmic factors.

Similar to the case in the coupon-collector problem, the zero-error guarantee is crucial to Theorem 6, since for instance, the n-bit AND function can be computed with constant error by a branching program of length $O(n)$ and width $O(1)$. However, when specified to the parity function, the best trade-off seems to be still quadratic even in the bounded-error setting. We propose the following conjecture:

▶ **Conjecture 1.** *Under the random-query model with the independent distribution, any branching program of length T and width 2^S which computes $x_1 \oplus \cdots \oplus x_n$ with error $1/3$ must satisfy $TS = \widetilde{\Omega}(n^2)$.*

▶ Remark 9. Besides the algorithm mentioned above, there is another essentially different algorithm for computing parity (which actually computes the Hamming weight) with bounded error: Equally partition $[n]$ into $O(S/\log n)$ parts. For each part P, record the number of steps t when a pair (i, x_i) such that $i \in P$ and $x_i = 1$ is received, and finally approximate the partial sum $\sum_{i \in P} x_i$ with the integer closest to tn/T. By Chernoff bound, $T = O(n^2 S^{-1} \log^2 n)$ is enough so that the approximation of each part is wrong with probability $O(n^{-1})$.

Notice that this algorithm does not work in the zero-error setting. While the previous algorithm corresponds directly to a set-labeled branching program, it is not clear whether this approximation algorithm is related to set-labeled branching programs or not.

5 Oblivious Branching Programs and Random-Query Model

The random input model with recurring distributions is closely related to oblivious branching programs. In this section, we present two potential directions to prove strong lower bounds for oblivious branching programs, both via proving lower bounds in the random-query model. Let $\mathrm{SURJ}_{n,m} : [n]^m \to \{0,1\}$ be the surjectivity function: $\mathrm{SURJ}_{n,m}(i) = 1$ if and only $\{i_1, \ldots, i_m\} = [n]$.

▶ **Theorem 10.** *For any $m \geq 2n(\log n + 1)$, any deterministic oblivious branching program computing $\mathrm{SURJ}_{n,m}$ is also a branching program for the coupon-collector problem that collects $[n]$ with zero-error under some recurring distribution.*

Proof. Suppose at level $t-1$ the oblivious branching program reads $i_{p(t)}$, for some function $p : \mathbb{Z}_+ \to [m]$. Use p as the partition in the recurring distribution in the random-query model, then the computation of the branching program for the coupon-collector problem is exactly the same as in the oblivious branching program with a uniformly random input $i \in [n]^m$. Proposition 1 shows that the probability of $\mathrm{SURJ}_{n,m}(i) = 1$ is at least $1/2$. As the deterministic oblivious branching program always outputs correctly, as a branching program for the coupon-collector problem it succeeds with zero-error. ◀

For any function $f : \{0,1\}^n \to \{0,1\}$ and $m \geq n$, let $f^* : [n]^m \times \{0,1\}^m \to \{0,1\}$ be a partial function defined as follows: $f^*(i,y)$ is well-defined for $i \in [n]^m$ and $y \in \{0,1\}^m$, if and only if $\mathrm{SURJ}_{n,m}(i) = 1$, and whenever $i_j = i_k$ it must hold $y_j = y_k$. When $f^*(i,y)$ is well-defined, the value of $f^*(i,y)$ is $f(y_{j_1}, \ldots, y_{j_n})$, where for every $\iota \in [n]$, j_ι is some $j \in [m]$ such that $i_j = \iota$.

▶ **Theorem 11.** *Given any function $f : \{0,1\}^n \to \{0,1\}$. For any $m \geq 3n(\log n + 1)$, if there is a deterministic oblivious branching program computing f^* of length T and width 2^S (on the inputs where f^* is well-defined), then there is a branching program of the same length and width, that computes f with error $1/3$ in the random-query model under some recurring distribution.*

Proof. Add dummy levels to the oblivious branching program to double the length, such that if originally at level t the branching program reads either i_j or y_j, now it reads i_j at level $2t$ and y_j at level $2t+1$. The oblivious branching program now can be regarded as the one of length T and width 2^S that at each level t reads a pair $(i_{p(t)}, y_{p(t)})$, for some function $p : \mathbb{Z}_+ \to [m]$. Use p as the partition in the recurring distribution. For any fixed $x \in \{0,1\}^n$, the computation in the random-query model on input x is exactly the same as in the oblivious branching program with a uniformly random $i \in [n]^m$, and input $y \in \{0,1\}^m$ defined as $y_j = x_{i_j}$. For such i and y, $f^*(i,y)$ is well-defined if and only if $\mathrm{SURJ}_{n,m}(i) = 1$, and Proposition 1 indicates the probability that $f^*(i,y)$ is well-defined is at least $2/3$. Since whenever $f^*(i,y)$ is well-defined, the deterministic oblivious branching program correctly outputs $f(y_{j_1}, \ldots, y_{j_n}) = f(x)$, as a branching program under the random-query model it computes f with error $1/3$. ◀

As a corollary, if in the random-query model we were able to prove a time-space lower bound that holds under any recurring distribution, either for the zero-error coupon-collector problem, or for any bounded-error computation, we would immediately have the same lower bound (up to logarithmic factors) on deterministic oblivious branching programs.

References

1. Miklós Ajtai. A Non-linear Time Lower Bound for Boolean Branching Programs. In *40th Annual Symposium on Foundations of Computer Science, FOCS '99, 17-18 October, 1999, New York, NY, USA*, pages 60–70, 1999. doi:10.1109/SFFCS.1999.814578.
2. Miklós Ajtai. Determinism versus Non-Determinism for Linear Time RAMs (Extended Abstract). In *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing, May 1-4, 1999, Atlanta, Georgia, USA*, pages 632–641, 1999. doi:10.1145/301250.301424.
3. László Babai, Noam Nisan, and Mario Szegedy. Multiparty Protocols, Pseudorandom Generators for Logspace, and Time-Space Trade-Offs. *J. Comput. Syst. Sci.*, 45(2):204–232, 1992. doi:10.1016/0022-0000(92)90047-M.
4. Paul Beame, Shayan Oveis Gharan, and Xin Yang. Time-Space Tradeoffs for Learning Finite Functions from Random Evaluations, with Applications to Polynomials. In *Conference On Learning Theory, COLT 2018, Stockholm, Sweden, 6-9 July 2018*, pages 843–856, 2018.
5. Paul Beame, T. S. Jayram, and Michael E. Saks. Time-Space Tradeoffs for Branching Programs. *J. Comput. Syst. Sci.*, 63(4):542–572, 2001. doi:10.1006/jcss.2001.1778.
6. Paul Beame, Michael E. Saks, Xiaodong Sun, and Erik Vee. Time-space trade-off lower bounds for randomized computation of decision problems. *J. ACM*, 50(2):154–195, 2003. doi:10.1145/636865.636867.
7. Yuval Dagan and Ohad Shamir. Detecting Correlations with Little Memory and Communication. In *Conference On Learning Theory, COLT 2018, Stockholm, Sweden, 6-9 July 2018*, pages 1145–1198, 2018.
8. Sumegha Garg, Ran Raz, and Avishay Tal. Extractor-based time-space lower bounds for learning. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 990–1002, 2018. doi:10.1145/3188745.3188962.
9. Sumegha Garg, Ran Raz, and Avishay Tal. Time-Space Lower Bounds for Two-Pass Learning. In *34th Computational Complexity Conference, CCC 2019, July 18-20, 2019, New Brunswick, NJ, USA*, pages 22:1–22:39, 2019. doi:10.4230/LIPIcs.CCC.2019.22.
10. Michael Kapralov, Slobodan Mitrovic, Ashkan Norouzi-Fard, and Jakab Tardos. Space Efficient Approximation to Maximum Matching Size from Uniform Edge Samples. *CoRR*, abs/1907.05725, 2019. arXiv:1907.05725.
11. Gillat Kol, Ran Raz, and Avishay Tal. Time-space hardness of learning sparse parities. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 1067–1080, 2017. doi:10.1145/3055399.3055430.
12. Dana Moshkovitz and Michal Moshkovitz. Mixing Implies Lower Bounds for Space Bounded Learning. In *Proceedings of the 30th Conference on Learning Theory, COLT 2017, Amsterdam, The Netherlands, 7-10 July 2017*, pages 1516–1566, 2017.
13. Dana Moshkovitz and Michal Moshkovitz. Entropy Samplers and Strong Generic Lower Bounds For Space Bounded Learning. In *9th Innovations in Theoretical Computer Science Conference, ITCS 2018, January 11-14, 2018, Cambridge, MA, USA*, pages 28:1–28:20, 2018. doi:10.4230/LIPIcs.ITCS.2018.28.
14. Ran Raz. Fast Learning Requires Good Memory: A Time-Space Lower Bound for Parity Learning. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 266–275, 2016. doi:10.1109/FOCS.2016.36.
15. Ran Raz. A Time-Space Lower Bound for a Large Class of Learning Problems. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 732–742, 2017. doi:10.1109/FOCS.2017.73.
16. Ohad Shamir. Fundamental Limits of Online and Distributed Algorithms for Statistical Learning and Estimation. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 163–171, 2014.

17 Vatsal Sharan, Aaron Sidford, and Gregory Valiant. Memory-sample tradeoffs for linear regression with small error. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pages 890–901, 2019. doi:10.1145/3313276.3316403.

18 Jacob Steinhardt, Gregory Valiant, and Stefan Wager. Memory, Communication, and Statistical Queries. In *Proceedings of the 29th Conference on Learning Theory, COLT 2016, New York, USA, June 23-26, 2016*, pages 1490–1516, 2016.

Strategy-Stealing Is Non-Constructive

Greg Bodwin
Georgia Tech, Atlanta, GA, USA

Ofer Grossman
MIT, Cambridge, MA, USA

Abstract

In many combinatorial games, one can prove that the first player wins under best play using a simple but non-constructive argument called *strategy-stealing*. This work is about the complexity behind these proofs: how hard is it to actually find a winning move in a game, when you know by strategy-stealing that one exists? We prove that this problem is PSPACE-Complete already for *Minimum Poset Games* and *Symmetric Maker-Maker Games*, which are simple classes of games that capture two of the main types of strategy-stealing arguments in the current literature.

2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness

Keywords and phrases PSPACE-hard, Hex, Combinatorial Game Theory

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.21

Funding *Greg Bodwin*: Supported in part by NSF awards CCF-1717349, DMS-183932 and CCF-1909756.
Ofer Grossman: Supported by the Fannie and John Hertz Foundation fellowship, an NSF GRFP award, NSF CNS-1413920, DARPA/NJIT 491512803, Sloan Foundation 996698, and MIT/IBM W1771646. This work was done in part at the Simons Institute for the Theory of Computing.

1 Introduction

Theoretical Computer Science includes a rich theory of the complexity class TFNP, defined as the set of NP search problems where a solution always exists. The interesting subclasses of TFNP are based on simple yet non-constructive existence proofs for these solutions. For example: given a circuit

$$C : \{0,1\}^n \to \{0,1\}^{n-1},$$

one sees immediately by the Pigeonhole Principle that there exist distinct inputs x_1, x_2 with matching output $C(x_1) = C(x_2)$. But can one find such a pair of inputs computationally? This problem is complete for a complexity class PWPP $\subseteq$ TFNP, and a similar story holds for various other problems with other non-constructive proofs of solution existence.

A major motivation for TFNP as an object of study is that it gives satisfying formalizations of the natural question of whether a type of proof is constructive ("is the Pigeonhole Principle constructive?" roughly corresponds to "is P = PWPP?"). But really, some non-constructive proof methods in mathematics do not correspond to NP search problems at all. Thus, we argue, a valuable direction for research in the spirit of TFNP is to look outside TFNP itself to analyze the constructiveness of proofs in other complexity classes. This paper is about one such instance: *strategy-stealing proofs*, which are fundamental existence results in combinatorial game theory that naturally lie in PSPACE.

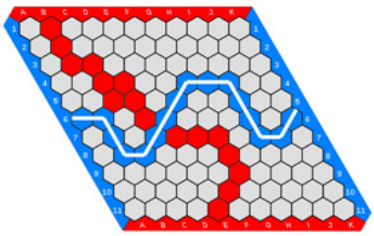


Figure 1 A Hex board with a winning configuration for Blue [1].

1.1 Combinatorial Games and Strategy-Stealing

A *combinatorial game* is a finite two-player game of perfect information. The players take turns choosing *moves* that manipulate a game board by some predefined rules, eventually reaching one out of a set of *terminal states* which determine the outcome of the game. Examples of combinatorial games include chess, go, tic-tac-toe, and some others that we will describe in detail shortly.

In general, deciding which player has a winning strategy in a combinatorial game is computationally hard. However, certain classes of games admit slick proofs that a certain player wins under best play (and thus determining who has a winning strategy in these games is not computationally hard). A famous example is the game *Hex*, in which two players named Red and Blue alternately color in hexagons in a symmetric board (pictured in Figure 1); Red wins if there is a continuous path of red hexagons connecting the top and bottom, and Blue wins if there is a continuous path of blue hexagons connecting the left and right. The *Hex Theorem* states that exactly one of the two players will achieve a winning configuration once all hexagons have been colored, so there are no draws. It was observed by Nash [15] that:

▶ **Theorem 1** ([15]). *The first player has a winning strategy in the game of Hex.*

Proof Sketch. Suppose for contradiction that the second player has a winning strategy. The first player can then make an arbitrary first move and then "steal" the winning strategy of the second player. That is, he will now pretend he is the second player, and play according to the second player's winning strategy. This will lead to a win for the first player anyways since their arbitrary initial move can only help them achieve a winning configuration. ◀

This has been dubbed the first *strategy-stealing* proof, referring to a now-broad collection of proofs that assume for contradiction that the second player can win, then repurpose the winning strategy to create a win for the first player. Another illustrative example is the game *Chomp*. Here, the game board is an $m \times n$ chocolate bar in which the top right square has been poisoned. The players alternately choose an uneaten square, and then eat that square and all remaining squares down and to the left. A player loses if they eat the poisoned square.

The following strategy stealing argument applies to Chomp:

▶ **Theorem 2** (Folklore). *The first player has a winning strategy in the game of Chomp.*

Proof Sketch. Consider the possible first move where the first player chomps off only the bottom-left-most square. There are two cases. Maybe the second player does not have a winning response, in which case the game is a win for the first player. Alternately, suppose the second player has a winning response by chomping off an $a \times b$ block. Since this block

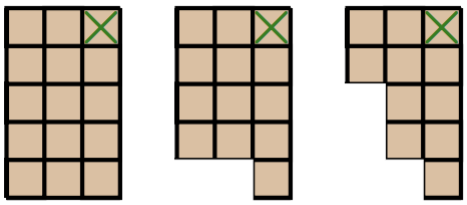


Figure 2 A valid two-move sequence from the starting position in 5×3 Chomp.

necessarily contains the bottom-left-most square, the board state is the same as if an $a \times b$ block had been chomped off with the first move of the game. It follows that this $a \times b$ chomp, instead, would have been a winning first move for the first player. ◀

Both of the proofs above *seem* non-constructive, in the sense that they do not yield an actual winning first move. Our central research question is whether this is inherent:

Are strategy-stealing proofs constructive?

To tackle this problem, we consider games that admit strategy stealing proofs, and we investigate the computational hardness of finding winning moves in such games. With this view, one can see that strategy-stealing proofs can essentially be *arbitrarily* non-constructive: for any combinatorial game X with two players P1 and P2, we can define a game X' in which the first player can decide whether he wishes to play as P1 or P2 in game X.[1] Then finding a winning move for P1 is the same as determining the winner of X, which in general is computationally hard. (We discuss this point in a little more detail in the conclusion.)

Thus, a more interesting direction is not to proceed in the maximally general case, but rather to investigate whether hardness persists in special classes of games to which strategy-stealing applies. In particular, we will study games which (to our eye) are the minimal natural classes captured by the two strategy-stealing arguments given above.

1.2 Our Results

To capture "Hex-type strategy stealing," we consider the well-studied class of *symmetric Maker-Maker games*:

▶ **Definition 3** (Symmetric Maker-Maker Game[2])**.** *In a* Maker-Maker Game*, two players alternately claim elements of a finite universe U. There are families of* winning sets *$W_1, W_2 \subseteq P(U)$; the first player wins as soon as they claim all the elements of any winning set $S \in W_1$, the second player wins as soon as they claim all the elements of any winning set $S \in W_2$, and the game is a draw if all of U is claimed without either player winning. The game is* symmetric *if W_1, W_2 are isomorphic, i.e., there is a permutation π of U and a bijection $\phi : W_1 \to W_2$ such that for all $S_1 = \{s_1, \ldots, s_k\} \in W_1$, we have $\phi(S_1) = \{\pi(s_1), \ldots, \pi(s_k)\}$.*

[1] Notice the strategy stealing argument that the first player has a winning strategy in the game X': suppose otherwise. Then we know if the first player chooses to play as P1, the second player has a winning strategy, so P2 has a winning strategy in X. But then the first player can choose to play as P2 in X' and use the winning strategy for P2 in X.

[2] This is a generalization of the usual definition: in the literature, a "Maker-Maker game" often implies $W_1 = W_2$.

Hex is a symmetric Maker-Maker (SMM) game, and indeed the proof of Theorem 1 generalizes immediately to imply that any SMM game is not a win for the second player. There are many other examples of SMM games, which will be surveyed later. In general an SMM game can be a draw under best play, but some games like Hex are *draw-free* and so a first-player win is the only remaining possibility. We associate a computational problem to these games as follows:

▶ **Definition 4** (SMMMove). *The problem* SMMMove *is defined as follows:*

- *Input: Circuits C_1, C_2, both with input wires labelled $x_1, \ldots, x_n$. C_1 and C_2 are the same up to relabelling of the wires. Call $X = \{x_1, \ldots, x_n\}$. The Maker-Maker game associated with this input is where W_1 contains any set of inputs $Y \subseteq X$ for which C_1 evaluates to true when the inputs in Y are set to true and $X \setminus Y$ to false. W_2 is defined similarly with respect to C_2.*
- *Output: any optimal first move for the first player in the associated game.*

We prove:

▶ **Theorem 5.** SMMMove *is* PSPACE*-hard,*[3] *even under the additional promise that the input defines a draw-free game with $W_1 = W_2$.*

Thus, Hex-type strategy stealing is a formally non-constructive style of proof, and additional draw-freeness results like the Hex Theorem do not generally help. To capture "Chomp-type strategy stealing," we consider:

▶ **Definition 6** (Minimum Poset Games). *In a* poset game*, two players alternately choose remaining elements of a poset P, removing the chosen element and all lesser elements at each step. A player loses if it is their turn but the poset is empty. The game is* minimum *if P has a minimum element (i.e., $m \in P$ that is comparable to and less than every other element in P).*

(For both these types of games, we refer to [4, 11] for some of their history and prior work.) Chomp is a minimum poset game, where the associated poset holds the squares of the chocolate bar, with the poisoned square removed, and squares are compared by the usual poset relation on $\mathbb{Z}^2$ (note that the bottom-left-most square is a minimum element). Theorem 2 generalizes to show that any minimum poset game is a win for the first player. Other examples of poset games, which may or may not have a minimum, include Nim, Hackendot, certain cases of Hackenbush, and many others. Computationally, we have:

▶ **Definition 7** (MPMove). *The problem* MPMove *is defined as follows:*

- *Input: a poset P (with elements and relations between them enumerated explicitly) with a minimum element.*
- *Output: any winning move for the first player in the poset game defined by P.*

▶ **Theorem 8.** MPMove *is* PSPACE*-hard.*

From a technical standpoint, both Theorems 5 and 8 are proved roughly as follows. We start with a theorem in prior work stating that it is PSPACE-hard to decide the winner in a related class of games: Hex from an arbitrary starting position [16], or a certain class of poset

[3] A straightforward algorithm solves SMMMove in polynomial space, so in some sense it is complete (ignoring subtleties in the terminology), but we will not discuss these easy upper bounds in this paper. A similar comment holds for MPMove below.

games [9]. We then apply transformations that introduce the necessary strategy-stealing properties to these games while arguing that the winner in the original game is implicitly encoded by the first player's winning move(s). In the case of minimum poset games, this is an easy extension of the theorem in [9]; we include this mostly to illustrate our conceptual goal of computationally formalizing non-constructiveness. For SMM games, the transformation is nontrivial and requires significant new ideas. Thus, the SMM result constitutes our main technical contribution.

1.3 Related Work

As mentioned, this paper is conceptually related to the study of the complexity class TFNP, defined in [14] and including notable subclasses PPAD, PPA, CLS, PPP, PWPP, PLS, among others. These classes all hold search problems in NP that admit proofs that a solution always exists. Our work is related in that our goal is to prove hardness of searching for a winning moves in games, when there are strategy-stealing proofs that one always exists. The key difference is that our problems are not in NP; there is not generally a short certificate that an optimal move is indeed the first one in some optimal strategy.

In [5], the author proves that it is PSPACE-hard to decide whether an SMM game is a win for the first player or a draw. Our work differs in that (1) to minimally generalize Hex we restrict attention to *draw-free* games, in which this decision problem is trivial, and (2) we are interested in constructively finding a winning move rather than deciding existence. Similarly related is [16], in which it is proved to be PSPACE-hard to decide whether Hex from a partially-filled board is a win for the first or second player. This can be viewed as a Maker-Maker game, but since the board is partially filled, it is not generally a symmetric Maker-Maker game and thus strategy-stealing does not apply.

There is a rich and developed theory of Maker-Maker games, poset games, and variants, most of which focuses on understanding these games under best play (rather than computational aspects of playing the games). See books [4, 11] for more information.

2 Non-Constructiveness of Strategy-Stealing

We will now prove our main results.

2.1 Symmetric Maker-Maker Games

Our first topic will be Symmetric Maker-Maker games, and eventually a proof of Theorem 5.

Examples

We first survey some famous examples of SMM games in the literature.

- Hex is an SMM game, as discussed above, which is draw-free and has non-equal winning sets.
- In the (n, k)-*Clique game*, the game board is a complete graph on n nodes and the players take turns claiming its edges. The first player to claim all edges in a k-clique wins. This is a symmetric Maker-Maker game, even with identical winning sets $W_1 = W_2$ (i.e., the underlying permutation is the identity). An interesting property of this game is that, for all k, if n is sufficiently large then the game is draw-free. This follows from *Ramsey's Theorem*, which states that any 2-coloring of the edges of the complete graph has a monochromatic clique of size $\Omega(\log n)$. Hence, for large enough n, strategy-stealing implies specifically that the first player has a winning strategy. For work on the Clique game and some natural variants, see e.g., [2, 7, 6, 3, 12].

- Tic-Tac-Toe is an SMM game, where the winning sets are the 8 possible "lines" in the 3×3 grid. This game is not draw-free.
- In (k, d) Tic-Tac-Toe, the game board is the elements of the k^d hypercube ($\{1, 2, \ldots, k\}^d$), and the winning sets $W_1 = W_2$ are the k-element subsets which are colinear in the hypercube. The *Hales-Jewett Theorem* [10] implies that for every k, if d is sufficiently large then the game is draw-free. For work on this game, see e.g., [4, 8].
- In the (n, k) *Arithmetic Progression game*, the universe is the set of integers $\{1, \ldots, n\}$, and the winning sets are any k elements that form an arithmetic progression (i.e., the difference between successive integers is equal). *Van-der-Waerden's Theorem* [17] implies that for every k, if n is sufficiently large then the game is draw-free. The Arithmetic Progression game has been studied e.g., in [13, 2, 12].

All of these games are SMM and hence admit strategy-stealing proofs that the second player does not win under best play. The problem of finding an optimal first move for the first player can thus be captured as a special case of SMMMove. All of these games are draw-free in the appropriate range of parameters (except standard Tic-Tac-Toe), and thus here they even fit the promise that the input to SMMMove defines a draw-free game. Of course, these special cases need not be as hard as SMMMove: for example, it is trivial to find a winning first move in the (n, k)-Clique game, since by symmetry of the game board all first moves are equivalent. (Perhaps a more interesting version of the Clique game computational problem is to determine the first player's optimal move on their second turn, since the game necessarily still retains the symmetry needed for a strategy-stealing argument after each player claims only one edge.)

Hardness for SMMMove

We now show computational hardness for SMMMove. We first outline the proof ideas, and then provide a full proof. Our starting point is the following result from prior work:

▶ **Definition 9** (DecisionHex). *The problem* DecisionHex *is defined as follows:*

- *Input: a partially-filled Hex board Q*
- *Output: does Red have a winning strategy in the Hex game starting from Q (assuming it is currently Red's turn to play)?*

▶ **Theorem 10** ([16]). DecisionHex *is* PSPACE*-complete.*

Our goal is to reduce DecisionHex to SMMMove. First, let us remark on why Theorem 10 does not *directly* give hardness for SMMMove, given that Hex is an SMM game. The result that DecisionHex is hard means that there exist families of positions from which deciding the winner is hard. However, these positions are not generally symmetric, so the game starting from these positions is not SMM. Additionally, a talented player *playing from the starting position* could still potentially be able to win the game while avoiding these hard settings of the game board, thus winning without ever really encountering a PSPACE-complete problem.

So, we are given a partially filled (possibly asymmetric) board Q on input to DecisionHex, representing a Hex game between Red and Blue where it is Red's turn to move, and our goal is to create a new SMM draw-free game between players First and Second that captures the structure of Q in some useful way. To build intuition, let us start with a first (incorrect) attempt at such a game G. Suppose we add two new elements to the universe called r and b. The idea will be that claiming r is choosing to play as Red in the Hex game defined by Q, and claiming b is choosing to playing as Blue. More formally, the winning sets $W_1 = W_2$ of the new SMM game would be:

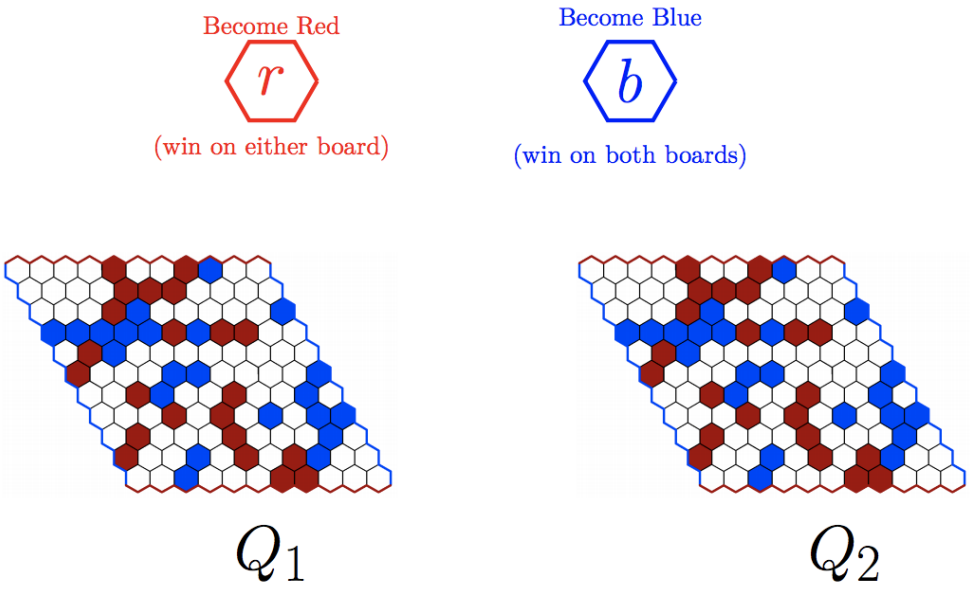


Figure 3 The universe used in our definition of an SMM game G.

- r and any set of hexagons that complete a win for Red in Q,
- b and any set of hexagons that complete a win for Blue in Q, and
- $\{r, b\}$.

If SMMMove$(G) = r$, this solves Q: the second player must claim b with their next move to block the winning set $\{r, b\}$, and then the game reduces to Q itself where First plays as Red and Second plays as Blue. Thus, if r is a winning move for First, then Q is a win for Red. Unfortunately, the other cases of the proof break down. For example: suppose the position on the board Q is such that whoever has the next move wins (so Q is a win for Red). Then it is winning for First to claim *either* r or b with their first move, meaning the output of SMMMove(G) is not very informative. Our fix is, intuitively, to amplify the game to avoid the possibility that the game winner depends on the turn order.

Proof of Theorem 5. We will prove that SMMMove is hard by reducing DecisionHex to it. Let Q an instance of DecisionHex. We will construct a Symmetric Maker-Maker game G which is draw-free and $W_1 = W_2$, such that finding a winning move in G allows us to find who has a winning strategy in Q.

The universe of our new game G will contain *two* identical copies Q_1, Q_2 of the input to DecisionHex, as well as new elements r, b like before. The winning sets $W_1 = W_2$ in G are (see Figure 3):

- r and any set of hexagons that completes a win for Red in *either* Q_1 or Q_2,
- b and any set of hexagons that completes a win for Blue in *both* Q_1 and Q_2, and
- $\{r, b\}$.

It is immediate that G is an SMM game, since the winning sets are identical. Additionally, we have:

▶ **Lemma 11.** *G is draw-free.*

Proof. Let S be any subset of the universe in G. We will show that either S or its complement S^C contains a winning set. First, if $r, b \in S$ then $\{r, b\} \subseteq S$, or if $r, b \notin S$ then $\{r, b\} \subseteq S^C$. So the nontrivial case is when S contains exactly one of r, b; let us assume without loss of generality that $r \in S, b \notin S$ (else switch the roles of S and S^C). For either board Q_i, by the Hex Theorem and the fact that the union of S_i, S_i^C covers the board, exactly one of the following two statements hold:

1. The elements of S on the board Q_i (call this S_i), combined with the elements on Q_i which are initially marked red, form a winning configuration for red on Q_i.
2. The elements of S^C on the board Q_i (call this set S_i^C), combined with the elements on Q_i which are initially marked blue, form a winning configuration for blue on Q_i.

Therefore, we conclude that in each board Q_i, either the elements of S complete a win for Red (we call Q_i a "red" board in this case), or the elements of S^C complete a win for Blue (we call Q_i a "blue" board in this case). If at least one of Q_1, Q_2 is red, then S contains r and a Red winning set. If both Q_1, Q_2 are blue, then S^C contains b and a Blue winning set on each board. In either case the lemma holds. ◀

Our goal is now to show that the winning move(s) for First in G completely determine the winner of Q. We consider two cases:

▶ **Lemma 12.** *If Blue has a winning strategy in Q, then the unique winning move for First in G is to claim b.*

Proof. We first show that claiming b is a winning move for First. In response, Second is forced to claim r to block the winning set $\{r, b\}$. First then claims an arbitrary hexagon, and then each time Second claims a hexagon on Q_1 or Q_2, First claims a hexagon on the same board to execute a winning strategy for Blue. Thus, First will have b and also a winning set for Blue on both boards, meaning First wins in G. (Note that Second will be unable to ever obtain a red winning set on either board, since it is not possible for both sides to obtain a winning configuration on any individual Hex board.)

We then show that, if First does not claim b with their first move, then it is a winning response for Second to claim b. Here we consider two cases. If First claims r, then after Second claims b, in each subsequent turn, each time First claims a hexagon on Q_1 or Q_2, Second can claim a hexagon on the same board to execute a winning strategy for Blue on that board, thus obtaining a winning set for Blue on both boards and hence winning in G. In the other case, if First claims a hexagon in (say) Q_1 with their first move, then after Second claims b, First must immediately claim r to block the winning set $\{r, b\}$. Second then claims a hexagon on Q_1 and from here this case reduces to the first one. ◀

▶ **Lemma 13.** *If Red has a winning strategy in Q, then it is not a winning move for First in G to claim b.*

Proof. Suppose that First claims b. The winning response for Second is to claim r. Without loss of generality, First then claims a hexagon in Q_1. Second then decides to permanently ignore Q_1 and focus entirely on Q_2, claiming exclusively hexagons in Q_2 for the rest of the game. Since Second is the first to move on Q_2, they can execute a winning strategy for Red on Q_2. Thus Second will eventually claim r and a winning set for Red on Q_2, meaning that Second wins in G. (Note, again, that First cannot possibly obtain a winning set in the meantime, since they cannot possibly hold a winning set for Blue on Q_2.) ◀

We now put the pieces together: after constructing the game G as described above, from Lemmas 12 and 13 we have

$$\text{SMMMove}(G) = b \quad \text{if and only if} \quad \neg\text{DecisionHex}(Q).$$

Since DecisionHex is PSPACE-complete, it follows that SMMMove is PSPACE-hard. ◀

2.2 Minimum Poset Games

Next, we prove Theorem 8. Our starting point is:

▶ **Definition 14** (DecisionPoset). *The problem* DecisionPoset *is defined as follows:*

- *Input: a poset P, described by explicitly listing its elements and the relations between them.*
- *Output: is the poset game (see Definition 6) associated to P a win for the first player under best play?*

▶ **Theorem 15** ([9]). DecisionPoset *is* PSPACE*-complete.*

We then argue:

Proof of Theorem 8. Given a poset game defined by P, generate a new poset P' by adding a new element m, defined to be less than every other element in P. We now argue that $\text{MPMove}(P') = m$ if and only if the original poset game defined by P was a win for the second player:

- Suppose P is a win for the first player. If in P' the first player claims m with their first move, then the game becomes equivalent to P with the turn order reversed. Thus claiming m is a losing move for the first player, and so $\text{MPMove}(P') \neq m$.
- Suppose P is a win for the second player. If in P' the first player claims m with their first move, then again the game is equivalent to P with the turn order reversed, so the first player has a winning strategy. This means we *can* have $\text{MPMove}(P') = m$, but since MPMove might return any winning move, we also need to rule out the possibility that any other move is winning. For this, we observe that any other first move in P' necessarily removes m and at least one other element from P', thus giving a position that can possibly be obtained after one move in P. Since P is a win for the second player, any such position must be losing for the player who creates it, and thus we have $\text{MPMove}(P') \notin P' \setminus \{m\}$, so $\text{MPMove}(P') = m$.

This completes the reduction from DecisionPoset to MPMove, and thus MPMove is PSPACE-hard. ◀

3 Open Questions

We conclude by listing some conceptual open questions left by this work.

TFPSPACE

Can we similarly analyze the computational properties of other interesting non-constructive proof techniques that lie outside of NP? Is there a satisfying theory of TFPSPACE, in analogy with TFNP?

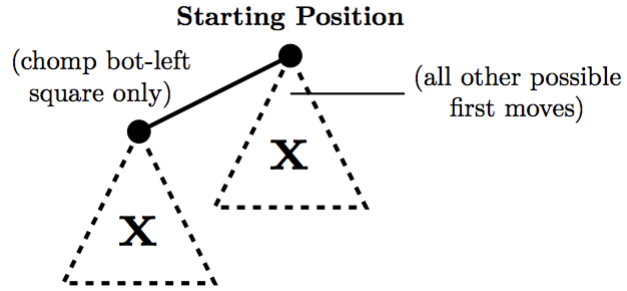


Figure 4 Due to the symmetry in the game tree of Chomp illustrated here, the value of the game tree can be expressed as X or $\neg X$, where X is the game tree after a 1×1 square has been chomped.

Bounded Computational Power

The existence of a winning strategy in a game does not necessarily shed much light on how *computationally bounded* players would play the game. To illustrate, consider the following game: player 1 declares a circuit C of their choice. Then, player 2 wins if they can declare an input x such that $C(x) = 1$. Then, player 1 then wins if *they* can declare an input x with $C(x) = 1$. If both players fail to declare such an input x, then player 2 wins. Here, there is clearly a winning strategy for the second player: if there exists an x such that $C(x) = 1$, then declare that x and win immediately; if there is no such x then player 2 also wins. However, if the players are represented by Turing machines that can only run for a polynomial amount of time, then the game is (probably) a win for player 1: for example, player 1 can pick a one way function f, compute it on some random x' of his choice to get output y, and then have the circuit C output 1 on all x such that $f(x) = y$. Under standard cryptography assumptions, player 2 will be unable to find such an x, and then player 1 will win in the next turn by declaring the x used to create the circuit.

Interestingly, for draw-free SMM games, such situations will not arise: even for computationally bounded players, it is preferable to play first, since playing an extra move is never disadvantageous. In contrast, this is not clearly true for poset games, where the wrong first move can possibly throw the game. Hence, this might be an interesting avenue to separate the computational properties of these two strategy-stealing arguments (since they are both PSPACE-hard under "best play," i.e., unbounded computational power). More generally, it would be interesting to further understand and formalize the effects of bounded computational power on various existence proofs for winning strategies in combinatorial game theory.

Generalized Strategy-Stealing

Many strategy-stealing arguments can be viewed as a reduction of the game tree to a tautology. To illustrate, the game tree of Chomp may be phrased as follows. Let X be the subgame tree from the starting chocolate bar with the bottom-left-most square removed. The proof of Theorem 2 essentially observes that the Chomp game tree is equal to X or $\neg X$, which is true as a formula (meaning a win for the first player) regardless of the value of X (see Figure 4).

Naturally, a reduction of the game tree to *any* tautology implies a first-player win, including more complicated tautologies in which multiple variables are assigned to multiple subgames. We might call this type of argument *generalized strategy-stealing*, as it extends the usual proofs

in the literature that use only X or $\neg X$. For any given tautology it is easy enough to invent an artificial game that admits a generalized strategy-stealing proof via that tautology. However, it would be interesting to find a "natural" game that admits a generalized strategy-stealing proof, using a tautology formally distinct from X or $\neg X$.

Hardness for Specific Games

While we have proved hardness for finding winning moves in game classes that include Hex and Chomp, our results do not imply hardness for Hex and Chomp specifically. In particular, it would be interesting to determine whether or not the following problem is in FP: given game board dimensions for Hex or Chomp (or basically any other game mentioned in this paper), written in unary, output a winning move for the first player. This problem is in PSPACE, but it will not be readily possible to prove it PSPACE-hard for the following reason: it is known that no unary language can be NP-complete unless P = NP; thus, a unary language complete for PSPACE would imply that P = NP or NP ≠ PSPACE, which is not known and would constitute a breakthrough in complexity theory. Thus it is unclear what hardness notion should be used to approach this question.

Other Notions of Constructiveness

We have proved that it is generally hard to find a winning move in a game, even when strategy-stealing arguments apply. Finding a good first move is one natural formalization of "constructiveness" in PSPACE, but there are others. For example, here is an open question that we have not addressed: for (say) the game Hex, does there necessarily exist a polynomial-size circuit that plays the game optimally, even if it is computationally hard to find the circuit?

References

1. Wining situation on a Hex Board. `https://en.wikipedia.org/wiki/Hex_(board_game)#/media/File:Hex-board-11x11-(2).jpg`.
2. József Beck. Van der Waerden and Ramsey type games. *Combinatorica*, 1(2):103–116, 1981.
3. József Beck. Positional games and the second moment method. *Combinatorica*, 22(2):169–216, 2002.
4. József Beck. *Combinatorial Games: Tic-Tac-Toe Theory*, volume 114. Cambridge University Press, 2008.
5. Jesper Makholm Byskov. Maker-maker and maker-breaker games are PSPACE-complete. *BRICS Report Series*, 11(14), 2004.
6. Paul Erdös and John L Selfridge. On a combinatorial game. *Journal of Combinatorial Theory, Series A*, 14(3):298–301, 1973.
7. Heidi Gebauer. On the clique-game. *European Journal of Combinatorics*, 33(1):8–19, 2012.
8. Solomon W Golomb and Alfred W Hales. Hypercube tic-tac-toe. *More Games of No Chance*, 42:167–180, 2002.
9. Daniel Grier. Deciding the winner of an arbitrary finite poset game is PSPACE-Complete. In *International Colloquium on Automata, Languages, and Programming*, pages 497–503. Springer, 2013.
10. Alfred W Hales and Robert I Jewett. Regularity and positional games. In *Classic Papers in Combinatorics*, pages 320–327. Springer, 2009.
11. Dan Hefetz, Michael Krivelevich, Miloš Stojaković, and Tibor Szabó. *Positional Games*. Springer, 2014.
12. Christopher Kusch. *Problems in Positional Games and Extremal Combinatorics*. PhD thesis, FU Berlin, 2017.

13 Christopher Kusch, Juanjo Rué, Christoph Spiegel, and Tibor Szabó. Random strategies are nearly optimal for generalized Van der Waerden games. *Electronic Notes in Discrete Mathematics*, 61:789–795, 2017.

14 Nimrod Megiddo and Christos H Papadimitriou. On total functions, existence theorems and computational complexity. *Theoretical Computer Science*, 81(2):317–324, 1991.

15 John F Nash. Some games and machines for playing them, 1952.

16 Stefan Reisch. Hex ist PSPACE-Vollständig. *Acta Informatica*, 15(2):167–191, 1981.

17 Bartel van der Waerden. Beweis einer baudetschen vermutung. *Nieuw Arch. Wisk.*, 19:212–216, 1927.

Distribution-Free Testing of Linear Functions on $\mathbb{R}^n$

Noah Fleming[1]
University of Toronto, Toronto, Canada
noahfleming@cs.toronto.edu

Yuichi Yoshida
National Institute of Informatics, Tokyo, Japan
yyoshida@nii.ac.jp

Abstract

We study the problem of testing whether a function $f\colon \mathbb{R}^n \to \mathbb{R}$ is linear (i.e., both additive and homogeneous) in the *distribution-free* property testing model, where the distance between functions is measured with respect to an unknown probability distribution over $\mathbb{R}^n$. We show that, given query access to f, sampling access to the unknown distribution as well as the standard Gaussian, and $\varepsilon > 0$, we can distinguish additive functions from functions that are ε-far from additive functions with $O\left(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon}\right)$ queries, independent of n. Furthermore, under the assumption that f is a continuous function, the additivity tester can be extended to a distribution-free tester for linearity using the same number of queries. On the other hand, we show that if we are only allowed to get values of f on sampled points, then any distribution-free tester requires $\Omega(n)$ samples, even if the underlying distribution is the standard Gaussian.

2012 ACM Subject Classification Theory of computation → Streaming, sublinear and near linear time algorithms

Keywords and phrases Property Testing, Distribution-Free Testing, Linearity Testing

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.22

Funding *Noah Fleming*: Supported by MITACS JSPS and NSERC.
Yuichi Yoshida: Supported by JSPS KAKENHI Grant Number JP17H04676 and JP18H05291.

1 Introduction

Property testing of Boolean functions studies the problem where, given query access to a function $f\colon \{0,1\}^n \to \{0,1\}$ and a parameter $\varepsilon > 0$, the goal is to distinguish with high probability the case that f satisfies some predetermined property P from the case that f is ε-far from satisfying P. That is, whether we need to change the values of $f(x)$ for at least an ε-fraction of $x \in \{0,1\}^n$ before f satisfies P. Since the seminal work by Blum, Luby and Rubinfeld [11], property testing has become a thriving field, and many properties of Boolean functions have been shown to be testable with a number of queries independent of n, including linear functions [11], low-degree polynomials [8, 26] and k-juntas [9, 10, 18]. For an introductory survey, we recommend [21].

In contrast to Boolean functions, only a few properties of functions on a Euclidean space, that is, $\mathbb{R}^n$, have been studied. For a measurable function $f\colon \mathbb{R}^n \to \mathbb{R}$, $\varepsilon > 0$, and a property P, we say that f is *ε-far* from P if

$$\Pr_{x\sim\mathcal{N}(0,I)}[f(x) \neq g(x)] > \varepsilon,$$

[1] Work done while visiting the National Institute of Informatics.

for any measurable function $g\colon \mathbb{R}^n \to \mathbb{R}$ satisfying P, where $\mathcal{N}(0, I)$ is the standard Gaussian. We say that an algorithm is a *tester* for a property P if, given query access to a measurable function $f\colon \mathbb{R}^n \to \mathbb{R}$, sampling access to the standard Gaussian, and $\varepsilon > 0$, it accepts with probability at least $2/3$ when f satisfies P, and rejects with probability at least $2/3$ when f is ε-far from P. Testability of a variety of properties has been considered, including surface area of a set [29,34], half spaces [31–33], linear separators [3], high-dimensional convexity [13], and linear k-junta [15].

Although the standard Gaussian is natural, it rarely appears in practice. In fact, we typically have little, if any, information about the underlying distribution. This raises the question of whether we can test when the underlying distribution of the data is unknown. For a measurable function $f\colon \mathbb{R}^n \to \mathbb{R}$, $\varepsilon > 0$, a distribution $\mathcal{D}$ over $\mathbb{R}^n$, and a property P, we say that f is *ε-far* from P with respect to $\mathcal{D}$ if

$$\Pr_{x\sim\mathcal{D}}[f(x) \neq g(x)] > \varepsilon,$$

for any measurable function $g\colon \mathbb{R}^n \to \mathbb{R}$ satisfying P. We say that an algorithm is a *distribution-free tester* for a property P if, given query access to a measurable function $f\colon \mathbb{R}^n \to \mathbb{R}$, sampling access to an *unknown* distribution $\mathcal{D}$ over $\mathbb{R}^n$ as well as the standard Gaussian, and $\varepsilon > 0$, it accepts with probability at least $2/3$ when f satisfies P, and rejects with probability at least $2/3$ when f is ε-far from P with respect to $\mathcal{D}$. Distribution-free property testing is an attractive model because it makes minimal assumptions on the environment, and models the scenario most often occurring in practice.

We say that a function $f\colon \mathbb{R}^n \to \mathbb{R}$ is *additive* if $f(x) + f(y) = f(x + y)$ for any $x, y \in \mathbb{R}^n$. In this work, we consider distribution-free testing of additivity of functions $f\colon \mathbb{R}^n \to \mathbb{R}$ and show the following.

▶ **Theorem 1.** *There exists a one-sided error distribution-free tester for additivity of $f\colon \mathbb{R}^n \to \mathbb{R}$ with $O\left(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon}\right)$ queries.*

Previously no algorithm was known even when the underlying distribution $\mathcal{D}$ is the standard Gaussian. As there is a trivial lower bound of $\Omega\left(\frac{1}{\varepsilon}\right)$, the query complexity of our tester is almost tight.

We say that a function $f\colon \mathbb{R}^n \to \mathbb{R}$ is *homogeneous* if $cf(x) = f(cx)$ for any $x \in \mathbb{R}^n$ and $c \in \mathbb{R}$. A function that is both additive and homogeneous is said to be *linear*. Although additivity and linearity are equivalent for functions over finite groups, there are (pathological) functions $f\colon \mathbb{R}^n \to \mathbb{R}$ that are additive but not homogeneous. Hence, the testability of additivity does not immediately imply the testability of linearity. However, when the input function is guaranteed to be continuous, we can also test linearity.

▶ **Theorem 2.** *Suppose that the input function is guaranteed to be continuous. Then, there exists a one-sided error distribution-free tester for linearity with $O\left(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon}\right)$ queries.*

It is also natural to assume that we can get values of the input function only on sampled points. Specifically, we say that a (distribution-free) tester is *sample-based* if it accesses the input function $f\colon \mathbb{R}^n \to \mathbb{R}$ through points sampled from the distributions $\mathcal{D}$ and $\mathcal{N}(0, I)$. We show a strong lower bound for sample-based testers.

▶ **Theorem 3.** *Any sample-based tester for the linearity of functions $f\colon \mathbb{R}^n \to \mathbb{R}$ requires $\Omega(n)$ samples, even when $\mathcal{D} = \mathcal{N}(0, I)$.*

This lower bound is tight; it is not difficult to see that $O(n)$ samples suffices to test linearity. Indeed $O(n)$ samples will, with high probability, contain n linearly independent vectors. The evaluations of f on these vectors uniquely determines the linear function. This theorem shows

a sharp contrast between query-based and sample-based testers for properties of functions on a Euclidean space. We note that we can show the same lower bound for testing additivity with an almost identical proof.

1.1 Related Work

The question of property testing first appeared (implicitly) in the work of Blum, Luby and Rubinfeld [11]. Among the problems that they studied was linearity testing. Their algorithm, now famously known as the BLR test, has played a key role in the design of probabilistically checkable proofs [2, 5, 25] and this connection was some of the early motivation for the field of property testing. Since the original paper, the parameters of the BLR test have been extensively refined. Much of this work focused on reducing the amount of randomness, due to this being a key parameter in probabilistically checkable proofs, as well as analyzing the rejection probability (see [36] for a survey). Another line of works considered the testing linearity over more general domains. The works of [7, 11, 35] showed that the BLR test can be used to test the linearity of any function with $f\colon G \to H$ for finite groups G and H with $O(1/\varepsilon)$ queries. Following this, a body of work [1, 17, 19, 27] constructed testers for linearity of functions $f\colon S \to \mathbb{R}$, where S is a finite subset of rational numbers, and the distance is measured with respect to the uniform distribution over S. See [28] for a survey. These results were phrased in terms of approximate self-testing and correcting programs. In this setting the queries to f return a finite approximation of $f(x)$. Although these results are arguably the most related to our work, our proof differs significantly from theirs and instead takes inspiration from the original BLR test.

Distribution-free testing (for graph properties) was first defined by Goldreich et al. [22], though the first distribution-free testers for non-trivial properties appeared much later in the work of Halevy and Kushilevitz [23]. Subsequently, distribution-free testers have been considered for a variety of Boolean functions including low-degree polynomials, dictators, and monotone functions [23], k-juntas [6,12,23,30], conjunctions, decision lists, and linear threshold functions [20], monotone and non-monotone monomials [16], and monotone conjunctions [14, 20]. However, to our knowledge the only (partial) distribution-free tester for a class of functions on the Euclidean space is due to Harms [24] who gave an efficient tester for half spaces, that is, functions $f\colon \mathbb{R}^n \to \{0,1\}$ of the form $f(x) = \operatorname{sgn}(w^\top x - \theta)$ for some $w \in \mathbb{R}^n$ and $\theta \in \mathbb{R}$, over any rotationally invariant distribution.

1.2 Proof Technique

The construction of our tester for additivity will be done in two steps. First, we construct a constant-query tester for additivity over the standard Gaussian distribution $\mathcal{N}(0, I)$. Our tester will accept linear functions with probability 1, and so the majority of the work is in showing that if the test accepts the given function $f\colon \mathbb{R}^n \to \mathbb{R}$ with high probability, then f is close to an additive function. To do so, we show that if f passes a series of tests then there exists a related function $g\colon \mathbb{R}^n \to \mathbb{R}$, defined from f, which is additive. Furthermore, if f is linear then $f = g$. The definition of g will allow us to obtain query access to it with high probability, and so we can simply estimate the distance between f and g. At a high-level, this is somewhat similar to the BLR test, however operating over $\mathcal{N}(0, I)$ rather than the uniform distribution presents its own set of non-trivial challenges. We discuss these, as well as the definition of g at the start of Section 3.1.

It is fairly straightforward to generalize this tester for additivity to a distribution-free tester. To do so, we run the additivity tester for the standard Gaussian, except that testing the distance between f and g will now be done using samples from the unknown $\mathcal{D}$. This crucially relies on our ability to draw samples from the standard Gaussian.

Any additive function $f\colon \mathbb{R}^n \to \mathbb{R}$ is linear over the rationals, meaning that $f(qx) = qf(x)$ for every $q \in \mathbb{Q}$. Therefore, in order to test linearity it remains to test whether this holds also for irrationals. Assuming that f is continuous we are able to modify our tester to show that this implies that the additive function g is continuous as well. We then leverage the fact that any continuous additive function is linear in order to obtain our linearity tester.

To prove Theorem 3, the lower bound on sample-based testers for linearity, we construct two distributions, one supported on linear functions, and the other supported on functions which are far from linear. Consider drawing a function f from one of these two distributions with equal probability. By Yao's minimax principle it suffices to show that any deterministic algorithm which receives n samples from $\mathcal{N}(0, I)$, together with their evaluations on f, is unable to distinguish, with high probability, which of the two distribution f came from. To construct the distribution on linear functions, we sample $w \sim \mathcal{N}(0, I)$ and return $f(x) := w^\top x$. Our distribution on functions which are far from linear is designed so that any function f from this distribution satisfies $f(x + y) \neq f(x) + f(y)$ with probability 1 over $x, y \sim \mathcal{N}(0, I)$. To do so, for every $x \in \mathbb{R}^n$ we sample ε_x from a one-dimensional Gaussian and return $f(x) := w^\top x + \varepsilon_x$. It is not difficult to show that such functions are far from linear.

1.3 Organization

The remainder of the paper is organized as follows. In Section 2 we review several useful facts about probability distributions. In Section 3 we develop our distribution-free tester for additivity by first constructing a tester for additivity over the standard Gaussian in Section 3.1. We generalize this tester to the distribution-free setting in Section 3.2 and to a tester for linearity in Section 4. Finally, we end with our lower bound on the sampling model in Section 5.

2 Preliminaries

Let $\mathcal{D}$ and $\mathcal{D}'$ be probability distributions on the same domain Ω. Then, the *total variation distance* between them, denoted by $\mathrm{d_{TV}}(\mathcal{D}, \mathcal{D}')$, is defined as

$$\mathrm{d_{TV}}(\mathcal{D}, \mathcal{D}') := \frac{1}{2} \int_\Omega |\mathcal{D}(x) - \mathcal{D}'(x)| dx.$$

The *Kullback-Leibler divergence* (or *KL-divergence*) of $\mathcal{D}'$ from $\mathcal{D}$, denoted $\mathrm{d_{KL}}(\mathcal{D} \| \mathcal{D}')$, is defined as

$$\mathrm{d_{KL}}(\mathcal{D} \| \mathcal{D}') = \int_\Omega \mathcal{D}(x) \log \left(\frac{\mathcal{D}(x)}{\mathcal{D}'(x)} \right) dx.$$

We will use the KL-divergence to upper bound the total variation distance, using the following inequality.

▶ **Theorem 4** (Pinsker's Inequality). *Let $\mathcal{D}$ and $\mathcal{D}'$ be probability distributions on the same domain Ω. Then,*

$$\mathrm{d_{TV}}(\mathcal{D}, \mathcal{D}') \leq \sqrt{\frac{1}{2} \mathrm{d_{KL}}(\mathcal{D} \| \mathcal{D}')}.$$

The following allows us to bound the KL-divergence between two Gaussian distributions.

▶ **Lemma 5.** *Let $\mathcal{D} = \mathcal{N}(\mu_1, \Sigma_1)$ and $\mathcal{D}' = \mathcal{N}(\mu_2, \Sigma_2)$ be multivariate Gaussian distributions with $\mu_1, \mu_2 \in \mathbb{R}^n$ and invertible $\Sigma_1, \Sigma_2 \in \mathbb{R}^{n \times n}$. Then,*

$$\mathrm{d}_{\mathrm{KL}}(\mathcal{D} \| \mathcal{D}') = \frac{1}{2}\left(\log\left(\frac{\det \Sigma_2}{\det \Sigma_1}\right) + \mathrm{tr}\left((\Sigma_2)^{-1}\Sigma_1\right) - n + (\mu_2 - \mu_1)^\top \Sigma_2^{-1}(\mu_2 - \mu_1)\right).$$

We record a useful lemma about total variation distance of Gaussians with shared covariance matrices.

▶ **Lemma 6.** *Consider two Gaussian distributions $\mathcal{N}(\mu_1, \Sigma), \mathcal{N}(\mu_2, \Sigma)$ with shared invertible covariance matrices $\Sigma \in \mathbb{R}^{n \times n}$. Then $\mathrm{d}_{\mathrm{TV}}(\mathcal{N}(\mu_1, \Sigma), \mathcal{N}(\mu_2, \Sigma)) \leq \phi$ holds if $\|\mu_1 - \mu_2\|_2 \leq 2\phi/\sqrt{\|\Sigma^{-1}\|_2}$.*

Proof. Denote $\mu := \mu_1 - \mu_2$. By Lemma 5, $\mathrm{d}_{\mathrm{TV}}(\mathcal{N}(\mu_1, \Sigma), \mathcal{N}(\mu_2, \Sigma)) = \sqrt{\frac{1}{4}\mu^\top \Sigma^{-1}\mu}$. Now, because Σ is PSD, $\mu^\top \Sigma^{-1}\mu \leq \|\mu\|_2^2 \|\Sigma^{-1}\|_2$, where $\|\cdot\|_2$ is the spectral matrix norm. Therefore, we have $\mathrm{d}_{\mathrm{TV}}(\mathcal{N}(\mu_1, \Sigma), \mathcal{N}(\mu_2, \Sigma)) \leq \frac{1}{2}\|\mu\|_2 \sqrt{\|\Sigma^{-1}\|_2} \leq \phi$. ◀

3 Testing Additivity

In this section, we develop our distribution-free tester for additivity. For convenience, we first describe a simpler tester for additivity over the standard Gaussian distribution $\mathcal{N}(0, I)$ in Section 3.1. Then, in Section 3.2, we describe how to generalize this algorithm to test additivity over an unknown distribution.

3.1 Tester for the Standard Gaussian

Our goal in this section is to design a constant-query tester for the additivity of a measurable function $f \colon \mathbb{R}^n \to \mathbb{R}$ over the standard Gaussian.

▶ **Theorem 7.** *There exists a one-sided error $\Omega\left(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon}\right)$-query tester for additivity over the standard Gaussian.*

At a high-level, our tester consists of two steps. First, we test whether f satisfies additivity over a set of samples drawn from the distribution. If f passes this test, then we conclude that there must be an additive function $g \colon \mathbb{R}^n \to \mathbb{R}$, which is a self-corrected version of f. Second, by testing the value of f on a correlated set of points, we are able to get query access to g with high probability, and therefore we can simply estimate the distance between f and g. Our tester relies on the fact that it has one-sided error: if f is additive then our test passes with probability 1. Otherwise, if f is non-additive and the second step passes with high probability, then f and g must be close.

The first step is inspired by the BLR test. Indeed, the evaluation of the function g at a point p is defined as the (weighted) majority value of $f(p - x) + f(x)$ over all $x \sim \mathcal{N}(0, I)$ (where, $f(p - x) + f(x)$ is weighted according to the probability of drawing $x \sim \mathcal{N}(0, I)$). However, there are some significant challenges in generalizing the BLR test to the standard Gaussian, the most obvious of which is that unlike the uniform distribution, every point in the support of the distribution does not have equal probability. In particular, $p - x$ is not distributed as $x \sim \mathcal{N}(0, I)$ for fixed $p \neq 0$. In order to overcome this, we exploit the fact that for additive functions f, we have $f(x) = qf(x/q)$ for every rational q. This allows us to restrict attention to a small ball $\mathrm{B}(0, 1/r)$ of radius $1/r$ centered at the origin. Then, for $p \in \mathrm{B}(0, 1/r)$, $p - x$ is approximately distributed as x for small enough $1/r$. Thus, we get around the issue of unevenly weighted points by defining g within $\mathrm{B}(0, 1/r)$, and then extrapolating to define g over $\mathbb{R}^n$.

Algorithm 1 Standard Gaussian Additivity Tester.

Given: Query access to $f\colon \mathbb{R}^n \to \mathbb{R}$, sampling access to the distribution $\mathcal{N}(0, I)$;
Reject if TestAdditivity(f) returns **Reject**;
for $N_1 := O(1/\varepsilon)$ *times* **do**
  Sample $p \sim \mathcal{N}(0, I)$;
  Reject if $f(p) \neq$ Query-$g(p, f)$ or if Query-$g(p, f)$ returns **Reject**.
Accept.

Algorithm 2 Subroutines.

Procedure TestAdditivity(f)
  Given: Query access to $f\colon \mathbb{R}^n \to \mathbb{R}$, sampling access to the distribution $\mathcal{N}(0, I)$;
  for $N_2 := O(1)$ *times* **do**
    Sample $x, y, z \sim \mathcal{N}(0, I)$;
    Reject if $f(-x) \neq -f(x)$;
    Reject if $f(x - y) \neq f(x) - f(y)$;
    Reject if $f\left(\frac{x-y}{2}\right) \neq f\left(\frac{x-z}{2}\right) + f\left(\frac{z-y}{2}\right)$;
  Accept.
Procedure Query-$g(p, f)$
  Given: $p \in \mathbb{R}^n$, query access to $f\colon \mathbb{R}^n \to \mathbb{R}$, sampling access to $\mathcal{N}(0, I)$;
  $N_2' := O(\log \frac{1}{\varepsilon})$;
  Sample $x_1, \ldots, x_{N_2'} \sim \mathcal{N}(0, I)$;
  Reject if there exists $i, j \in [N_2']$ such that
   $f(p/k_p - x_i) + f(x_i) \neq f(p/k_p - x_j) + f(x_j)$;
  return $k_p\,(f(p/k_p - x_1) + f(x_1))$.

Concretely, we will define g as follows. First, let r be a sufficiently large integer ($r = 50$ suffices). For each point $p \in \mathbb{R}^n$ define

$$k_p := \begin{cases} 1 & \text{if } \|p\|_2 \leq 1/r, \\ \lceil r \cdot \|p\|_2 \rceil & \text{if } \|p\|_2 > 1/r. \end{cases}$$

Now, define $g\colon \mathbb{R}^n \to \mathbb{R}$ as

$$g(p) := k_p \cdot \operatorname*{maj}_{\mathcal{N}(0,I)} \left[f\left(\frac{p}{k_p} - x\right) + f\left(x\right) \right],$$

where $\operatorname{maj}_{\mathcal{N}(0,I)}$ is the *weighted majority function* where a value $f(p/k_p - x) + f(x)$ is weighted according to its probability mass under $x \sim \mathcal{N}(0, I)$. Observe that either $p \in \mathrm{B}(0, 1/r)$, or $g(p)$ first maps p to a point p/k_p in $\mathrm{B}(0, 1/r)$. The value of g is the most likely value (according to $\mathcal{N}(0, I)$) of $f(p/k_p - x) + f(x)$. If f is close to additive, then taking this majority should allow us to correct for the errors in f.

An equivalent definition of g which will be useful is the following. For $p \in \mathbb{R}^n$ let P_p be the Lebesgue measurable function such that $\int_A P_p(x)dx$ gives the probability (over $\mathcal{N}(0, I)$) that $f(p/k_p - x) + f(x)$ takes value in A. Then g is defined as $g(p) := \operatorname{argmax}_x P_p(x)$ if $P_p(x) \geq 1/2$.

Our algorithm is given in Algorithm 1, which uses subroutines given in Algorithm 2. The Query-g subroutine allows us to obtain query access to g with high probability, while the TestAdditivity subroutine tests the conditions that we require in order to prove that g is additive.

▶ **Lemma 8.** *If* TestAdditivity(f) *accepts with probability at least* $1/10$*, then* g *is a well-defined, additive function, and furthermore,* $\Pr_{x\sim\mathcal{N}(0,I)}[g(p) \neq k_p(f(p/k_p - x) + f(x))] < 1/2$.

We first prove Theorem 7 assuming that Lemma 8 holds.

Proof of Theorem 7. First, observe that if f is an additive function then Algorithm 1 always accepts. It is immediate that TestAdditivity(f) always accepts. To see that it also passes the remaining tests, observe that by additivity, $k_p\left(f(p/k_p - x) + f(x)\right) = k_p f(p/k_p) = f(p)$, where the final inequality holds because $k_p \in \mathbb{Z}$ and by homogeneity over the rationals $f(qx) = qf(x)$ for every $q \in \mathbb{Q}$.

We now show that if f is ε-far from additive functions, then Algorithm 1 rejects with probability at least $2/3$. If TestAdditivity(f) accepts with probability at most $1/10$, we can reject f with probability at least $1 - 1/10 > 2/3$. Hence, we assume that TestAdditivity(f) accepts with probability at least $1/10$. Then by Lemma 8, the function g is additive and hence f is ε-far from g. Now, we want to bound the probability that Step 2 of Algorithm 1 passes.

First, we bound the probability that Query-$g(p, f)$ fails to recover the value of $g(p)$. That is, we bound the probability that $f(p/k_p - x_i) + f(x_i) = f(p/k_p - x_j) + f(x_j)$ for all $i, j \in [N_2']$, but $g(p) \neq k_p\left(f(p/k_p - x_i) + f(x_i)\right)$. By Lemma 8, the probability that we draw N_2' points which satisfy this is at most $2^{-N_2'} \leq \varepsilon/2$ by choosing the hidden constant in N_2' to be large enough. Therefore, the probability that we correctly recover $g(p)$ is at least $1 - \varepsilon/2$.

Now that we have established that we can obtain query access to g with high probability, it remains to show that we can test whether f and g are close. Indeed, the probability that Step 2 of Algorithm 1 fails to reject is at most

$$\begin{aligned}
&\left(\Pr_{p\sim\mathcal{N}(0,I)}[f(p) = g(p) \vee \text{Query-}g(p,f) \text{ fails to correctly recover } g(p)]\right)^{N_1} \\
\leq&\left(1 - \Pr_{p\sim\mathcal{N}(0,I)}[f(p) \neq g(p)] + \Pr_{p\sim\mathcal{N}(0,I)}[\text{Query-}g(p,f) \text{ fails to correctly recover } g(p)]\right)^{N_1} \\
<&\left(1 - \frac{\varepsilon}{2}\right)^{N_1} < \frac{1}{10},
\end{aligned}$$

by choosing the hidden constant in N_1 to be large enough. Therefore, Algorithm 1 rejects with probability at least $1 - 1/10 > 2/3$. ◀

It remains to prove Lemma 8 showing that if Algorithm 1 succeeds, then g is an additive function with high probability.

3.1.1 Additivity of the Function g

First, we record the basic, but useful observation that if the TestAdditivity subroutine passes then each of its tests hold with high probability over $\mathcal{N}(0, I)$.

▶ **Lemma 9.** *If* TestAdditivity*(f) accepts with probability at least* $1/10$*, then*

$$\Pr_{x,y\sim\mathcal{N}(0,I)}[f(x-y)=f(x)-f(y)]\geq\frac{99}{100}, \tag{1}$$

$$\Pr_{x\sim\mathcal{N}(0,I)}[f(-x)=-f(x)]\geq\frac{99}{100}, \tag{2}$$

$$\Pr_{x,y,z\sim\mathcal{N}(0,I)}\left[f\left(\frac{x-y}{2}\right)=f\left(\frac{x-z}{2}\right)+f\left(\frac{z-y}{2}\right)\right]\geq\frac{99}{100}. \tag{3}$$

Proof. Suppose for contradiction that at least one of (1), (2), and (3) does not hold. We here assume that (1) does not hold as other cases are similar.

We accept only when all the sampled pairs (x,y) satisfy $f(x+y)=f(x)+f(y)$. By setting the hidden constant in N_2 to be large enough, this happens with probability at most

$$\left(1-\Pr_{x,y\sim\mathcal{N}(0,I)}[f(x+y)\neq f(x)+f(y)]\right)^{N_2}<\left(\frac{99}{100}\right)^{N_2}<\frac{1}{10},$$

which is a contradiction. ◀

In order to argue that g is additive, we will first argue that g is additive on points within the tiny ball $\mathrm{B}(0,1/r)$. To do so, we will crucially use the fact that $p-x$ is distributed approximately as $x\sim\mathcal{N}(0,I)$ if $\|p\|_2$ is small. By Lemma 6 we have a bound on the total variation distance between x and $x+p$.

▷ Claim 10. Let $p\in\mathbb{R}^n$ satisfying $\|p\|_2\leq k/r$ for some $k\in\mathbb{Z}^{>0}$. Then $\mathrm{d_{TV}}(\mathcal{N}(0,I),\mathcal{N}(p,I))\leq k/100$.

Proof. By Lemma 6, for $\mathrm{d_{TV}}(\mathcal{N}(0,I),\mathcal{N}(p,I))\leq k/100$ it is enough that p satisfies $\|0-p\|_2\leq 2k/(100\sqrt{\|I\|_2})$. Because $\|p\|_2\leq k/r=2k/100=2k/(100\sqrt{\|I\|_2})$. ◁

After arguing that g is additive in $\mathrm{B}(0,1/r)$, it will follow that g is additive elsewhere because g is defined by extrapolating the value of g within this ball. Therefore, we will focus on proving the additivity of g within $\mathrm{B}(0,1/r)$.

▶ **Lemma 11.** *Suppose that* (1) – (3) *of Lemma 9 hold. For every* $p,q\in\mathbb{R}^n$ *with* $\|p\|_2,\|q\|_2,\|p+q\|_2\leq 1/r$ *it holds that* $g(p+q)=g(p)+g(q)$*.*

The proof of this lemma will crucially rely on the following two lemmas, which say that the conclusions of Lemma 9 hold with high probability even when one of the points are fixed to a point $\mathrm{B}(0,1/r)$. A consequence of this is that g is well-defined.

▶ **Lemma 12.** *Suppose that* (1) – (3) *of Lemma 9 hold, then* g *is well-defined, and for every* $p\in\mathbb{R}^n$ *with* $\|p\|_2\leq 1/r$*,*

$$\Pr_{x\sim\mathcal{N}(0,I)}[g(p)=f(p-x)+f(x)]\geq\frac{9}{10}.$$

Proof. Fix a point $p\in\mathbb{R}^n$ with $\|p\|_2\leq 1/r$. We will bound the following probability.

$$A:=\Pr_{x,y\sim\mathcal{N}(0,I)}[f(p-x)+f(x)=f(p-y)+f(y)].$$

Observe that

$$\begin{aligned}
A &= \Pr_{x,y\sim\mathcal{N}(0,I)}[f(x)-f(y)\neq f(p-y)-f(p-x)] \\
&\leq \Pr_{x,y\sim\mathcal{N}(0,I)}[f(x)-f(y)\neq f(x-y)] + \Pr_{x,y\sim\mathcal{N}(0,I)}[f(x-y)\neq f(p-y)-f(p-x)] \\
&< \frac{1}{100} + \Pr_{x,y\sim\mathcal{N}(0,I)}[f(x-y)\neq f(p-y)-f(p-x)] && \text{(By Lemma 9)}
\end{aligned}$$

It remains to bound the second term. Intuitively, because $x-p, y-p \sim \mathcal{N}(-p, I)$ and $p \approx 0$, the random variables $p-x$ and $p-y$ should be distributed similarly to x and y. Indeed,

$$\begin{aligned}
&\Pr_{x,y\sim\mathcal{N}(0,I)}[f(x-y)\neq f(p-y)-f(p-x)] \\
&= \Pr_{x,y\sim\mathcal{N}(0,I)}[f(x-p+p-y)\neq f(p-y)-f(p-x)] \\
&= \Pr_{x,y\sim\mathcal{N}(-p,I)}[f(x-y)\neq f(-y)-f(-x)] \\
&\leq \Pr_{x,y\sim\mathcal{N}(0,I)}[f(x-y)\neq f(-y)-f(-x)] + 2\,\mathrm{d_{TV}}\left(\mathcal{N}(0,I),\mathcal{N}(-p,I)\right) \\
&\leq \Pr_{x,y\sim\mathcal{N}(0,I)}[f(x-y)\neq f(x)-f(y)] + \frac{2}{100} + 2\Pr_{x\sim\mathcal{N}(0,I)}[f(-x)\neq f(x)] && \text{(Claim 10)} \\
&\leq \frac{3}{100} + \frac{2}{100} = \frac{5}{100}. && \text{(By (1) and (2) in Lemma 9)}
\end{aligned}$$

Plugging this into our previous bound on A, we can conclude that

$$A \geq 1 - \left(\frac{1}{100} + \frac{5}{100}\right) = 1 - \frac{6}{100} > \frac{9}{10}.$$

Next, we bound A above in terms of the probability that $g(p) \neq f(p-x)+f(x)$. Define $P_p\colon \mathbb{R}^n \to \mathbb{R}^+$ to be the bounded Lebesgue-measurable function such that $\int_B P_p(x)dx$ is the probability that $f(p-x)+f(x)$ takes value in the (measurable) set B. By Hölder's inequality with $p=1, q=\infty$ we have

$$A = \int_{\mathbb{R}} P_p^2(x)dx \leq \|P_p\|_\infty \int_{\mathbb{R}} P_p(x)dx = \|P_p\|_\infty,$$

where the last equality follows because P_p is a density and $\int_{\mathbb{R}} P_p(x)dx = 1$ holds. Therefore,

$$\frac{9}{10} \leq A \leq \|P_p\|_\infty.$$

Because $\operatorname{argmax}_x P_p(x) \geq 9/10 > 1/2$, we have $g(p) = \operatorname{argmax}_x P_p(x)$ and hence $\Pr_{x\sim\mathcal{N}(0,I)}[g(p) = f(p-x)+f(x)] \geq 9/10$. ◀

The following lemma is essentially condition (3) of Lemma 9 with two fixed points.

▶ **Lemma 13.** *Suppose that* (1) – (3) *of Lemma 9 hold then, for every* $p, q \in \mathbb{R}^n$ *with* $\|p\|_2, \|q\|_2, \|p+q\| \leq 1/r$,

$$\Pr_{x,y,z\sim\mathcal{N}(0,I)}\left[g(p+q) \neq f\left(p-\frac{x-z}{2}\right) + f\left(q-\frac{z-y}{2}\right) + f\left(\frac{x-y}{2}\right)\right] \leq \frac{2}{10}.$$

Proof. Fix a pair of points $p, q \in \mathbb{R}^n$ with $\|p\|_2, \|q\|_2 \le 1/r$. We can bound the probability

$$\begin{aligned}
&\Pr_{x,y,z\sim\mathcal{N}(0,I)}\left[g(p+q) \neq f\Big(p-\frac{x-z}{2}\Big)+f\Big(q-\frac{z-y}{2}\Big)+f\Big(\frac{x-y}{2}\Big)\right]\\
\le&\Pr_{x,y,z\sim\mathcal{N}(0,I)}\left[g(p+q) \neq f\Big(p+q-\frac{x-y}{2}\Big)+f\Big(\frac{x-y}{2}\Big)\right]\\
&+\Pr_{x,y,z\sim\mathcal{N}(0,I)}\left[f\Big(p+q-\frac{x-y}{2}\Big) \neq f\Big(p-\frac{x-z}{2}\Big)+f\Big(q-\frac{z-y}{2}\Big)\right]
\end{aligned}$$

To bound the first term, observe that if $x, y \sim \mathcal{N}(0, I)$, then the random variable $(x-y)/2$ is also distributed according to $\mathcal{N}(0, I)$. Furthermore, because $\|p+q\|_2 \le 1/r$, we can apply Lemma 12 and conclude that

$$\Pr_{x,y,z\sim\mathcal{N}(0,I)}\left[g(p+q) \neq f\Big(p+q-\frac{x-y}{2}\Big)+f\Big(\frac{x-y}{2}\Big)\right] \le \frac{1}{10}.$$

To bound the second term, observe that

$$\begin{aligned}
&\Pr_{x,y,z\sim\mathcal{N}(0,I)}\left[f\Big(p+q-\frac{x-y}{2}\Big) \neq \Big(p-\frac{x-z}{2}\Big)+\Big(q-\frac{z-y}{2}\Big)\right]\\
=&\Pr_{x,y,z\sim\mathcal{N}(0,I)}\left[f\left(\frac{(2q+y)-(x-2p)}{2}\right) \neq \left(\frac{(2q+y)-z}{2}\right)+\left(\frac{z-(x-2p)}{2}\right)\right]\\
=&\Pr_{\substack{x\sim\mathcal{N}(-2p,I)\\ y\sim\mathcal{N}(2q,I)\\ z\sim\mathcal{N}(0,1)}}\left[f\Big(\frac{y-x}{2}\Big) \neq \Big(\frac{y-z}{2}\Big)+\Big(\frac{z-x}{2}\Big)\right]\\
\le&\Pr_{x,y,z\sim\mathcal{N}(0,I)}\left[f\Big(\frac{x-y}{2}\Big) \neq \Big(\frac{x-z}{2}\Big)+\Big(\frac{z-y}{2}\Big)\right]+\mathrm{d_{TV}}\left(\mathcal{N}(0,I),\mathcal{N}(-2p,I)\right)\\
&+\mathrm{d_{TV}}\left(\mathcal{N}(0,I),\mathcal{N}(2q,I)\right)\\
\le&\frac{1}{100}+\frac{2}{100}+\frac{2}{100}=\frac{5}{100}. \qquad \text{(By Lemma 9 and Claim 10)}
\end{aligned}$$

Combining both of these bounds, we have $\Pr_{x,y,z\sim\mathcal{D}}[g(p+q) \neq f(p-\frac{x-z}{2})+f(q-\frac{z-y}{2})+f(\frac{x-y}{2})] \le 1/10+5/100 \le 2/10$. ◀

The additivity of g within $\mathrm{B}(0, 1/r)$ is an immediate consequence of these two lemmas.

Proof of Lemma 11. Let $p, q \in \mathbb{R}^n$ be any pair of points satisfying $\|p\|_2, \|q\|_2, \|p+q\|_2 \le 1/r$. Our aim is to show that $g(p+q) = g(p)+g(q)$. By a union bound over Lemmas 12 and 13, the probability that $x, y, z \sim \mathcal{N}(0, I)$ simultaneously satisfy

1. $g(p+q) = f(p-\frac{x-z}{2})+f(q-\frac{z-y}{2})+f(\frac{x-y}{2})$,
2. $g(p) = f(p-\frac{x-z}{2})+f(\frac{x-z}{2})$,
3. $g(q) = f(q-\frac{z-y}{2})+f(\frac{z-y}{2})$,
4. $f(\frac{x-y}{2}) = f(\frac{x-z}{2})-f(\frac{z-y}{2})$

is at least $1-(2/10+2\cdot 1/10+1/10) > 0$. Here we are using the fact that $((x-y)/2)$ is distributed as $\mathcal{N}(0, I)$. Fixing such a triple (x, y, z), we conclude that

$$\begin{aligned}
g(p+q) &= f\Big(p-\frac{x-z}{2}\Big)+f\Big(q-\frac{z-y}{2}\Big)+f\Big(\frac{x-y}{2}\Big)\\
&= g(p)+g(q)+f\Big(\frac{x-y}{2}\Big)-f\Big(\frac{x-z}{2}\Big)-f\Big(\frac{z-y}{2}\Big)\\
&= g(p)+g(q).
\end{aligned}$$

Therefore g is additive within $\mathrm{B}(0, 1/r)$. ◀

Finally, we argue that g is additive everywhere. Intuitively this should be true because the values of g on points outside of $\mathrm{B}(0,1/r)$ are defined by extrapolating the values of g on points within $\mathrm{B}(0,1/r)$, where we know g is additive. For the proof, it will be useful to record the following fact.

▶ Fact 14. Provided that (1) – (3) of Lemma 9 hold then, for every $p \in \mathbb{R}^n$ with $\|p\|_2 \leq 1/r$ and $c \in \mathbb{Z}^{>0}$, we have $g(p) = cg(p/c)$.

Proof. Observe that $g(p) = g((c/c)p) = g\left(\sum_{i=1}^{c} p/c\right) = \sum_{i=1}^{c} g(p/c) = c \cdot g(p/c)$, where the third equality follows by Lemma 11, noting that $\|kp/c\|_2 \leq 1/r$ for every $k \in [c-1]$. ◀

Proof of Lemma 8. Fix a pair of points $p, q \in \mathbb{R}^n$, we will argue that $g(p+q) = g(p) + g(q)$. Recall that $g(p) := k_p g(p/k_p)$, $g(q) := k_q g(q/k_q)$, and $g(p+q) := k_{p+q} g((p+q)/k_{p+q})$. Then,

$$g(p)+g(q) = k_p \cdot g\left(\frac{p}{k_p}\right) + k_q \cdot g\left(\frac{p}{k_q}\right) = k_p k_q k_{p+q} \cdot g\left(\frac{p}{k_p k_q k_{p+q}}\right) + k_p k_q k_{p+q} \cdot g\left(\frac{p}{k_p k_q k_{p+q}}\right),$$

where the second equality follows by Fact 14, noting that $k_p, k_q, k_{p+q} \in \mathbb{Z}^{>0}$ and so $p/k_p, q/k_q \in \mathrm{B}(0,1/r)$. Furthermore, because $p/(k_p k_q k_{p+q}), q/(k_p k_q k_{p+q})$, $(p+q)/(k_p k_q k_{p+q}) \in \mathrm{B}(0,1/r)$, we can apply Lemma 11 to obtain

$$\begin{aligned} k_p k_q k_{p+q}\left(g\left(\frac{p}{k_p k_q k_{p+q}}\right) + g\left(\frac{p}{k_p k_q k_{p+q}}\right)\right) &= k_p k_q k_{p+q} \cdot g\left(\frac{p+q}{k_p k_q k_{p+q}}\right) \\ &= k_{p+q} \cdot g\left(\frac{p+q}{k_{p+q}}\right) \\ &= g(p+q), \end{aligned}$$

where the second equality follows by Fact 14, noting that $k_p k_q \in \mathbb{Z}^{>0}$ and $(p+q)/k_{p+q} \in \mathrm{B}(0,1/r)$.

Finally, by Lemma 12, g is well-defined within $\mathrm{B}(0,1/r)$. Because g is defined by extrapolating from its value within this ball, it is well-defined everywhere. ◀

▶ Remark 15. This tester (and the same proof) will in fact work over any Gaussian $\mathcal{N}(0,\Sigma)$ for arbitrary covariance matrix $\Sigma \in \mathbb{R}^{n \times n}$ by setting the value of r to be $50\sqrt{\|\Sigma^{-1}\|_2}$.

3.2 Distribution-Free Tester

In this section, we prove Theorem 1 by adapting our tester for additivity over the standard Gaussian (Algorithm 1) to a distribution-free tester.

Assuming that we are able to draw samples from the standard Gaussian (or in fact any Gaussian), the modification to Algorithm 1 is straight forward. Indeed, we will only have to modify Algorithm 1, the two subroutines will remain the same. Let $\mathcal{D}$ be our unknown distribution by which we will measure the distance of f to an additive function. The high-level idea is to first run the TestAdditivity subroutine over the standard Gaussian. If it passes, then we know that with high probability g is additive. We can obtain query access to $g(p)$ (with high probability) as before by sampling points $x \sim \mathcal{N}(0,I)$ and checking that the values of $k_p(f(p/k_p - x) + f(x))$ agree for all of the x that we sample. To test whether f and g are ε-far according to $\mathcal{D}$ it suffices to sample points $p \sim \mathcal{D}$ and check whether $f(p)$ and $g(p)$ agree.

Our algorithm is given in Algorithm 3. We stress that both subroutines TestAdditivity and Query-$g(p_i)$ are being performed over $\mathcal{N}(0,I)$, i.e., they do not use $\mathcal{D}$.

Algorithm 3 Distribution-Free Additivity Tester.

Given: query access to $f : \mathbb{R}^n \to \mathbb{R}$, sampling access to an unknown distribution $\mathcal{D}$, and sampling access to $\mathcal{N}(0, I)$;

1 **Reject** if TestAdditivity(f) returns **Reject**;
2 **for** $N_3 := O(1/\varepsilon)$ *times* **do**
3 Sample $p \sim \mathcal{D}$;
4 **Reject** if $f(p) \neq$ Query-$g(p, f)$ or if Query-$g(p, f)$ returns **Reject**.
5 **Accept**.

Proof of Theorem 1. The proof is nearly identical to the proof of Theorem 7. Again, observe that if f is an additive function then Algorithm 3 always accepts.

It remain to show that if f is ε-far from additive functions, then Algorithm 3 rejects with probability at least 2/3. If TestAdditivity(f) accepts with probability at most 1/10, we can reject f with probability at least $1 - 1/10 > 2/3$. Hence, we assume that TestAdditivity(f) accepts with probability at least 1/10. By Lemma 8, the function g is additive and hence f is ε-far from g. Note that the probability that Query-$g(p, f)$ fails to correctly recover $g(p)$ is at most $\varepsilon/2$ by the same argument as before. It remains to bound the probability that Step 3 fails to reject, which is

$$\left(\Pr_{p\sim\mathcal{N}(0,I)}[f(p) = g(p) \vee \text{Query-}g(p) \text{ fails to correctly recover } g(p)]\right)^{N_3} < \left(1 - \frac{\varepsilon}{2}\right)^{N_3} < \frac{1}{10}$$

by choosing the hidden constant in N_3 to be large enough, by the same argument as before. Therefore, Algorithm 3 rejects with probability at least $1 - 1/10 > 2/3$. ◀

4 Testing Linearity of Continuous Functions

In this section, we prove Theorem 2 by adapting the tester from the previous section (Algorithm 3) to test whether f is linear, given that f is a continuous function.

We would like to argue that if f is continuous and Algorithm 3 passes then g is in fact a linear function with high probability. However, in order to exploit continuity, we need f to satisfy $f(-x) = -f(x)$ for every $x \in \mathbb{R}^n$. First, we will show how to argue that g is linear assuming that $f(-x) = -f(x)$. After that, we will handle the case when this property does not hold.

▶ **Lemma 16.** *If $f : \mathbb{R}^n \to \mathbb{R}$ is a continuous function satisfying $f(-x) = -f(x)$ and the assumptions of Lemma 8 hold, then the function g is linear.*

The proof will rely on the following claim which was originally proved by Darboux in 1875.

▷ Claim 17. Any additive function $f : \mathbb{R}^n \to \mathbb{R}$ which is continuous at a point $x_0 \in \mathbb{R}^n$ is a linear function.

Proof. First, it is well-known that any additive function which is continuous at a point is continuous everywhere (see e.g., [4]). Next, we argue that the continuity of f implies that $f(rx) = rf(x)$ for every $r \in \mathbb{R}$ and $x \in \mathbb{R}^n$. Because f is additive, this homogeneity holds for every $r \in \mathbb{Q}$, so it suffices to assume that r is irrational.

Fix $x \in \mathbb{R}^n$ and irrational r. Then for any $\zeta > 0$, we can always find $\tilde{r} \in \mathbb{Q}$ such that $|\tilde{r} - r| < \zeta$ and $\|\tilde{r}x - rx\|_2 < \zeta$. Now, by the continuity of f, for any $\xi > 0$ there exists $\zeta > 0$ such that whenever $\|\tilde{r}x - rx\|_2 < \zeta$, we have $|f(\tilde{r}x) - f(rx)| < \xi$. Now, take a sequence $\{\xi_i\}_i$ with $\xi_i \to 0$ and consider the corresponding sequence $\{\zeta_i\}_i$ with $\zeta_i \to 0$. Let $\{\tilde{r}_i\}_i$ with $r_i \in \mathbb{Q}$ be the sequence of approximations such that $|\tilde{r}_i - r| \leq \zeta_i$ and $\|\tilde{r}_i x - rx\|_2 \leq \zeta_i$. Then,

$$|f(rx) - rf(x)| \leq |f(rx) - f(\tilde{r}_i x)| + |f(\tilde{r}_i x) - rf(x)| \leq \xi_i + |\tilde{r}_i f(x) - rf(x)| \leq \xi_i + \zeta_i |f(x)|.$$

Because $\zeta_i, \xi_i \to 0$, $|f(rx) - rf(x)| \to 0$ and so $f(rx) = rf(x)$. ◁

With this claim in hand, we are ready to prove Lemma 16.

Proof of Lemma 16. Let f be a continuous function satisfying $f(-x) = -f(x)$. By Lemma 8, the function g is additive. Conditioned on this event, we will show that the continuity of f implies that g is linear as well. To do so, we will argue that g is continuous at the origin and then appeal to Claim 17 to conclude that g is linear.

Let B be a ball of mass $1/2$ (with respect to $\mathcal{N}(0, I)$) centred at the origin. Let $\{p_i\}_i$ be any sequence of points with $p_i \in B$, $\|p_i\|_2 \leq 1/r$ and $p_i \to 0$. Now, let $\{x_i\}_i$ be a sequence of points such that $g(p_i) = f(p_i - x_i) + f(x_i)$ and $x_i \in B$. Such a sequence exists because, by Lemma 8 $\Pr_{x \sim \mathcal{N}(0,I)}[g(x) = f(p_i - x) + f(x)] \geq 1/2$ and so for every p_i there must exist such an x_i in B.

Let S be the ball centred at the origin with twice the radius of B. As S is compact and f is continuous, f is uniformly continuous on S. Thus for every $\xi > 0$, there exists $\zeta > 0$ such that $|f(p_i - x_i) - f(-x_i)| = |f(p_i - x_i) + f(x_i)| < \xi$ whenever $\|(p_i - x_i) + x_i\|_2 < \zeta$. Now, take a sequence $\{\xi_i\}_i$ with $\xi_i \to 0$ and consider the corresponding sequence $\{\zeta_i\}_i$. As $p_i \to 0$, for every i, there exists j such that $\|(p_j - x_j) + x_j\|_2 < \zeta_i$ which in particular implies that $|g(p_j)| = |f(p_j - x_j) + f(x_j)| < \xi_i$. Thus, $g(p_i) \to 0$, and g is continuous at the origin. By Claim 17, we can conclude that g is a linear function. ◀

Now we consider the case when $f(-x) \neq -f(x)$ for some x. Luckily, in this case we can *force* f to satisfy $f(-x) = -f(x)$. To do so, we test whether f is $\varepsilon/2$-far from satisfying this property. If it is, then we reject f, otherwise, we can replace f with a function f' guaranteed to satisfy this property, by defining

$$f'(x) := \frac{f(x) - f(-x)}{2}.$$

We then continue to work over f' rather than f. Our modified algorithm is given in Algorithm 4, which uses Algorithm 5 as a subroutine.

▷ Claim 18. If FORCENEGATIVITY$(f, \mathcal{D})$ accepts with probability at least $1/10$, then $\Pr_{x \sim \mathcal{D}}[f(x) = f'(x)] \geq 1 - \varepsilon$.

Proof. Suppose for contradiction that $\Pr_{x \sim \mathcal{D}}[f(x) = f'(x)] \leq 1 - \varepsilon$. Observe that for a point $x \in \mathbb{R}$, $f'(x) \neq f(x)$ iff $f(-x) \neq -f(x)$. Therefore, by choosing the hidden constant in N_5 to be large enough, the probability that all the sampled points x satisfy $f(x) = -f(x)$ is at most

$$\left(\Pr_{x \sim \mathcal{D}}[f(-x) = f(x)]\right)^{N_5} < (1 - \varepsilon)^{N_5} \leq \frac{1}{10},$$

which is a contradiction. ◁

Algorithm 4 Distribution-Free Linearity Tester.

Given: query access to a continuous $f\colon \mathbb{R}^n \to \mathbb{R}$, sampling access to an unknown distribution $\mathcal{D}$, and sampling access to $\mathcal{N}(0, I)$;

Reject if FORCENEGATIVITY$(f, \mathcal{D})$ returns **Reject**;
Let f' be the returned function;
Reject if TESTADDITIVITY(f') returns **Reject**;
for $N_4 := O(1/\epsilon)$ *times* **do**
    Sample $p \sim \mathcal{D}$;
    Reject if $f'(p) \neq$ QUERY-$g(f', p)$ or if QUERY-$g(f', p)$ returns **Reject**.
Accept.

Algorithm 5 Force Negativity Subroutine.

Procedure FORCENEGATIVITY$(f, \mathcal{D})$
    Given: query-Access to $f\colon \mathbb{R}^n \to \mathbb{R}$ and sampling access to an unknown distribution $\mathcal{D}$;
    for $N_5 := O(1/\epsilon)$ *times* **do**
        Sample $x \sim \mathcal{D}$;
        Reject if $f(-x) \neq -f(x)$;
    Return a function $f'\colon \mathbb{R}^n \to \mathbb{R}$ where $f'(x) := \frac{f(x)-f(-x)}{2}$;

Therefore if FORCENEGATIVITY$(f, \mathcal{D})$ accepts with probability at least $1/10$, f and f' are $\varepsilon/2$-close. Furthermore, because f is continuous and f' is the sum of continuous functions, f' is continuous as well, and so we can proceed with f' in place of f.

Proof of Theorem 2. First, observe that if f is linear then $f = f'$ and Algorithm 4 always accepts.

Now, we show that if f is ε-far from linear functions, then Algorithm 4 rejects with probability at least $2/3$. If either the TESTADDITIVITY subroutine or the FORCENEGATIVITY subroutine passes with probability at most $1/10$, we can reject f with probability at least $1 - 1/10 > 2/3$. Hence, we assume both the subroutines pass with probability at least $1/10$. Then by Lemma 18, f is $\varepsilon/2$-close to f', which means that f' is $\varepsilon/2$-far from linear. Also by Lemma 16, because f' is continuous and satisfies $f'(-x) = -f'(x)$, the function g is linear, and so f' is $\varepsilon/2$-far from g. Therefore, Algorithm 4 rejects f with probability at least $1 - 1/10 > 2/3$. ◀

5 Lower Bounds on Testing Linearity in the Sampling Model

In this section, we prove Theorem 3, that is, we show without query access, any tester requires a linear number of samples in order to test linearity and additivity over the standard Gaussian. We note that we can obtain the same lower bound for testing additivity just by replacing linearity with additivity in the proof.

By Yao's minimax principle it suffices to construct two distributions, $\mathcal{D}_{\text{yes}}$ over linear functions and $\mathcal{D}_{\text{no}}$ over functions which are (with probability 1) $1/3$-far from linear such that any deterministic n-sample algorithm cannot distinguish between them with probability at least $2/3$. Let $\delta \in \mathbb{R}^{\geq 0}$ be some parameter to be set later; we will think of δ as tiny. Instances from these two distributions are generated as follows:

- $\mathcal{D}_{\text{yes}}$: Sample $w \sim \mathcal{N}(0, I)$ and return $f(x) := \langle w, x \rangle$.
- $\mathcal{D}_{\text{no}}$: Sample $w \sim \mathcal{N}(0, I)$ and for every $x \in \mathbb{R}^n$ sample $\varepsilon_x \sim \mathcal{N}(0, \delta)$. Return $f(x) := \langle w, x \rangle + \varepsilon_x$.

The functions in the support of $\mathcal{D}_{\text{yes}}$ are linear by definition. It remains to show that the instances in the support of $\mathcal{D}_{\text{no}}$ are far from linear.

▶ **Lemma 19.** *With probability* 1 *any* $f \sim \mathcal{D}_{\text{no}}$ *is* 1/3*-far from linear.*

The proof of this lemma will hinge on the following claim.

▷ Claim 20. Let $f \sim \mathcal{D}_{\text{no}}$, for $x, y, z \sim \mathcal{N}(0, 1)$, $\Pr[f(\frac{x-y}{2}) \neq f(\frac{x-z}{2}) + f(\frac{z-y}{2})] = 1$.

Proof. Observe that $\Pr\left[f\left(\frac{x-y}{2}\right) = f\left(\frac{x-z}{2}\right) + f\left(\frac{z-y}{2}\right)\right] = \Pr\left[\varepsilon_{(x-y)/2} = \varepsilon_{(x-z)/2} + \varepsilon_{(z-y)/2}\right]$, where the probability is over $\varepsilon_{(x-y)/2}, \varepsilon_{(x-z)/2}, \varepsilon_{(z-y)/2} \sim \mathcal{N}(0, \delta)$. Define the random variable $z := \varepsilon_{(x-y)/2} - \varepsilon_{(x-z)/2} - \varepsilon_{(z-y)/2}$, and note that z is distributed according to $\mathcal{N}(0, 3\delta)$. Then

$$\Pr_{z \sim \mathcal{N}(0,3)}\left[\varepsilon_{(x-y)/2} = \varepsilon_{(x-z)/2} + \varepsilon_{(z-y)/2}\right] = \Pr_{z \sim \mathcal{N}(0,3)}[z = 0].$$

By standard arguments, we have $\Pr_{z \sim \mathcal{N}(0,3\delta)}[z = 0] = 0$. ◁

Proof of Lemma 19. Let f^* be the closest linear function to f. For a point $x \in \mathbb{R}^n$, say that $f(x)$ is *bad* if $f(x) \neq f(x^*)$. Construct the following matrix: the rows are labelled by every triple $(\frac{x-y}{2}, \frac{x-z}{2}, \frac{z-y}{2})$ and there are three columns. The entries at row $(\frac{x-y}{2}, \frac{x-z}{2}, \frac{z-y}{2})$ are $f(\frac{x-y}{2})$, $f(\frac{x-z}{2})$, and $f(\frac{z-y}{2})$. Note that because $x, y, z \sim \mathcal{N}(0, 1)$, the points $\frac{x-y}{2}, \frac{x-z}{2}, \frac{z-y}{2}$ are distributed according to $\mathcal{N}(0, 1)$.

Henceforth, we will measure mass in terms of probability mass over $\mathcal{N}(0, 1)$. By Claim 20, the probability that each row contains a bad entry is 1. Therefore, there must be some column for which the probability mass of the bad entries is at least 1/3. This implies that a mass of at least 1/3 of f must be changed to obtain f^*. Because f^* is the closest linear function to f, this implies that f is 1/3-far from linear. ◀

Having defined our distributions over linear and far-from-linear functions, it remains to argue that no algorithm receiving n samples can distinguish between them with high probability.

Proof of Theorem 3. Let $\mathcal{D}$ be the distribution that with probability 1/2 draws $f \sim \mathcal{D}_{\text{yes}}$ and otherwise draws $f \sim \mathcal{D}_{\text{no}}$. Let A be any deterministic algorithm which receives n samples $x_1, \ldots, x_n \sim \mathcal{N}(0, I)$. By Yao's minimax principle, it suffices to show that A cannot correctly distinguish which distribution of the distributions $\mathcal{D}_{\text{yes}}$ or $\mathcal{D}_{\text{no}}$ a given sample $f \sim \mathcal{D}$ comes from with probability at least 2/3. That is, we would like to show that

$$\left| \Pr_{\substack{f \sim \mathcal{D}_{\text{yes}} \\ x_1, \ldots, x_n \sim \mathcal{N}(0,I)}} [A(f(x_1), \ldots, f(x_n)) = 1] - \Pr_{\substack{f \sim \mathcal{D}_{\text{no}} \\ x_1, \ldots, x_n \sim \mathcal{N}(0,I)}} [A(f(x_1), \ldots, f(x_n)) = 1] \right| \quad (4)$$

is $o(1)$. Suppose for contradiction that an algorithm A exists that with probability at least 2/3 distinguishes these distributions.

Observe that the (4) can be bounded from above by the total variation distance between the distributions $(f^y(x_1), \ldots, f^y(x_n))$ for $f^y \sim \mathcal{D}_{\text{yes}}$, and $(f^n(x_1), \ldots, f^n(x_n))$ for $f^n \sim \mathcal{D}_{\text{no}}$, for $x_1, \ldots, x_n \sim \mathcal{N}(0, I)$, as applying the algorithm A can only make the total variation distance smaller. By the definition of $\mathcal{D}_{\text{yes}}$ and $\mathcal{D}_{\text{no}}$, this means bounding the total variation distance between $(w_y^\top x_1, \ldots, w_y^\top x_n)$ and $(w_n^\top x_1 + \varepsilon_{x_1}, \ldots, w_n^\top x_n + \varepsilon_{x_n})$, where $w_y \sim \mathcal{D}_{\text{yes}}$ and $w_n \sim \mathcal{D}_{\text{no}}$.

Now, let $X \in \mathbb{R}^n$ be the matrix whose rows are $x_1, \ldots, x_n$. Because $w_y, w_n \sim \mathcal{N}(0, I)$ and $\varepsilon_{x_i} \sim \mathcal{N}(0, \delta)$, it follows that

$$(w^\top x_1, \ldots, w^\top x_n) \sim \mathcal{N}(0, XX^\top),$$
$$(w_n^\top x_1, \ldots, w_n^\top x_n) + (\varepsilon_{x_1}, \ldots, \varepsilon_{x_n}) \sim \mathcal{N}(0, XX^\top + \delta I).$$

Therefore,

$$(4) \leq \mathrm{d_{TV}}(\mathcal{N}(0, XX^\top), \mathcal{N}(0, XX^\top + \delta I)).$$

To bound this distance we will appeal to Pinkser's inequality and Lemma 5. Thus, it will be useful to first record some facts about the covariance matrices of these distribution. First, we show that the rows of the matrix X are linearly independent with high probability.

▶ Fact 21. $\Pr_{x_1,\ldots,x_n \sim \mathcal{N}(0,1)}[\mathrm{span}(x_1, \ldots, x_n) = \mathbb{R}^n] = 1.$

It follows that the covariance matrices of these two distributions are positive definite with high probability.

▷ Claim 22. With probability 1 the matrices $XX^\top$ and $XX^\top + \delta I$ are positive definite.

Proof. That $XX^\top \succ 0$ is immediate from Fact 21, which implies that rows of X are linearly independent with probability 1. Let $\lambda_1, \ldots, \lambda_n$ be the eigenvalues of $XX^\top$. To prove that $XX^\top + \delta I \succ 0$ note that adding δI simply adds δ to each of the eigenvalues. Thus, the eigenvalues of $XX^\top + \delta I$ are all positive. ◁

With these facts in hand we turn to bounding the total variation distance between $\mathcal{N}(0, XX^\top)$ and $\mathcal{N}(0, XX^\top + \delta I)$. Denote by $\Sigma_{\mathrm{yes}} := XX^\top$ and $\Sigma_{\mathrm{no}} := XX^\top + \delta I$. By Pinkser's inequality (Theorem 4) and Lemma 5,

$$\mathrm{d_{TV}}\left(\mathcal{N}(0, XX^\top), \mathcal{N}(0, XX^\top + \delta I)\right) \leq \sqrt{\frac{1}{4}\left(\log\left(\frac{\det \Sigma_{\mathrm{yes}}}{\det \Sigma_{\mathrm{no}}}\right) + \mathrm{tr}\left(\Sigma_{\mathrm{yes}}^{-1}\Sigma_{\mathrm{no}}\right) - n\right)}.$$

We will bound each of these terms separately.

Bounding the Determinant

For simplicity of notation, we will bound the inverse of $\det(\Sigma_{\mathrm{yes}})/\det(\Sigma_{\mathrm{NO}})$ below. We have

$$\begin{aligned}\frac{\det \Sigma_{\mathrm{no}}}{\det \Sigma_{\mathrm{yes}}} &= \frac{\det(XX^\top + \delta I)}{\det(XX^\top)} \\ &= \det\left(XX^\top\left(XX^\top\right)^{-1} + \delta\left(XX^\top\right)^{-1}\right) \\ &= \det\left(I + \delta\left(XX^\top\right)^{-1}\right).\end{aligned}$$

▷ Claim 23. If A is a diagonalizable matrix with eigenvalues $\lambda_1, \ldots, \lambda_n$ then $\det(A + I) = \prod_{i=1}^n (\lambda_i + 1)$.

Applying this claim, we have $\det(I + \delta(XX^\top)^{-1}) = (\delta\lambda_1^{-1} + 1)\ldots(\delta\lambda_n^{-1} + 1)$, where $\lambda_1, \ldots, \lambda_n$ are the eigenvalues of $XX^\top$. By Claim 22 the matrix $XX^\top$ is positive definite and so $\lambda_i > 0$ for all i. Therefore, $(\delta\lambda_i^{-1} + 1) > 1$ for all i, and we can conclude that $\det \Sigma_{\mathrm{n}}/\det \Sigma_{\mathrm{yes}} > 1$. Thus we can upper bound $\det \Sigma_{\mathrm{yes}}/\det \Sigma_{\mathrm{no}}$ by 1.

Bounding the Trace

Next, we bound

$$\begin{aligned}\operatorname{tr}\left(\Sigma_{\text{yes}}^{-1}\Sigma_{\text{no}}\right) &= \operatorname{tr}\left(\left(XX^\top\right)^{-1}(XX^\top+\delta I)\right)\\ &= \operatorname{tr}\left(I+\delta\left(X^\top\right)^{-1}X^{-1}\right)\\ &\leq \operatorname{tr}(I)+\delta\operatorname{tr}\left(\left(X^\top\right)^{-1}X^{-1}\right)\\ &= n+\delta\sum_{i,j}\left(X_{i,j}^{-1}\right)^2\\ &\leq n+\delta n^2\cdot\lambda_{\max}(X^{-1})^2,\end{aligned}$$

where $\lambda_{\max}$ is the largest eigenvalue of X^{-1}. Noting that the eigenvalues of X^{-1} are the inverse of the eigenvalues of X, we have $\operatorname{tr}\left(\Sigma_{\text{yes}}^{-1}\Sigma_{\text{no}}\right) \leq n+\delta n^2/\lambda_{\min}(X)^2$. Setting $\delta := C\lambda_{\min}(X)^2/n^2$ for some tiny $C>0$ to be set later, we can conclude that $\operatorname{tr}\left(\Sigma_{\text{yes}}^{-1}\Sigma_{\text{no}}\right) \leq n+C$.

Completing the proof

Putting our previous bounds together we conclude that

$$\mathrm{d}_{\mathrm{TV}}\left(\mathcal{N}(0,XX^\top),\mathcal{N}(0,XX^\top+\delta I)\right) \leq \sqrt{\frac{1}{4}\left(\log(1)+n+C-n\right)} = \frac{1}{2}C^{1/2}.$$

By our previous argument we have

$$(4) \leq \mathrm{d}_{\mathrm{TV}}\left(\mathcal{N}(0,XX^\top),\mathcal{N}(0,XX^\top+\delta I)\right) \leq \frac{1}{2}C^{1/2}.$$

Setting $C<(2/3)^2$ contradicts our assumption of the existence of an algorithm A which distinguishes a sample drawn from $\mathcal{D}_{\text{yes}}$ from one drawn from $\mathcal{D}_{\text{no}}$ with probability at least $2/3$, completing the proof. ◀

Finally, observe that the same proof goes through for testing additivity as well. Indeed, $\mathcal{D}_{\text{yes}}$ is supported on additive functions, while $\mathcal{D}_{\text{no}}$ is supported on functions which are far from additive with probability 1.

▶ **Corollary 24.** *Any sampler for additivity of functions $f\colon \mathbb{R}^n \to \mathbb{R}$ requires $\Omega(n)$ samples when $\mathcal{D}=\mathcal{N}(0,I)$.*

References

1. Sigal Ar, Manuel Blum, Bruno Codenotti, and Peter Gemmell. Checking approximate computations over the reals. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing (STOC)*, pages 786–795, 1993. doi:10.1145/167088.167288.
2. Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof Verification and Hardness of Approximation Problems. In *Proceedings of the 33rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 14–23, 1992. doi:10.1109/SFCS.1992.267823.
3. Maria-Florina Balcan, Eric Blais, Avrim Blum, and Liu Yang. Active Property Testing. In *Proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 21–30, 2012. doi:10.1109/FOCS.2012.64.
4. Robert Gardner Bartle and Donald R Sherbert. *Introduction to real analysis*. Hoboken, NJ: Wiley, 2011.

5 Mihir Bellare, Oded Goldreich, and Madhu Sudan. Free Bits, PCPs and Non-Approximability - Towards Tight Results. In *Proceedings of the 36th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 422–431, 1995. doi:10.1109/SFCS.1995.492573.

6 Aleksandrs Belovs. Quantum Algorithm for Distribution-Free Junta Testing. In *Proceedings of the 14th International Computer Science Symposium in Russia (CSR)*, pages 50–59, 2019. doi:10.1007/978-3-030-19955-5_5.

7 Michael Ben Or, Don Coppersmith, Mike Luby, and Ronitt Rubinfeld. Non-Abelian homomorphism testing, and distributions close to their self-convolutions. *Random Structures & Algorithms*, 32(1):49–70, January 2008.

8 Arnab Bhattacharyya, Swastik Kopparty, Grant Schoenebeck, Madhu Sudan, and David Zuckerman. Optimal Testing of Reed-Muller Codes. In *Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 488–497, 2010.

9 Eric Blais. Testing juntas nearly optimally. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing (STOC)*, pages 151–158, 2009.

10 Eric Blais, Amit Weinstein, and Yuichi Yoshida. Partially Symmetric Functions Are Efficiently Isomorphism Testable. *SIAM Journal on Computing*, 44(2):411–432, 2015.

11 Manuel Blum, Michael Luby, and Ronitt Rubinfeld. Self-Testing/Correcting with Applications to Numerical Problems. *Journal of Computer and System Sciences*, 47(3):549–595, 1993. doi:10.1016/0022-0000(93)90044-W.

12 Nader H. Bshouty. Almost Optimal Distribution-Free Junta Testing. In *Proceedings of the 34th Computational Complexity Conference (CCC)*, pages 2:1–2:13, 2019. doi:10.4230/LIPIcs.CCC.2019.2.

13 Xi Chen, Adam Freilich, Rocco A. Servedio, and Timothy Sun. Sample-Based High-Dimensional Convexity Testing. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM)*, pages 37:1–37:20, 2017. doi:10.4230/LIPIcs.APPROX-RANDOM.2017.37.

14 Xi Chen and Jinyu Xie. Tight Bounds for the Distribution-Free Testing of Monotone Conjunctions. In *Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 54–71, 2016. doi:10.1137/1.9781611974331.ch5.

15 Anindya De, Elchanan Mossel, and Joe Neeman. Is your function low-dimensional? *arXiv e-prints*, page arXiv:1806.10057, 2018. arXiv:1806.10057.

16 Elya Dolev and Dana Ron. Distribution-Free Testing for Monomials with a Sublinear Number of Queries. *Theory of Computing*, 7(1):155–176, 2011. doi:10.4086/toc.2011.v007a011.

17 Funda Ergün, Ravi Kumar, and Ronitt Rubinfeld. Checking Approximate Computations of Polynomials and Functional Equations. *SIAM Journal on Computing*, 31(2):550–576, 2001. doi:10.1137/S0097539798337613.

18 Eldar Fischer, Guy Kindler, Dana Ron, Shmuel Safra, and Alex Samorodnitsky. Testing juntas. *Journal of Computer and System Sciences*, 68(4):753–787, 2004.

19 Peter Gemmell, Richard J. Lipton, Ronitt Rubinfeld, Madhu Sudan, and Avi Wigderson. Self-Testing/Correcting for Polynomials and for Approximate Functions. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing (STOC)*, pages 32–42, 1991. doi:10.1145/103418.103429.

20 Dana Glasner and Rocco A. Servedio. Distribution-Free Testing Lower Bounds for Basic Boolean Functions. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM)*, pages 494–508, 2007. doi:10.1007/978-3-540-74208-1_36.

21 Oded Goldreich. *Introduction to Property Testing*. Cambridge University Press, 2017. doi:10.1017/9781108135252.

22 Oded Goldreich, Shafi Goldwasser, and Dana Ron. Property Testing and its Connection to Learning and Approximation. *Journal of the ACM*, 45(4):653–750, 1998. doi:10.1145/285055.285060.

23 Shirley Halevy and Eyal Kushilevitz. Distribution-Free Property-Testing. *SIAM Journal on Computing*, 37(4):1107–1138, 2007. doi:10.1137/050645804.

24 Nathaniel Harms. Testing Halfspaces over Rotation-Invariant Distributions. In *Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 694–713, 2019. doi:10.1137/1.9781611975482.44.

25 Johan Håstad. Clique is Hard to Approximate Within $n^{1-\epsilon}$. In *Proceedings of the 37th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 627–636, 1996. doi:10.1109/SFCS.1996.548522.

26 Tali Kaufman and Dana Ron. Testing Polynomials over General Fields. *SIAM Journal on Computing*, 36(3):779–802, 2006. doi:10.1137/S0097539704445615.

27 Marcos A. Kiwi, Frédéric Magniez, and Miklos Santha. Approximate Testing with Relative Error. In *Proceedings of the 31st Annual ACM Symposium on Theory of Computing (STOC)*, pages 51–60, 1999. doi:10.1145/301250.301269.

28 Marcos A. Kiwi, Frédéric Magniez, and Miklos Santha. Exact and Approximate Testing/-Correcting of Algebraic Functions: A Survey. In *Theoretical Aspects of Computer Science, Advanced Lectures (First Summer School on Theoretical Aspects of Computer Science, Tehran, Iran, July 2000)*, pages 30–83, 2000. doi:10.1007/3-540-45878-6_2.

29 Pravesh Kothari, Amir Nayyeri, Ryan O'Donnell, and Chenggang Wu. Testing Surface Area. In *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1204–1214, 2014. doi:10.1137/1.9781611973402.89.

30 Zhengyang Liu, Xi Chen, Rocco A. Servedio, Ying Sheng, and Jinyu Xie. Distribution-free Junta Testing. *ACM Transactions on Algorithms*, 15(1):1:1–1:23, 2019. doi:10.1145/3264434.

31 Kevin Matulef, Ryan O'Donnell, Ronitt Rubinfeld, and Rocco A Servedio. Testing Halfspaces. *SIAM Journal on Computing*, 39(5):2004–2047, 2010.

32 Kevin Matulef, Ryan O'Donnell, Ronitt Rubinfeld, and Rocco A. Servedio. Testing (Subclasses of) Halfspaces. In *Property Testing - Current Research and Surveys*, pages 334–340. Springer, 2010. doi:10.1007/978-3-642-16367-8_27.

33 Kevin Matulef, Ryan O'Donnell, Ronitt Rubinfeld, and Rocco A Servedio. Testing ± 1-weight halfspace. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM)*, pages 646–657. Springer, 2009.

34 Joe Neeman. Testing surface area with arbitrary accuracy. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing (STOC)*, pages 393–397, 2014. doi:10.1145/2591796.2591807.

35 Kenta Oono and Yuichi Yoshida. Testing properties of functions on finite groups. *Random Structures & Algorithms*, 49(3):579–598, 2016.

36 Sofya Raskhodnikova and Ronitt Rubinfeld. Linearity and Group Homomorphism Testing/Testing Hadamard Codes. *Encyclopedia of Algorithms*, pages 1–6, 2014.

Random Sketching, Clustering, and Short-Term Memory in Spiking Neural Networks

Yael Hitron
Weizmann Institute of Science, Rehovot, Israel
yael.hitron@weizmann.ac.il

Nancy Lynch
Massachusetts Institute of Technology, Cambridge, MA, USA
lynch@csail.mit.edu

Cameron Musco
University of Massachusetts, Amherst, MA, USA
cmusco@cs.umass.edu

Merav Parter
Weizmann Institute of Science, Rehovot, Israel
merav.parter@weizmann.ac.il

Abstract

We study input compression in a biologically inspired model of neural computation. We demonstrate that a network consisting of a random projection step (implemented via random synaptic connectivity) followed by a sparsification step (implemented via winner-take-all competition) can reduce well-separated high-dimensional input vectors to well-separated low-dimensional vectors. By augmenting our network with a third module, we can efficiently map each input (along with any small perturbations of the input) to a unique *representative neuron*, solving a neural clustering problem.

Both the size of our network and its processing time, i.e., the time it takes the network to compute the compressed output given a presented input, are independent of the (potentially large) dimension of the input patterns and depend only on the number of distinct inputs that the network must encode and the pairwise relative Hamming distance between these inputs. The first two steps of our construction mirror known biological networks, for example, in the fruit fly olfactory system [9, 29, 17]. Our analysis helps provide a theoretical understanding of these networks and lay a foundation for how random compression and input memorization may be implemented in biological neural networks.

Technically, a contribution in our network design is the implementation of a *short-term* memory. Our network can be given a desired *memory time* t_m as an input parameter and satisfies the following with high probability: any pattern presented several times within a time window of t_m rounds will be mapped to a single representative output neuron. However, a pattern not presented for $c \cdot t_m$ rounds for some constant $c > 1$ will be "forgotten", and its representative output neuron will be released, to accommodate newly introduced patterns.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

Keywords and phrases biological distributed computing, spiking neural networks, compressed sensing, clustering, random projection, dimensionality reduction, winner-take-all

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.23

1 Introduction

In this work we study brain-like networks that receive potentially complex and high-dimensional inputs (e.g., from sensory neurons representing odors, faces, or sounds) and encode these inputs in a very compressed way. Specifically, we consider networks with n input neurons and k output neurons, where n may be very large. When presented with up to k sufficiently different but otherwise arbitrary input patterns, the goal of the network is

to represent the inputs in such a way that they can be recognized when presented again: each input should be uniquely mapped to a single *representative output neuron* that fires if that input pattern is reintroduced. Further, any small perturbations of a presented input should be recognized by the same representative neuron. We call the above problem the *neural clustering problem*.

Clustering, input memorization, and compression are fundamental problems in biological neural networks. Our work is also inspired by the important *novelty detection* problem [25, 41]. Novelty detection requires detecting inputs that differ significantly from previously seen inputs. It is easy to see that this problem can be solved with a neural clustering network, in which all sufficiently far inputs are mapped to different representative neurons and all sufficiently close inputs are mapped to the same neuron. A novel input is detected whenever a new representative neuron is assigned. The novelty detection problem has been considered recently in the fruit fly olfactory system [16], where it is believed to be solved using a random projection based method. The high level structure of this method closely resembles the initial stages of our clustering algorithm, and we see a major contribution of our work as providing a theoretical understanding of how random projection can be implemented in biologically inspired neural networks. For further discussion about the connection to fruit fly novelty detection see Section 1.2.

1.1 Our Results

We study the neural clustering problem in a biologically inspired model of *stochastic spiking neural networks* (stochastic SNNs), which was previously defined in [33, 34, 35]. In these networks, computation proceeds in discrete rounds with each neuron either firing (spiking) in a round or remaining silent. Each neuron spikes randomly, with probability determined by its membrane potential. This potential is induced by spikes from neighboring neurons, which can have either an excitatory or inhibitory effect (increasing or decreasing the potential). In general, the input to an SNN is a stream of binary vectors, corresponding to spikes of the input neurons. In our setting we will consider a single binary vector as the input pattern and assume that each input vector is presented for a certain number of consecutive rounds before changing. This allows the network time to stabilize to the correct output associated with the given input.

We demonstrate that clustering can be solved efficiently in these networks, where the cost is measured by (i) the number of *auxiliary neurons*, besides the input and output neurons, that are required to solve the clustering task and (ii) the number of rounds required to converge to the correct output for a given input, which corresponds to the number of rounds for which the input must be presented for before moving to the next input.

In the clustering problem, we consider a (potentially large) set of n-length patterns that are clustered around k base patterns. It is then required to map all patterns in the same cluster to a unique output in $[k]$.

Clustering with Output Reassignment. We also want our network to be reusable, with a *memory duration* t_m that is given as an input parameter. Instead of considering a single infinite input stream with at most k distinct patterns (or clusters of patterns), our memory module allows one having many distinct patterns, as long as their presentation times are sufficiently spaced out. That is, in any window of $\Theta(t_m)$ rounds, the network is presented at most k distinct patterns. To handle distinct patterns in each $\Theta(t_m)$-round window, the network must *forget* patterns that have not been introduced for a while and release their allocated outputs so that they can be assigned to new inputs. Specifically, for some fixed

constant c, our network remembers a pattern (its cluster) for at least t_m rounds and at most $c \cdot t_m$ round. The output of any pattern not introduced for $c \cdot t_m$ rounds is released with high probability and can be reassigned to represent another input.

The Neural Clustering Problem. We now formally define the neural clustering problem, which is parametrized by several parameters: the input dimension n, the number of distinct input patterns k, the memory duration t_m, a bound on the relative distance of input patterns Δ, and the allowed failure probability δ. We require that every pattern introduced as input, remains the input pattern for at least $t_p = \text{poly}(k, 1/\Delta, \log(1/\delta))$ (i.e., independent of n) consecutive rounds. The t_p parameter is the processing time or mapping time, i.e., the time it takes for the network to converge to the output neuron. Throughout, we will assume that all patterns have p non-zero entries. We conjecture that this assumption can be easily removed however keep it to simplify our arguments.

Define the *relative Hamming distance* between two inputs $\overline{X}_i, \overline{X}_j \in \{0,1\}^n$ to be:

$$\mathcal{RD}(\overline{X}_i, \overline{X}_j) = \frac{||\overline{X}_i - \overline{X}_j||_1}{\max\{||\overline{X}_i||_1, ||\overline{X}_j||_1\}} \ .$$

In the basic clustering problem, the network is introduced to a possibly large number of distinct patterns that are *clustered* around k-centers. That is, in every window of t_m rounds, the patterns introduced are clustered around a base-set of k patterns $\overline{X}_1, \ldots, \overline{X}_k \in \{0,1\}^n$ such that the relative difference between each pair in the base-set is at least Δ, and any other pattern introduced is sufficiently close to one of the patterns in the base-set (with relative distance $\leq \Delta/\alpha$ for some $\alpha = \tilde{O}(1)$). In the clustering problem the network maps *similar* patterns to the same unique output q_i for $i \in [k]$ (i.e., the cluster name) and *non-similar* patterns to distinct names. Formally:

▶ **Definition 1** (Clustering Input Condition). *An infinite input sequence $\overline{Z}_1, \overline{Z}_2, \ldots$ is a well-behaved clustering input sequence with input size n, output size k, memory duration t_m, relative distance parameter Δ, closeness parameter α, and input persistence time t_p if:*

- *For any set of t_m rounds $T = \{t, t+1, ..., t+(t_m-1)\}$ there exist $\overline{X}_1, \overline{X}_2, ..., \overline{X}_k \in \{0,1\}^n$ such that $\mathcal{RD}(\overline{X}_i, \overline{X}_j) \geq \Delta$ for all $i \neq j$ and for all $i \in T$, $\mathcal{RD}(\overline{Z}_i, \overline{X}_j) \leq \Delta/\alpha$ for some $j \in [k]$.*
- *If $\overline{Z}_i \neq \overline{Z}_{i-1}$, then $\overline{Z}_i = \overline{Z}_{i+1} = ... = \overline{Z}_{i+t_p}$.*

▶ **Definition 2** (Clustering Network). *A network $\mathcal{N}$ solves the clustering problem for input size n, output size k, memory duration t_m, relative distance parameter Δ, closeness parameter α, input duration t_p, and failure probability δ if, on a well-behaved input sequence for the same parameters (Definition 1), on any fixed window of t_m rounds, with probability $\geq 1 - \delta$:*

- *Each input pattern $\overline{Z}$ is* mapped *to some output q_j for $j \in [k]$. That is, whenever the input changes to $\overline{Z}$ round i (so $\overline{Z}_{i-1} \neq \overline{Z}$ but $\overline{Z}_i = \overline{Z}$), there is a unique output neuron q_j that fires at round $i + t_p$ and continues to fire as long as the input remains fixed to $\overline{Z}$.*
- *Any pair of far patterns $\overline{Z}, \overline{Z}'$ with $\mathcal{RD}(\overline{Z}, \overline{Z}') \geq \Delta$ introduced within the t_m time window are mapped to different outputs.*
- *Any pair of closed-patterns $\overline{Z}, \overline{Z}'$ with $\mathcal{RD}(\overline{Z}, \overline{Z}') \leq \Delta/\alpha$ introduced within the same t_m time window will be mapped to the same output neuron.*

Our goal is to design a clustering network that uses small number of auxiliary neurons and requires small input persistence time t_p. We show the following theorem.

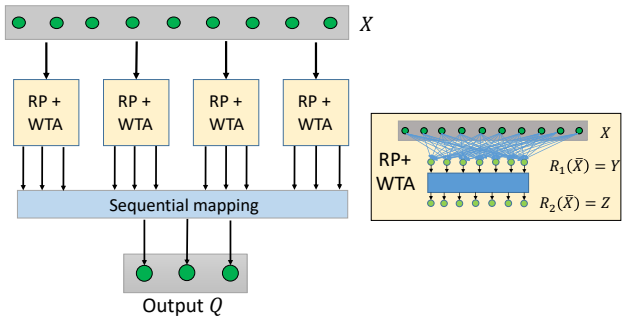


Figure 1 High level illustration of the clustering network. Right: The input pattern $\bar{X} \in \{0,1\}^n$ is mapped to an intermediate sparser vector in two steps: random projection and WTA sparsification. Left: In the clustering network, the input $\bar{X}$ is mapped by applying $O(\log(k/\delta))$ parallel repetitions of the random projection + WTA mapping. As a result, $\bar{X}$ is mapped to a vector $\bar{Z}$ with $O(\frac{\log(k/\delta)}{\Delta})$ neurons. This vector is mapped to the output unit vector in $\{0,1\}^k$ via a sequential mapping module.

▶ **Theorem 3.** *For any parameters n, k, t_m, δ and Δ, there is a network $\mathcal{N}$ with $O\left(\frac{\log(1/\Delta)^3 \log(t_m/\delta) \log(1/\delta)}{\Delta^{3/2}}\right)$ auxiliary neurons that solves the clustering problem with these parameters, input persistence time $t_p = O\left(\frac{\log(1/\Delta)^2 \log(t_m/\delta)}{\Delta}\right)$ and closeness parameter $\alpha = O(\log(1/\Delta)^4)$.*

Note that the number of auxiliary neurons and the convergence time of Theorem 3 are *independent of the input dimension n*, which may potentially be very large. The spiking neural network construction that achieves Theorem 3 involves in three steps. The first two steps reduce the input from n neurons to $m \ll n$ neurons, while approximately preserving the relative distances between inputs. These steps use a biologically inspired construction that mirrors circuits seen, for example, in the fruit fly olfactory system [9, 29, 17]. In particular the first step maps the input to a set of intermediate neurons via random projection, and the second step sparsifies the outputs of these intermediate neurons to yield a sparse code representing the input. The final *sequential mapping* step then solves the clustering problem given these m intermediate neurons as inputs, avoiding the high cost of directly solving the problem on the n-dimensional input. See Figure 1 for an illustration.

1.2 Comparison to Previous Work

1.2.1 Broader Agenda: Algorithmic Theory for Brain Networks

Understanding how the brain works, as a computational device, is a central challenge of modern neuroscience and artificial intelligence. Different research communities tackle this problem in different ways, ranging from studies that examine neural network structure as a clue to computational function [43, 3], to functional imaging that studies neural activation patterns [40, 31], to theoretical work on simplified models of neural computation [23, 36], to the engineering of complex neural-inspired machine learning architectures [21, 27]. This paper joins a recent line of work [44, 37, 45, 38, 33, 30, 46, 34, 28, 32, 39, 10, 22] that approach this problem using techniques from *distributed computing theory and other branches of theoretical computer science.* The ultimate goal of this research direction is to develop an *algorithmic theory for brain networks*, based on stochastic graph-based neural network models. To understand neural behavior from a theory of computing point of view, we design networks

to solve abstract problems that are inspired by tasks that seem to be solved in actual brains. We believe that the rigorous analysis of such networks in terms of static costs (e.g., the number of neurons), and dynamic costs (e.g., the time to converge to a solution) will lead to a better understanding of how these tasks may be performed in biological neural networks.

1.2.2 Connections to Sparse Recovery

Our work is closely related to sparse recovery (compressed sensing), where the goal is to map high-dimensional but sparse vectors (with dimension n and $s \ll n$ nonzero entries) into a much lower dimensional space, such that the vectors can be uniquely identified and efficiently recovered [19]. We can see that this goal is essentially identical to that of our first two network layers, before the sequential mapping step. Two different s-sparse binary vectors have relative hamming distance $\geq 1/s$. Additionally there are $k = O(n^s)$ s-sparse binary vectors in n dimensions. Thus, as a Corollary of Lemma 12, our first two layers can uniquely compress all such vectors with high probability into dimension $m = \tilde{O}\left(\frac{\log k}{\Delta^{1/2}}\right) = O(s^{3/2} \log n)$.

It is known that optimal sparse recovery reducing the dimension to $O(s \log n)$ can be achieved using random projections [13]. However, unlike in our setting, these random projections have real valued outputs, which cannot be directly represented by binary spiking neurons. The case when output of the random projection is thresholded to be a binary value has been studied extensively, under the name "one-bit compressed sensing" [6]. In this setting, it is known that dimension $\tilde{\Theta}\left(s^2 \log n\right)$ can be achieved and is required for general sparse recovery [1]. If the input vectors are restricted to be binary (as in our case), dimension $O(s^{3/2} \log n)$ is possible [24, 1]. Our results match this bound up to logarithmic factors.

1.2.3 Connections to Fruit Fly Novelty Detection via Bloom Filters

Recently, Dasgupta et al. [16] demonstrated that the fruit fly olfactory circuit implements a variant of a classic Bloom filter [5] to assess the novelty of odors. A bloom filter is a data structure that maintains a set of items, allowing for membership queries, with the possibility of occasional false positives. The filter has m bits and r random hash functions mapping the input space to integers in $1, ..., m$. When an item is inserted, it is hashed using these r functions and the bits corresponding to the hashed values are set to 1. A membership query is answered by hashing the input in question and checking that all r bits corresponding to its hashed values are set to 1.

Such a filter can be used to implement novelty detection – a novel pattern is detected whenever an insertion operation sets a new bit to 1 or an membership query returns false. Dasgupta et al. [16] demonstrate that a such a scheme is used in the fly olfactory circuit. The hashing step consists of a random projection followed by winner-take-all sparsification, which maps each input into a r-sparse binary vector. The r entries of this vector represent the r hash function outputs. This step closely resembles the first two layers of our clustering network.

Our third layer operates differently than a bloom filter, associating each sparsified intermediate vector to with unique output via sequential mapping rather than simply marking the bits corresponding to its entries. However, it can implement the same functionality (and correspondingly can implement novelty detection). Specifically, to implement insertion and deletion operations we can make the following modifications to the sequential mapping sub-network:

- The input layer contains an extra neuron that is set to 1 if the operation is insertion, and 0 if the operation is a membership query.
- In the sequential mapping step the output layer fires only if this extra neuron fires. In this way, new outputs will only be mapped during insertion operations and not membership queries.
- For query operations, we add an output neuron that fires only if there exists an index $j \in [k]$ for which many memory modules m_{ji}, as well as an association neuron a_{ji} fire.
- Novelty detection can be implemented via an additional output neuron that responds when an insertion causes a new output to be mapped or when a query operation returns false.

2 Computational Model and Preliminaries

We start by defining our model of stochastic spiking neural networks.

Network Structure. A *Spiking Neural Network* (SNN) $\mathcal{N} = \langle X, Q, A, w, b\rangle$ consists of n input neurons $X = \{x_1, \ldots, x_n\}$, m output neurons $Q = \{q_1, \ldots, q_m\}$, and ℓ auxiliary neurons $A = \{a_1, ..., a_\ell\}$. The directed, weighted synaptic connections between X, Q, and A are described by the weight function $w : [X \cup Q \cup A] \times [X \cup Q \cup A] \to \mathbb{R}$. A weight $w(u, v) = 0$ indicates that a connection is not present between neurons u and v. Finally, for any neuron v, $\beta(v) \in \mathbb{R}_{\geq 0}$ is the activation bias – as we will see, roughly, v's membrane potential must reach $\beta(v)$ for a spike to occur with good probability.

The in-degree of every input neuron x_i is zero. That is, $w(u, x) = 0$ for all $u \in [X \cup Q \cup A]$ and $x \in X$. Additionally, each neuron is either *inhibitory* or *excitatory*: if v is inhibitory, then $w(v, u) \leq 0$ for every u, and if v is excitatory, then $w(v, u) \geq 0$ for every u.

Neuron Chains. In our implementation, we make use of *chains* of neurons to create a delay in a response. For a neuron u, and integer ℓ, let $C_\ell(u)$ be a directed path of length ℓ starting with u. All neurons on the chain are excitatory. We then say that a chain $C_\ell(u)$ is connected to v if each neuron $w \in C_\ell(u)$ has an outgoing edge to v.

The SNN Model. An SNN evolves in discrete, synchronous rounds as a Markov chain. The firing probability of every neuron at time t depends on the firing status of its neighbors at time $t - 1$, via a standard sigmoid function, with details given below. For each neuron u, and each time $t \geq 0$, let $u^t = 1$ if u fires (i.e., generates a spike) at time t. Let u^0 denote the initial firing state of the neuron. Our results will specify the initial input firing states $x_j^0 = 1$ and assume that $u^0 = 0$ for all $u \in [Q \cup A]$. The firing state of each input neuron x_j in each round is the input to the network, and our results will specify to which sequences of input firing patterns they apply.

For each non-input neuron u and every $t \geq 1$, let $\text{pot}(u, t)$ denote the membrane potential at round t and $p(u, t)$ denote the corresponding firing probability ($\Pr[u^t = 1]$). These values are calculated as:

$$\text{pot}(u,t) = \sum_{v \in X \cup Q \cup A} w_{v,u} \cdot v^{t-1} - \beta(u) \text{ and } p(u,t) = \frac{1}{1 + e^{-\text{pot}(u,t)/\lambda}} \tag{1}$$

where $\lambda > 0$ is a *temperature parameter*, which determines the steepness of the sigmoid. It is easy to see that λ does not affect the computational power of the network. A network can be made to work with any λ simply by scaling the synapse weights and biases appropriately.

For simplicity we assume throughout that $\lambda = \frac{1}{\Omega(\log(n \cdot k \cdot \Delta \cdot t_m \cdot 1/\delta))}$, where $n, k, \delta, \Delta, t_m$ are the parameters of the clustering problem, defined in Section 1.1. Thus by (1), if $\text{pot}(u,t) \geq 1$, then $u^t = 1$ w.h.p. and if $\text{pot}(u,t) \leq -1$, $u^t = 0$ w.h.p. , where w.h.p. denotes with probability at least $1 - (1/\delta \cdot n \cdot k \cdot \Delta \cdot t_m)^{-c}$ for some constant c.

The remainder of the paper is devoted to proving Theorem 3. Our analysis considers the three stages of the network in sequence: random projection, sparsification, and sequential mapping to the final outputs.

3 Layer 1: Random Projection

The goal of this step is to reduce the input size from n input neurons to $m \ll n$ neurons while ensuring that the relative distance between any two n-length input vectors is approximately preserved after the mapping. In this way, we can solve the clustering problem working with the much smaller m neuron representation instead of the original n neuron input. While there are many ways in which distance may be preserved, we consider one in particular, based on the membrane potentials induced on the intermediate neurons by the inputs:

▶ **Definition 4** (Distance Preserving Dimensionality Reduction). *Consider $\overline{X}_1, \ldots, \overline{X}_k \in \{0,1\}^n$ with $\mathcal{RD}(\overline{X}_i, \overline{X}_j) \geq \Delta$ for $i \neq j$. Consider a network $\mathcal{N}$ mapping n input neurons to m intermediate neurons, which are split into b buckets each containing m/b neurons. $\mathcal{N}$ is distance preserving for $\overline{X}_1, \ldots, \overline{X}_k$ if, for any two $\overline{X}_i, \overline{X}_j$, and any round t, in the large majority of buckets, the identity of the neuron that in round $t+1$* has the highest membrane potential below a fixed threshold τ *is different if $\overline{X}_i$ is presented at round t than if $\overline{X}_j$ were presented.*[1]

Our network satisfies Definition 4 with parameters $m = \tilde{O}\left(\frac{1}{\sqrt{\Delta}}\right)$ and $b = \tilde{O}(1)$. We implement the dimensionality reduction step via *random projection*. We note that random projection has been studied extensively as a dimensionality reduction tool in computer science, with applications in data analysis [4, 7, 12], fast linear algebraic computation [42, 11], and sparse recovery [8]. See [47] for a survey. In neuroscience, it is thought that random projection may play a key role in neural dimensionality reduction [20, 2]. Random projection for example, underlies sparse coding of inputs in the fruit fly olfactory circuit [9, 17]. Random connections have also been studied in theoretical models for memory formation, in which inputs are mapped to representative output neurons [45, 38, 28].

We start with describing the construction and then analyzing its properties. The main outputs of this section are Corollaries 9 and 11 which show that, with high probability, the identities of the neurons with maximum membrane potential below some threshold τ in each bucket of the intermediate layer share little overlap for far inputs (with relative distance $\geq \Delta$) and significant overlap for close inputs (with relative distance $\leq \Delta/\alpha$ for $\alpha = O(\log(1/\Delta)^4)$). That is, the network satisfies the distance preserving dimensionality reduction guarantee of Definition 4 for far inputs, along with an analogous guarantee for close inputs.

Our mapping can be understood as an example of *local sensitive hashing* [18, 15, 17]. In each bucket, the input is hashed to the identity of the maximum potential neuron below τ in that bucket. Near inputs have many hash collisions, and thus there is significant overlap in the identities of the mapped neurons. Far input have fewer collisions and thus less overlap.

[1] We formally define how the membrane potential is calculated in Section 2. "Large majority" will be a constant fraction of the buckets significantly larger than 1/2, which will be specified in our bounds.

Layer Description. The random projection layer consists of $m \cdot \ell$ intermediate auxiliary neurons for $m = \Theta\left(\frac{\log(1/\Delta)}{\sqrt{\Delta}}\right)$ and $\ell = \Theta(\log(t_m/\delta))$. The layer is subdivided into ℓ buckets $b_1, ..., b_\ell$ containing m neurons each. Each input neuron has an excitatory connection to each neuron in the intermediate layer with weight sampled as a Chi-squared random variable (with one degree of freedom). We denote this random weight matrix connecting the two layers by $\mathrm{A} \in \mathbb{R}^{m\cdot\ell\times n}$. For $b \in 1, ..., \ell$, we let $\mathrm{A}_b \in \mathbb{R}^{m\times n}$ denote the rows of A corresponding to the intermediate neurons in bucket b. In typical applications of random projection, the entries of A are most commonly either Gaussian or Rademacher random variable. Here we use Chi-squared random variables as they are non-negative, a requirement in our setting where the outgoing edge weights from each neuron (corresponding to the entries in A) must be either all positive or all negative.

Layer Analysis. When the input neurons X fire with input pattern $\overline{X}_i \in \{0,1\}^n$ at time t, by (1) the vector of membrane potentials of the intermediate neurons at time $t+1$ is given by $\mathrm{A}\,\overline{X}_i \in \mathbb{R}^{m\times\ell}$. Our analysis will focus on the properties of this vector of potentials, which can be viewed as a real valued compressed representation of the input $\overline{X}_i$. Later, we will show how these properties lead to desirable properties of the spiking patterns of the intermediate neurons.

For technical reasons, we will not focus on the actual largest entry of $A_b\overline{X}_i$, but on the *largest entry bounded by some fixed threshold* τ which can still be identified via a minor modification to a traditional WTA circuit. We begin with a preliminary lemma showing that a Chi-squared distribution (the distribution of each entry in $A_b\overline{X}_i$) is roughly uniform around its mean. We give a proof in Appendix A.

▶ **Lemma 5** (Chi-squared uniformity). *Let $\mathcal{D}_p$ be the Chi-squared distribution with p degrees of freedom. For any c with $1 \le c < p^{1/2}$ there are constants c_ℓ, c_u (depending on c) such that, for any interval $[r_1, r_2] \subseteq \left[p - cp^{1/2}, p + cp^{1/2}\right]$, we have: $\frac{c_\ell(r_2-r_1)}{p^{1/2}} \le \Pr_{x\sim\mathcal{D}_p}[x \in [r_1, r_2]] \le \frac{c_u(r_2-r_1)}{p^{1/2}}$. That is, $\mathcal{D}_p$ is roughly uniform on the range $\left[p - cp^{1/2}, p + cp^{1/2}\right]$.*

We also use the fact that the Chi-squared distribution decays far from its mean, which follows from standard sub-exponential concentration bounds.

▶ **Lemma 6** (Chi-squared decay). *Let $\mathcal{D}_p$ be the Chi-squared distribution with p degrees of freedom. For any $c \le 1$ there is a constant c_1 (depending on c) such that:*

$$\Pr_{x\sim\mathcal{D}_p}\left[x \notin [p - c_1p^{1/2}, p + c_1p^{1/2}]\right] \le c.$$

Using the near-uniform distribution property of Lemma 5, we can show that with good probability, for every compressed vector $\mathrm{A}_b\,\overline{X}_i \in \mathbb{R}^m$ the gap between the two largest entries (bounded by the threshold) is $\Omega(p^{1/2}/m)$ – since there are m entries roughly uniformly distributed in a range of size $O(p^{1/2})$. This gap will be necessary for the neuron with the largest membrane potential (and hence the highest firing probability) to be reliably identified in the second sparsification layer of our network. We remark that in non-neural applications of random projection such a gap would not be necessary: the largest entry in the bucket can be typically be identified exactly.

The complete proof is given in Appendix A.

▶ **Lemma 7** (Sufficient Gap). *Consider our construction with bucket size $m = c_1$. Let $\overline{X} \in \{0,1\}^n$ be any input vector with $\|\overline{X}\| = p$ for $p \geq 5$. Let $\tau = p + 2p^{1/2}$ and for any $b \in [\ell]$ define:*

$$i_{1,b}(\overline{X}) = \underset{i\in[m]:[A_b\overline{X}](i)\leq\tau}{\arg\max} [A_b\overline{X}](i) \quad \text{and} \quad i_{2,b}(\overline{X}) = \underset{i\in[m]\setminus i_{1,b}(\overline{X}):[A_b\overline{X}](i)\leq\tau}{\arg\max} [A_b\overline{X}](i),$$

where we set $i_{1,b}(\overline{X}), i_{2,b}(\overline{X}) = 0$ in the case that no index satisfies the constraint. For sufficiently large constants c_1, c_2, with probability $\geq 99/100$ over the random choice of A_b, $i_{1,b}(\overline{X}) \neq 0$, $[\mathrm{A}_b\,\overline{X}](i_{1,b}(\overline{X})) \geq p$, and either $[\mathrm{A}_b\,\overline{X}](i_{1,b}(\overline{X})) - [\mathrm{A}_b\,\overline{X}](i_{2,b}(\overline{X})) \geq \frac{p^{1/2}}{c_2 m}$ or $i_{2,b}(\overline{X}) = 0$.

Along with Lemma 7 we prove that, with good probability, the neuron with the maximum potential below τ in each bucket differs for any two far inputs.

▶ **Lemma 8** (Low Collision Probability – Far Inputs). *Let $\overline{X}_1, \overline{X}_2 \in \{0,1\}^n$ be two vectors with $\|\overline{X}_1\| = \|\overline{X}_2\| = p$ [2] and $\mathcal{RD}(\overline{X}_1, \overline{X}_2) \geq \Delta$. Assume that $p \geq c$ for some sufficiently large constant c. Consider our construction with bucket size $m = \frac{c_1 \log(1/\Delta)}{\sqrt{\Delta}}$. Then for sufficiently large constants c_1, c_2, for any $b \in [\ell]$, defining $i_{1,b}(\cdot)$ and $i_{2,b}(\cdot)$ as in Lemma 7, with probability ≥ 0.9165:*

- *$i_{1,b}(\overline{X}_1) \neq i_{1,b}(\overline{X}_2)$.*
- *For both $j = 1, 2$: $i_{1,b}(\overline{X}_j) \neq 0$, $[A_b\overline{X}_j](i_{1,b}(\overline{X}_j)) \geq p$, and $[A_b\overline{X}_j](i_{1,b}(\overline{X}_j)) - [A_b\overline{X}_j](i_{2,b}(\overline{X}_j)) \geq \frac{p^{1/2}}{c_2 \cdot m}$ or $i_{2,b}(\overline{X}_j) = 0$.*

See Appendix A for the complete proof of Lemma 8. Intuitively, if $\overline{X}_1$ and $\overline{X}_2$ each have Hamming weight p and relative distance Δ they differ on $\Omega(\Delta p)$ entries. If *just the shared entries* of these vectors fired as inputs, by Lemma 5 each intermediate neuron in the bucket of size m would have its potential distributed roughly uniformly in a range of width $O([(1-\Delta)p]^{1/2}) = O(p^{1/2})$. On average these potentials would be spaced out by $O(p^{1/2}/m)$. By setting $m = \tilde{O}(1/\sqrt{\Delta})$ we have average spacing $\tilde{O}(\Delta^{1/2}p^{1/2})$. This is a small enough spacing, so that when we consider the $\Omega(\Delta p)$ non-shared neurons in the inputs, their contribution to the potential will be large enough to significantly reorder the potentials of the intermediate neurons, so that the neuron with maximum potential is unlikely to be the same for the two different inputs.

From Lemma 8 we can show that our network satisfies the distance preserving dimensionality reduction guarantee of Definition 4, along with the additional condition that the gap between the membrane potentials of the neurons with the largest potentials under $\tau = p + 2p^{1/2}$ is sufficiently large, so that these neurons can be distinguished reliably in the second sparsification layer:

▶ **Corollary 9** (Overall Success – Far Inputs). *For $m = O\left(\frac{\log(1/\Delta)}{\sqrt{\Delta}}\right)$, and $\ell = O(\log(t_m/\delta))$, for any window of t_m rounds, with probability $\geq 1-\delta$, for all pairs of inputs $\overline{X}_1, \overline{X}_2$ presented during these rounds with $\mathcal{RD}(\overline{X}_1, \overline{X}_2) \geq \Delta$, on at least $91/100 \cdot \ell$ of the ℓ buckets, letting $\tau = p + 2p^{1/2}$ and defining $i_{1,b}(\cdot)$ and $i_{2,b}(\cdot)$ as in Lemma 7:*

- *$i_{1,b}(\overline{X}_1) \neq i_{1,b}(\overline{X}_2)$*
- *For both $j = 1, 2$: $i_{1,b}(\overline{X}_j) \neq 0$, $[A_b\overline{X}_j](i_{1,b}(\overline{X}_j)) \geq p$, and $[A_b\overline{X}_j](i_{1,b}(\overline{X}_j)) - [A_b\overline{X}_j](i_{2,b}(\overline{X}_j)) = \Omega\left(\frac{p^{1/2}}{m}\right)$ or $i_{2,b}(\overline{X}_j) = 0$.*

[2] When $\overline{X}$ is binary we often drop the subscript and just use $\|\overline{X}\|$ to denote the ℓ_1 norm which is equal to the number of nonzero entries, $|\operatorname{supp}(\overline{X})|$.

Proof. By Lemma 8 and a Chernoff bound, since $\ell = \Theta(\log(t_m/\delta)) = \Theta(\log(t_m^2/\delta))$, for any fixed pair of inputs with $\mathcal{RD}(\overline{X}_1, \overline{X}_2) \geq \Delta$, the conditions hold on at least $91/100 \cdot \ell$ buckets with probability $\geq 1 - \delta/t_m^2$. The corollary follows since at most t_m inputs can be presented in t_m rounds, and so we can union bound over at most t_m^2 pairs of far inputs. ◀

We can give a complementary statement to Lemma 8: if $\overline{X}_1$ and $\overline{X}_2$ are close to each other, it is relatively likely that the index of the largest value of $\mathrm{A}_b\,\overline{X}_1$ and $\mathrm{A}_b\,\overline{X}_2$ are the same. We defer the proof to Appendix A.

▶ **Lemma 10** (High Collision Probability – Close Inputs). *Let $\overline{X}_1, \overline{X}_2 \in \{0,1\}^n$ be two vectors with $\mathcal{RD}(\overline{X}_1, \overline{X}_2) \leq \Delta/\alpha$. Consider our construction with bucket size $m = \frac{c_1 \log(1/\Delta)}{\sqrt{\Delta}}$. Then for sufficiently large constants c_1, c_2 and $\alpha = O(\log(1/\Delta)^4)$, for any $b \in [\ell]$, defining $i_{1,b}(\cdot)$ and $i_{2,b}(\cdot)$ as in Lemma 7, with probability ≥ 0.97:*

- *$i_{1,b}(\overline{X}_1) = i_{1,b}(\overline{X}_2)$.*
- *For both $j = 1, 2$: $i_{1,b}(\overline{X}_j) \neq 0$, $[A_b\overline{X}_j](i_{1,b}(\overline{X}_j)) \geq p$, and $[A_b\overline{X}_j](i_{1,b}(\overline{X}_j)) - [A_b\overline{X}_j](i_{2,b}(\overline{X}_j)) \geq \frac{p^{1/2}}{c_2 \cdot m}$ or $i_{2,b}(\overline{X}_j) = 0$.*

Lemma 10 yields an analogous corollary to Corollary 9, which follows via a Chernoff bound and a union bound over at most t_m^2 pairs of close inputs that may be presented over t_m rounds.

▶ **Corollary 11** (Overall Success – Close Inputs). *For $m = O\left(\frac{\log(1/\Delta)}{\sqrt{\Delta}}\right)$, $\ell = O(\log(t_m/\delta))$, and $\alpha = O(\log(1/\Delta)^4)$, for any window of t_m rounds, with probability $\geq 1 - \delta$, for all pairs of inputs $\overline{X}_1, \overline{X}_2$ presented during these rounds with $\mathcal{RD}(\overline{X}_1, \overline{X}_2) \leq \Delta/\alpha$, on at least $96/100 \cdot \ell$ of the ℓ buckets, letting $\tau = p + 2p^{1/2}$ and defining $i_{1,b}(\cdot)$ and $i_{2,b}(\cdot)$ as in Lemma 7:*

- *$i_{1,b}(\overline{X}_1) = i_{1,b}(\overline{X}_2)$*
- *For both $j = 1, 2$: $i_{1,b}(\overline{X}_j) \neq 0$, $[A_b\overline{X}_j](i_{1,b}(\overline{X}_j)) \geq p$, and $[A_b\overline{X}_j](i_{1,b}(\overline{X}_j)) - [A_b\overline{X}_j](i_{2,b}(\overline{X}_j)) = \Omega\left(\frac{p^{1/2}}{m}\right)$ or $i_{2,b}(\overline{X}_j) = 0$.*

4 Layer 2: Sparsification via Winner Takes All

Corollaries 9 and 11 show that the random projection step preserves significant information about input distance, encoded in the membrane potentials of the intermediate neurons, which correspond to the entries of $\mathrm{A}\,\overline{X}$ when the network is given input $\overline{X}$. These membrane potentials cause the intermediate neurons to fire randomly, as Bernoulli processes with different rates. The goal of our second layer is to convert this random behavior to a uniquely identifying sparse code for each input. We achieve this through a winner-takes-all (WTA) based sparsification process, which is thought to play a major role in neural computation [26, 14, 37]. A separate winner-take-all instance is applied to each bucket, "selecting" the neuron with the highest membrane potential below τ by inducing its corresponding neuron in the sparsification layer to fire with high probability while all other neurons in the bucket do not fire.Let $\overline{Y} \in \mathbb{R}^m$ denote the vector of membrane potentials of a single bucket of the intermediate layer: $\overline{Y} = \mathrm{A}_b\,\overline{X}$. Our WTA layer maps each $\overline{Y}$ into a binary unit-vector $\overline{Z}$ of the same length, in which the only firing neuron corresponds to the neuron with the largest potential in $\overline{Y}$ that is bounded by the threshold parameter τ. As explained in Section 3, the random projection step produces $\ell = O(\log(t_m/\delta))$ random compressed vectors, one for each of the ℓ buckets. Each such copy is an input to an independent WTA circuit and thus, in this section, we focus on our construction restricted to just a single bucket, bearing in mind that in fact our network consists of ℓ repetitions of this module.

The first part of the WTA circuit is devoted to *reading*: the circuit collects firing statistics for a period of $T = \tilde{O}(m^2)$ rounds to obtain a good estimate of the neuron in the bucket that 1) has potential $\leq \tau$ and 2) has the largest firing rate. This neuron corresponds to the neuron with the highest potential in $\overline{Y}$ bounded by τ. This is done by augmenting each neuron i in the bucket with a directed chain H_i of neurons of length T. The j^{th} neuron in the chain triggers the firing of the $(j+1)^{th}$ neuron with high probability. As a result, after T rounds, the number of firing neurons in the chain H_i is equal to the number of times i fired within the last T rounds, with high probability. We thus refer to this H_i chain as the *history* chain of the i^{th} neuron in the bucket. The second part of the circuit first excludes all neurons with potential $\geq \tau$ and then applies a standard WTA circuit to pick the neuron remaining that fires the most in this T-length time interval. See Fig. 2 for an illustration of the overall clustering network and the WTA module. The main result of this section is as follows.

▶ **Lemma 12.** *For every pair of input patterns $\overline{X}_i, \overline{X}_j$ presented over a period of t_m rounds, with probability at least $1-\delta$ the following hold:*

(I) *If $\mathcal{RD}(\overline{X}_i, \overline{X}_j) \geq \Delta$, then $\text{supp}(\overline{Z}_i) \setminus \text{supp}(\overline{Z}_j) \geq 0.9 \cdot \ell$.*

(II) *If $\mathcal{RD}(\overline{X}_i, \overline{X}_j) \leq \Delta/\alpha$, then $\text{supp}(\overline{Z}_i) \cap \text{supp}(\overline{Z}_j) \geq 0.9 \cdot \ell$.*

We first give a detailed description of the specification step via WTA (see Figure 2). We focus on a single bucket, bearing in mind that in fact our network consists of ℓ repetitions of this module.

Reading via History Chain. Every neuron $i \in \{1, \ldots, m\}$ in the bucket is connected to a chain H_i of length $T = \Theta(\log(1/\delta) \cdot m^2)$ of neurons where the j^{th} neuron in this chain fires in round t with high probability iff its incoming neighbor on that chain fires in round $t-1$. This is done by setting the bias value of each neuron to 1 and the edge weights to be $1/2$. As a result we get that the number of firing neurons in this chain equals to the number of times i fires within the last T rounds with high probability.

Omitting the Neurons Exceeding the Threshold Value. For every neuron $i \in \{1, \ldots, m\}$ we introduce an inhibitor copy r_i that has the same incoming weights as i and therefore also has the same potential. We set the bias of r_i such that with high probability it fires iff its potential exceeds the threshold value τ. We then connect r_i to all neurons in the chain H_i with large negative weight. As a result, if the potential of neuron i exceeds τ with high probability all neurons in H_i will not fire.

Selecting the Maximum Firing Rate with Pairwise Comparisons. For every ordered pair of neurons $i, j \in [1, m]$, we have a designated (threshold gate) neuron $y_{i,j}$ that fires iff the i^{th} neuron in the bucket fires more than the j^{th} neuron within the last T rounds. To accomplish this, each of the neurons in the chain H_i (respectively, H_j) is connected to $y_{i,j}$ with a positive (respectively, negative) edge weight of ± 1. Hence, the total weighted sum incoming to $y_{i,j}$ is exactly the difference between $R(i)$ and $R(j)$ where $\overline{R}(i), \overline{R}(j)$ are the number of times that the i^{th} and j^{th} neurons fired in the last T rounds. We set the bias of $y_{i,j}$ such that it fires with high probability iff $\overline{R}(i) - \overline{R}(j) \geq 1$. The i^{th} output neuron in $\overline{Z}$ computes the AND-gate of the threshold-gates $y_{i,1}, \ldots, y_{i,m}$. That is, z_i fires in round t only if every $y_{i,1}, \ldots, y_{i,m}$ fired in round $t-1$. The AND-gate can be implemented by setting the incoming edge weight from each $y_{i,j}$ to z_i to be $1/m$, and the bias of $\overline{Z}_i$ to $1 - 1/(2m)$.

Analysis. The requirement from the WTA module is that the firing frequency vector $\overline{R}$ has its largest entry in the same position as the largest entry of $\overline{Y}$ that is $\leq \tau$. If this is the case, the WTA circuit indeed selects the neuron corresponding to the largest firing rate $\leq \tau$, and the only entry in the support of $\overline{Z}$ is the one corresponding to this entry. For the largest entry in $\overline{R}$ to reflect the largest entry in $\overline{Y} \leq \tau$ with probability $\geq 1-\delta$, the gap between the largest and second largest firing rates must be $\Omega\left(\sqrt{\log(1/\delta)/T}\right)$. Using the gap condition of Corollary 9 we will show that this gap is $\Omega(1/m)$, letting us set $T = O(\log(1/\delta) \cdot m^2)$. The desired gap is achieved in a large fraction of the buckets, this implies that the WTA picks the maximal entry in most of the buckets as well.

▷ Claim 13. Let $\overline{Y}$ be a vector with $i = \arg\max_{j:\overline{Y}(j)\leq\tau} \overline{Y}(j)$ and $\overline{Y}(i) - \overline{Y}(j) = \Omega(p^{1/2}/m)$ for every[3] $j \neq i$ with $\overline{Y}(j) \leq \tau$. Then in the output vector $\overline{Z}$, $\overline{Z}(i) = 1$ and $\overline{Z}(j) = 0$ for every $j \neq i$ with probability at least 99/100. If $\overline{Y}$ is first introduced in round t, the desired output vector $\overline{Z}$ fires in round $t + T + 2$ w.h.p.

The proof of Claim 13 and the complete proof of Lemma 12 is given in Appendix B.

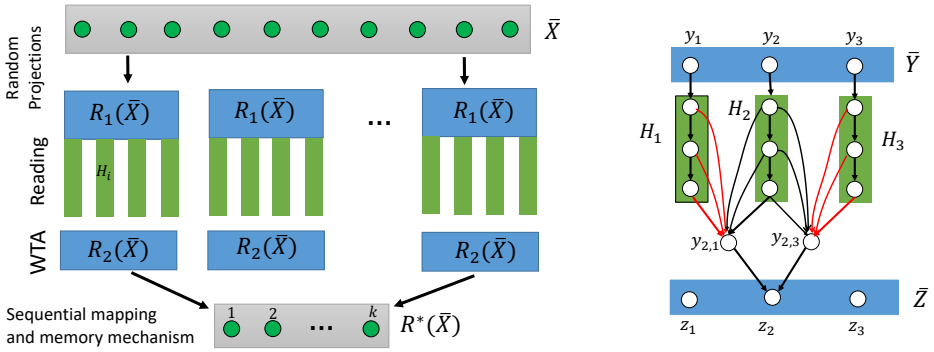


Figure 2 Left: Overall network description, the input pattern $\overline{X}$ is mapped to unique output neuron in $[1, k]$ via three main steps. Right: Description of the WTA circuit. For clarity we only show the connections for the second output neuron, but same holds for all k output neurons. Every input neuron i in $\overline{Y}$ is connected to a history chain H_i of length T that is used to collect firing statistics. For each pair of input neurons i, j, there is a threshold gate $q_{i,j}$ that fires only if i fired at least $T/2m$ more times than j within T rounds. Each history neuron in H_i, H_j is connected with weight 1 (respectively -1) to $q_{i,j}$ and the bias of $q_{i,j}$ is $T/2m$. Finally, each output neuron q_i computes the AND gate of $q_{i,1}, \ldots, q_{i,m}$, i.e., fires only if all these gates fire in the previous round. As a result a winner q_i is selected only if $y_i - y_j = \Omega(1/m)$ for every $j \neq i$.

5 Layer 3: Sequential Mapping

We conclude by discussing the final sequential mapping layer of our network, which maps the *binary* patterns $\overline{Z}_i$ of length $r = O(\ell \cdot m)$ to a single output neuron. The inputs to the third layer are the r neurons $Z = \{z_1, z_2, \ldots, z_r\}$ and its outputs are the k output neurons $Q = \{q_1, q_2, \ldots, q_k\}$. The r-length patterns will be mapped to their unique output neuron in a sequential manner, where at each given round, a newly introduced pattern will be mapped to the available output with the smallest index. The mapping will satisfy the following properties: (1) patterns $\overline{Z}_i, \overline{Z}_j$ that correspond to *far* input patterns $\overline{X}_i, \overline{X}_j$ respectively will be mapped to distinct outputs, (2) patterns $\overline{Z}_i, \overline{Z}_j$ that correspond to *close* input patterns

[3] This required gap is based on Lemma 7/Corollary 9.

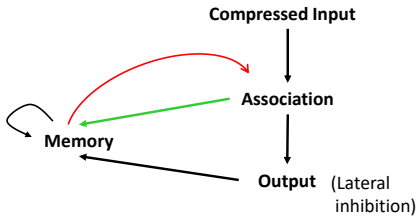


Figure 3 Schematic description.

presented within the same time window of $\Theta(t_m)$ rounds will be mapped to the same outputs. Recall that t_m is the memory duration which is a parameter of the network. A key component in our network is the *memory module* that remembers the association between each previously introduced pattern and its selected output for $\Theta(t_m)$ rounds. Roughly speaking, our network has two intermediate layers: an *association* layer and a *memory* layer (see Figure 3), which we describe below.

We first describe the construction by considering the case where a new pattern $\overline{Z}$ is introduced (and no close pattern to it was introduced before). When $\overline{Z}$ is presented to the network for the first time, it activates the association layer which contains r neurons $a_{i,1}, \ldots, a_{i,r}$ for each output q_i. Let $\text{supp}(\overline{Z})$ be the non-zero entries of $\overline{Z}$. Since[4] $|\text{supp}(\overline{Z})| \leq \ell$ it can activate at most $\ell \cdot k$ many neurons $a_{i,j}$ for every $j \in \text{supp}(\overline{Z})$ and $i \in \{1, \ldots, k\}$. Every output q_i is connected to its association neurons $a_{i,1}, \ldots, a_{i,r}$ and fires only if *many* of them fire.

Our construction will make sure that the number of active *association* neurons of a *taken* output (i.e., output already mapped to other pattern, far from $\overline{Z}$) will be small, which will prevent the firing of these outputs when a far pattern is presented. This will be provided due to the *memory module* appended to each output q_i which remembers the pattern (in fact the cluster of patterns) that were mapped to q_i in the *past*. For each j in the support of the pattern associated with q_i, the memory module corresponding to q_i and z_j inhibits all other association neurons associated with q_i, while activating the association a_{ij}. This association will be remembered – by the memory module – for at least $c_1 \cdot t_m$ rounds and at most $c_2 \cdot t_m$ rounds, for $c_1 < c_2$ with high probability.

For every available output q_i, all its association neurons $a_{i,j}$ for $j \in \text{supp}(\overline{Z})$ will start firing once $\overline{Z}$ is presented, which will in turn activate q_i. To select exactly one output neuron among all the available ones, the output layer is connected via a lateral inhibition, where every neuron q_i inhibits all q_j for $j \geq i + 1$.

Overall, our sequential mapping module satisfies:

▶ **Theorem 14** (The Sequential Mapping Module). *There exists a sequential mapping module with r input neurons, $\widetilde{\Theta}(r \cdot k)$ auxiliary neurons, and k output neurons that for every pattern $\overline{Z}$ that is introduced in round t satisfy the following with probability $1 - \delta$:*

(1) *The pattern $\overline{Z}$ is mapped to one of the outputs $q_1, \ldots, q_k$ in round $t + 6$.*

(2) *Any pair of* close *patterns $\overline{Z}, \overline{Z}'$ introduced within a span of $c_1 \cdot t_m$ rounds are mapped to the same output neuron.*

(3) *Any pair of far patterns $\overline{Z}, \overline{Z}'$ introduced within a span of $c_1 \cdot t_m$ rounds are mapped to different output neurons.*

In addition, if a pattern $\overline{Z}$ (or a pattern close to it) is not introduced for t_m rounds, then its unique mapped output q_j is released after $c \cdot t_m$ rounds, for some constant $c \geq 1$.

[4] As the WTA module picks at most one winning entry in each of the ℓ buckets.

5.1 Complete Network Description of the Sequential Mapping

Next we precisely describe the neurons and edge weights of the sequential mapping sub-network.

The association layer. For each neuron z_i in the input layer, and each neuron q_j in the output layer, we introduce an *association neuron* denoted as $a_{j,i}$. The neuron $a_{j,i}$ has positive and negative incoming edges from the memory modules that is described in the next paragraph. It also has a *positive* incoming edge from the neuron z_i with weight $w(z_i, a_{j,i}) = 2\ell$, and bias $\beta(a_{j,i}) = (19/10)\ell - 1$. We set the connections to this neuron in a way that guarantees it fires only if z_i fired in the previous round, and no other (far) pattern is already mapped to q_j.

The memory modules. For each neuron z_i in the input layer and output neuron q_j we introduce a *memory of association* module $M_{j,i}$ which remembers the association of neuron z_i and q_j. The memory module $M_{j,i}$ contains $c \cdot \log(\frac{1}{\delta'})$ excitatory neurons denoted as $M^+_{j,i}$ where $\delta' = \delta/\ell$ and c is chosen to be a sufficiently large constant. For every $m \in M^+_{j,i}$ we introduce a *feedback* neuron f_m which starts exciting m once the memory module is being activated.In addition, we introduce a delay chain $C^M_j = C_5(q_j)$ that starts at the output q_j and ends at each of the neurons $m \in M^+_{j,i}$. Finally, the memory module contains two head neurons, an excitatory neuron $m^+_{j,i}$ and an inhibitory neuron $m^-_{j,i}$.

Each excitatory neuron $m \in M^+_{j,i}$ has positive incoming edges from $a_{j,i}, q_j, C^M_j$, as well as from the corresponding feedback neuron f_m with the following weights and bias

$$w(a_{j,i}, m) = 2\lambda, \quad \forall u \in C^M_j \quad w(u, m) = 2 \ , \ w(f_m, m) = \lambda \cdot (\chi + 2) + 9 \quad \beta(m) = 9 + 2 \cdot \lambda \ ,$$

where $\chi = \log(t_m - 1)$. Note that if the feedback neuron f_m fired in the previous round, the memory neuron m fires with probability at least $\frac{1}{1+e^{-\chi}} = 1 - 1/t_m$. The feedback neuron f_m for $m \in M^+_{j,i}$ has *positive* incoming edges from m and $m^+_{j,i}$ with weights $w(m, f_m) = 2$, $w(m^+_{j,i}, f_m) = 2$, and bias $\beta(f_m) = 3$. Hence, w.h.p. f_m fires iff m and $m^+_{j,i}$ fired in the previous round. The excitatory head neuron $m^+_{j,i}$ has *positive* incoming edges from all $m \in M^+_{j,i}$ with weights $w(m, m^+_{j,i}) = 2$ and bias $\beta(m^+_{j,i}) = c \cdot \log(1/\delta') + 1$. The head neuron $m^-_{j,i}$ is an inhibitory copy of $m^+_{j,i}$ with the same incoming edges, bias and potential function.

Each association neuron $a_{j,i}$ has a *positive* incoming edge from the head memory neuron $m^+_{j,i}$ with weight $w(m^+_{j,i}, a_{j,i}) = \ell$. In addition, $a_{j,i}$ has *negative* incoming edges from the inhibitory memory neurons $m^-_{j,i'}$ for every $i' = \{1, 2, \ldots k\} \setminus \{i\}$ with weights $w(m^-_{j,i'}, a_{j,i}) = -1$. Note that w.h.p. $a_{j,i}$ fires in round t only if $z_i = 1$ in round $t - 1$. In case where there are at least $1/10\ell$ memory modules $m_{j,i'}$ that inhibit $a_{j,i}$, it fires only if its own memory module, namely, $M_{j,i}$ is active. To prevent a situation of partial memory where only part of the memory modules associated with a pattern are released, if at most 0.9ℓ of the memory modules $M_{j,1}, \ldots, M_{j,r}$ are active, we activate the inhibition of these firing modules. For that purpose, for every output q_j, we introduce 3 *deletion* neurons d^1_j, d^2_j, d^3_j. The neurons d^1_j, d^2_j detect this situation and the inhibitor d^3_j kills the partial memory. The deletion neuron d^1_j has incoming edges from all head neurons $m^+_{j,i}$ for $i = 1 \ldots r$ with weights $w(m^+_{j,i}, d^1_j) = 2$ and bias $\beta(d^1_j) = 1$. Hence, w.h.p. d^1_j fires in round t iff at least one memory module fired in round $t - 1$. The second deletion neuron has incoming edges from all the inhibitor head neurons $m^-_{j,i}$ for $i = 1 \ldots r$ with weights $w(m^-_{j,i}, d^2_j) = -1$ and bias $\beta(d^2_j) = -0.9\ell + 1$. Thus, w.h.p. d^2_j fires in round t iff at most 0.9ℓ memory module fired in round $t - 1$. The third deletion neuron d^3_j has incoming edges from d^1_j and d^2_j with weights $w(d^1_j, d^3_j) = w(d^2_j, d^3_j) = 2$ and bias $\beta(d^3_j) = 3$. Hence, d^3_j fires in round t iff d^1_j and d^2_j fired in round $t - 1$. In addition, the head neurons $m^+_{j,i}, m^-_{j,i}$ have a *negative* incoming edge from d^3_j with weight $w(d^3_j, m^+_{j,i}) = w(d^3_j, m^-_{j,i}) = -2c \log(1/\delta')$.

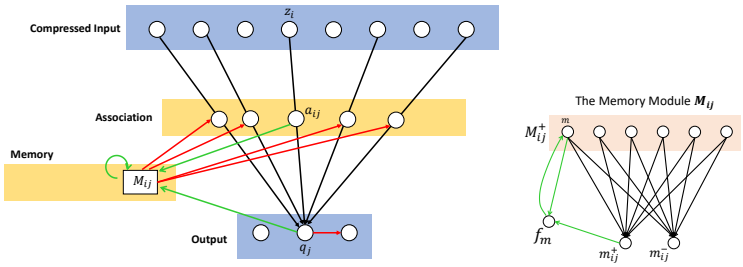


Figure 4 Left: an illustration of the network. The green edges correspond to edges with positive weight where the red edges correspond to negative weights. For simplicity we omitted the history and deletion neurons as well as the rest on the association and memory modules. Right: The memory module and the feedback loop mechanism.

History neurons. If an input pattern $\overline{Z}$ is already mapped to an output neuron, our goal is to map every pattern close to $\overline{Z}$ to the same output. To make sure that close patterns $\overline{Z}, \overline{Z}'$ are indeed mapped to the same output, for each output neuron q_j we introduce an inhibitory *history* neuron h_j. The role of the history neuron is to take care of a situation where a pattern $\overline{Z}$ is mapped to output q_j, but when a close pattern $\overline{Z}'$ is presented later on, an output q_i for $i < j$ is free. Recall that in our construction, each pattern is mapped to the first available output. To do that, the network parameters of the history neurons are defined as follows. Each history neuron h_j has *positive* incoming edges from all associated excitatory memory neurons $m^+_{j,i}$ for $i = 1 \dots r$ with weights $w(m^+_{j,i}, h_j) = 1$. In addition, it has a *positive* incoming edge from the output neuron q_j with weight $w(q_j, h_j) = \ell$ and bias $\beta(h_j) = -(3/2)\ell - 1$. Thus, the history neuron h_j fires if the output neuron q_j fired and at least a large fraction of the memory modules corresponding to q_j are active (the latter indicates that q_j is indeed taken). The history neuron h_j then inhibits all the preceding output neurons $q_1, \dots, q_{j-1}$, preventing the input pattern from being mapped to a different output.

The output layer. The output layer Q consists of excitatory neurons. In order to map the input pattern sequentially, for each $q_j \in Q$ we introduce an inhibitor output neuron q_j^- which inhibits the output neurons $q_{j'}$ for $j' \in \{j+1, \dots, k\}$. The neuron q_j is connected to q_j^- via a delay chain of length 3 denoted as $C_j^I = C_3(q_j)$. The neuron q_j^- has incoming edges from C_j^I with weights 2, and a negative bias of $\beta(q_j^-) = 5$. Hence, w.h.p. q_j^- fires iff q_j fired for 3 consecutive rounds.

Each output neuron q_j has *positive* incoming edges from the association neuron $a_{j,i}$ for every $i = \{1, 2, \dots k\}$. In addition, q_j has *negative* incoming edges from all preceding neurons q_i^- for $i < j$ and all successive history neurons h_i where $i > j$. The weights and bias are given by

$$w(a_{j,i}, q_j) = 2 \;\; \forall i \in [r], \quad w(q_i^-, q_j) = -3\ell \;\; \forall i < j, \quad w(h_i, q_j) = -3\ell \;\; \forall i > j, \quad \beta(q_j) = \ell - 1$$

Note that q_j fires in round t only if at least $(1/2)\ell$ association neurons fired in round $t-1$, and no history or inhibitor output neuron inhibit it.

As in previous sections, we assume that before the first round no neuron fires (i.e $v^0 = 0$ for every neuron v in the network). Figure 4 illustrates the structure of the network and Figure 5 demonstrates the network flow with two inputs.

5.2 Network Dynamics

Before providing the detailed analysis of the network, we give a more detailed description of the network behavior in the two orthogonal cases: mapping close patterns to the same output and mapping far patterns to distinct outputs.

Introduction of a New Pattern $\overline{X}_j$. A pattern $\overline{X}_j$ is introduced in round t where $q_1, \ldots, q_{j-1}$ are already allocated. We will describe how $\overline{X}_j$ is mapped to q_j. First, in Step (1), $\overline{X}_j$ is mapped to a vector $\overline{Y}_j = R_1(\overline{X}_j)$. In Step (2), $\overline{Y}_j$ is mapped to a *binary* vector $\overline{Z}_j$ which is the input to the sequential mapping sub-network. Let t' be the time in which $\overline{Z}_j$ fires. This will cause the firing of the association layer in the following manner. Let $\overline{X}_1, \ldots, \overline{X}_{j-1}$ be the patterns mapped to $q_1, \ldots, q_{j-1}$.

- For every allocated neuron q_i, $i \leq j-1$, and every entry $i_1 \in \text{supp}(Z_j) \setminus \text{supp}(Z_i)$, their association neuron a_{i,i_1} is inhibited by the memory modules M_{i,i_2} for every $i_2 \in \text{supp}(Z_i)$.
- Thus, for every allocated neuron q_i, when introducing Z_j, at most $|\text{supp}(Z_i) \cap \text{supp}(Z_j)| \leq 0.1 \cdot \ell$ association neurons $a_{i,j'}$ are active.
- Since an output q_i fires only if at least $1/2\ell$ association neurons are active, q_i would not fire.
- For every free output q_i for $i \in \{j, \ldots, k\}$, all the association neurons a_{i,i_1} for every $i_1 \in \text{supp}(Z_j)$ are now active. Hence, in the next round, all $q_j, \ldots, q_k$ fire.
- Since we have a lateral inhibition, q_j inhibits[5] all other q_i for $i \in \{j+1, \ldots, k\}$.
- Only at the point where $q_{j+1}, \ldots, q_k$ are inhibited, the memory modules M_{j,i_1} of the winner output q_j start being active, for every $i_1 \in \text{supp}(Z_j)$. This memory module continues firing from that point on for $\Theta(t_m)$ rounds, even when X_j is not introduced.
- Each activated module M_{j,i_1} for every $i_1 \in \text{supp}(Z_j)$ inhibits each of the other association neurons a_{j,i_2} for every $i_2 \neq i_1$. In addition, each M_{j,i_1} excites its own association neuron a_{j,i_1} for $i_1 \in \text{supp}(Z_j)$, thus canceling the inhibition from the other M_{j,i_2} modules. As a result, the only inhibited association neurons are a_{j,i_2} for $i_2 \notin \text{supp}(Z_j)$.

Re-Introduction of a Close-Pattern $\overline{X}_j$. We now consider the situation where $\overline{X}_j$ is introduced in round t, and a close-pattern $\overline{X}_{j'}$ was introduced in the past (e.g., in a window of $\Theta(t_m)$ rounds). We would like to show that $\overline{X}_j$ will be mapped to the exact same output neuron $q_{j'}$ as $\overline{X}_{j'}$.

- For every allocated neuron q_i and every entry $i_1 \in \text{supp}(Z_j) \setminus \text{supp}(Z_i)$, their association neuron a_{i,i_1} is inhibited by the memory modules M_{i,i_2} for every $i_2 \in \text{supp}(Z_i)$.
- Thus, for every allocated neuron q_i for $i \neq j'$, when introducing Z_j, at most $|\text{supp}(Z_i) \cap \text{supp}(Z_j)| \leq 0.1 \cdot \ell$ association neurons a_{i,i_1} are active. As a result, q_{i_1} will not fire.
- In contrast, for the desired output neuron $q_{j'}$, only $|\text{supp}(Z_j) \setminus \text{supp}(Z_{j'})|$ association neurons are inhibited, while the remaining ones, namely, a_{j',i_1} for $i_1 \in \text{supp}(Z_j) \cap \text{supp}(Z_{j'})$ are active. Since $|\text{supp}(Z_j) \cap \text{supp}(Z_{j'})|$ is sufficiently large, $q_{j'}$ will fire.
- Due to lateral inhibition of $q_{j'}$, all other free outputs $q_{i'}$ for $i' \geq j'+1$ will not fire.
- It remains to show that all other *free* outputs q_i for $i \leq j'-1$ will not be active. Recall that these outputs have a lateral inhibition on $q_{j'}$ that starts inhibiting $q_{j'}$ within a small number of rounds since the activation of q_i. It is therefore important to neutralize these outputs before their inhibition on $q_{j'}$ comes into play. Indeed this is the reason for introducing the delay to the lateral inhibition mechanism.

[5] In fact, its inhibitor copy will do this inhibition.

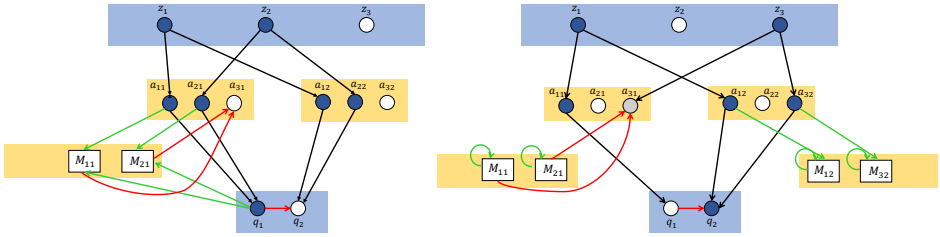


Figure 5 Left: network's state where first pattern $(1,1,0)$ is presented. Since all outputs are free at that point, the pattern is mapped to the first output q_1, which activates all its memory modules. Right: network description when the second input $(1,0,1)$ is presented. Because the memory modules $M_{1,1}$ and $M_{1,2}$ are active, the association neuron $a_{1,3}$ is inhibited and this q_1 will not fire. As a result, $(1,0,1)$ is mapped to q_2, activating corresponding memory modules $M_{2,1}$ and $M_{2,3}$.

- To indicate the fact that $q_{j'}$ was already allocated to a pattern close to X_j, we have a history neuron $h_{j'}$ that works as follows. It gets input from all the memory modules of $q_{j'}$, as well as from $q_{j'}$ itself. Since the close patterns X_j and $X_{j'}$ have many entries in common, sufficiently many memory modules of $q_{j'}$ will activate $h_{j'}$. For a free output q_i for $i \leq j'-1$, the memory modules of q_i are not active and hence the history neuron would not be active.
- The history neuron $h_{j'}$ then inhibits all prior outputs q_i for $i \leq j'-1$ just before their lateral inhibition chain affects $q_{j'}$. In addition, the inhibition on q_i also occurs before the memory modules of q_i start being active. That is, since we want to remember only the association to the correct output $q_{j'}$, we delay the activation of the memory model. The latter starts only after $q_{j'}$ fires for a consecutive constant number of rounds.

5.2.1 Correctness

The following definitions are useful in our context.

▶ **Definition 15.** *A pattern $\overline{Z}$ is* mapped *to an output neuron q_j in round t if when presenting $\overline{Z}$ to the sequential mapping network in round $t-1$, q_j is the only firing output neuron in round t.*

▶ **Definition 16.** *$M_{j,i}$ is* active *in round t, if its head neurons $m^+_{j,i}$, $m^-_{j,i}$ fired in round t.*

In order to prove the main Theorem 14, we start by establishing useful auxiliary claims and observations.

▶ **Observation 17.** *For every output neuron q_j if the number of active memory modules M_{ij} in round t is between 1 and 0.9ℓ, then w.h.p. there are no active memory modules in round $t+3$.*

Proof. For output neuron q_j if the number of active memory modules M_{ij} in round t is at least 1 w.h.p. the deletion neuron d^1_j fires in round $t+1$. If the number of active memory modules M_{ij} is also less than 0.9ℓ then w.h.p. d^2_j fires in round $t+1$ and therefore d^3_j fires in round $t+2$, inhibiting all memory modules M_{ij} for $i=1,\ldots,r$. ◀

▶ **Observation 18.** *Given that the deletion neurons of output q_j did not fire in round $t-1$, w.h.p. a memory module $M_{j,i}$ is active in round t iff at least $(c/2)\log(1/\delta')$ neurons $m \in M^+_{j,i}$ fired in round $t-1$.*

Proof. Recall that a memory module $M_{j,i}$ is *active* in round t if the excitatory neuron $m^+_{j,i}$ fired. The potential function of $m^+_{j,i}$ is given by

$$pot(m^+_{j,i}, t) = \sum_{m \in M^+_{j,i}} 2 \cdot m^{t-1} - 2c\log(1/\delta')(d^3_j)^{t-1} - c\log(1/\delta') + 1 \ .$$

If at least $\frac{c}{2}\log(1/\delta')$ neurons in $M^+_{j,i}$ fired in round $t-1$, the potential of $m^+_{j,i}$ in round t is at least 1 and the probability that $m^+_{j,i}$ fire in round t is at least $\frac{1}{1+e^{-1/\lambda}} \geq 1 - \Theta(\frac{\delta}{n \cdot k \cdot \Delta \cdot t_m \cdot \log 1/\delta})$. On the other hand, if less than $\frac{c}{2}\log(1/\delta')$ neurons in $M^+_{j,i}$ fired, the potential of $m^+_{j,i}$ is at most -1 and the probability that $m^+_{j,i}$ fired in round t is at most $\frac{1}{1+e^{1/\lambda}} \leq \Theta(\frac{\delta}{n \cdot k \cdot \Delta \cdot t_m \cdot \log 1/\delta})$. ◀

▷ Claim 19. If $\overline{Z}_1, \overline{Z}_2$ are close and $\overline{Z}_2, \overline{Z}_3$ are close, then $\overline{Z}_1, \overline{Z}_3$ are close.

Proof. Let $\overline{X}_1, \overline{X}_2, \overline{X}_3$ be the corresponding input patterns, where $\overline{Z}_i = R_2(\overline{X}_i)$ for $i \in \{1, 2, 3\}$. By the definition of the clustering instance, every pair of patterns $\overline{X}_i, \overline{X}_j$ are either with relative distance at least $\Delta/2$ (i.e., if these patterns belong to different clusters), or have relative distance at most Δ/α (i.e., if they belong to the same cluster) for $\alpha = \Omega(\log(1/\Delta))$.

By Lemma 12, input patterns $\overline{X}_i, \overline{X}_j$ that belong to different (resp., same) clusters are mapped to far (resp., close) vectors $\overline{Z}_i, \overline{Z}_j$. We therefore have that $\overline{X}_1, \overline{X}_2$ are in the same cluster, and also $\overline{X}_2, \overline{X}_3$ are in the same cluster, concluding that $\overline{X}_1, \overline{X}_2, \overline{X}_3$ are all in the same cluster. ◁

▷ Claim 20. For every $j \in [k]$ and $i \in [\ell]$ w.h.p. the memory module $M_{j,i}$ is active in round t given that it was not active in round $t-3$, only if C^M_j and $a_{j,i}$ fired in round $t-2$.

Proof. By Observation 18 $M_{j,i}$ is activated in round t only if at least $(c/2)\log(1/\delta')$ neurons $m \in M^+_{j,i}$ fire in round $t-1$. Since $M_{j,i}$ was not active in round $t-3$ all feedback neurons f_m for $m \in M^t_{j,i}$ was not active in round $t-2$ and the potential of each $m \in M^+_{j,i}$ in round $t-1$ is $\sum_{u \in C^M_j} 2 \cdot (u)^{t-2} + 2\lambda \cdot (a_{ij})^{t-2} - 9 - 2\lambda$. Hence, if C^M_j and $a_{j,i}$ fired in round $t-2$, in the next round the potential of each $m \in M^+_{j,i}$ is at least 1 and m fires in round $t-1$ with probability at least $1 - \Theta(\frac{\delta}{n \cdot k \cdot \Delta \cdot t_m \cdot \log 1/\delta})$. Thus, by Chernoff bound w.h.p. at least $(c/2)\log(1/\delta')$ neurons $m \in M^+_{j,i}$ fired in round $t-1$.

On the other hand if C^M_j and $a_{j,i}$ did not fire together in round $t-2$, the potential of every $m \in M^t_{j,i}$ in round $t-1$ is at most -2λ and m fires with probability at most $\frac{1}{1+e^2}$. Using Chernoff bound and choosing c to be sufficiently large, we conclude that $(c/2)\log(1/\delta')$ neurons $m \in M^+_{j,i}$ fire in round $t-1$ with probability at most δ'. ◁

▶ **Observation 21.** *For every output q_j at each round w.h.p. the number of memory modules $M_{j,i}$ that are active is at most ℓ.*

Proof. Since every pattern has at most ℓ non zero entries, in each round at most ℓ association neurons $a_{j,i}$ fire. By Claim 20, at each round at most ℓ memory modules $M_{j,i}$ are activated for the first time. If in round $t-3$ more then 0.1ℓ memory modules were active, the only association neurons firing in round $t-2$ correspond to the activated memory modules and therefore w.h.p. no new modules are activated in round t. Else, by Observation 17 w.h.p. the deletion neuron d^3_j kills the active memory modules and no memory module is active in round t. ◀

Using the same arguments, since the deletion neurons erase the partial memory, we can also conclude that for every output neuron all its active memory modules correspond to the same input pattern.

▶ **Observation 22.** *For each output neuron q_i in each round if it has active memory modules, there exists an input pattern $\overline{Z}$ s.t if $M_{i,j}$ is active then $j \in \text{supp}(\overline{Z})$.*

▷ Claim 23. If $\overline{Z}$ is mapped to q_j in round t, with probability greater than $1-\delta$ at least 0.8ℓ memory modules $M_{j,i}$ where $i \in \text{supp}(\overline{Z})$ are active for $c_1 \cdot t_m$ consecutive round starting from round $t+8$.

Proof. Let $\overline{Z}$ be a pattern mapped to q_j in round t. Recall that we assume persistence and therefore w.h.p. $\overline{Z}$ is also mapped to q_j in rounds $t+1$ to $t+8$.

- First we argue that at least 0.8ℓ of the association neurons $a_{j,i}$ for $i \in \text{sup}(\overline{Z})$ fire in round $t+6$. From Observation 17 either there where no memory modules corresponding to q_j active before $\overline{Z}$ was introduces or at least 0.9ℓ [6]. If there where no memory modules active, all association neurons $a_{j,i}$ for $i \in \text{sup}(\overline{Z})$ fire starting round $t+1$ ahead as long as $\overline{Z}$ persist. Otherwise, since q_j fired in round $t+7$, we conclude that at least 0.5ℓ association neurons $a_{j,i}$ fired in round $t+6$. The association neurons that fired are from the support of $\overline{Z}$ and together with Observation 22 we conclude that the pattern previously mapped to q_j is close to $\overline{Z}$ and at least 0.8ℓ association neurons fired in rounds $t+6$ (due to Lemma 12).
- For $i \in sup(\overline{Z})$ for which a_{ij} fired in rounds $t+6$, we now calculate the probability that M_{ij} is active in round $t+8$. By Observation 18 its enough to calculate the probability that at least $(c/2)\log(1/\delta')$ neurons $m \in M^+_{j,i}$ fired in round $t+7$. The potential function of every $m \in M^+_{j,i}$ is given by

$$\text{pot}(m,t) = \sum_{u \in C^M_j} 2 \cdot (u)^{t-1} + 2\lambda \cdot (a_{ij})^{t-1} + (9 + \lambda \cdot (2+\chi)) \cdot (f_m)^{t-1} - 9 - 2\lambda \ .$$

  Since q_j fires in rounds t to $t+7$, the delay chain C^M_j fired in round $t+6$, and the probability m fires in round $t+7$ is at least $1 - \Theta(\frac{\delta}{n \cdot k \cdot \Delta \cdot t_m \cdot \log 1/\delta})$. Using Chernoff bound with probability greater than $1-\delta/3\ell$ at least $\frac{c\log(1/\delta')}{2}$ neurons in $M_{j,i}$ fire in round $t+7$ and the head memory neuron $m^+_{j,i}$ fires in round $t+8$.
- Next we calculate the probability that $m \in M^+_{j,i}$ fires $c_1 t_m$ consecutive rounds starting round $t+8$ given that $m^+_{j,i}$ fires in round $t+8$. Since $m^+_{j,i}$ fired in round $t+8$, for every $m \in M^+_{j,i}$ that fired in round $t+8$, the feedback neuron f_m is activated in round $t+9$ and m fires in round $t+10$ with probability at least $1-1/t_m$. Hence, the probability $m \in M^+_{j,i}$ fires in rounds $t+8$, $t+9$ and $c_1 t_m$ consecutive rounds is at least $(1-\Theta(\frac{\delta}{n \cdot k \cdot \Delta \cdot t_m \cdot \log 1/\delta}))^2 \cdot (\frac{1}{e^{c_1}})$. We chose c_1 such that this is greater than $1/2$. Thus, using Chernoff bound and a large enough c (depending on c_1) the probability that at least $\frac{c\log(1/\delta')}{2}$ neurons $m \in M^+_{j,i}$ fire in rounds $t+8$, $t+9$ and then for $c_1 t_m$ consecutive rounds is at least $1-\delta'/3 = 1-\delta/3\ell$.
- Summing things up, the probability $M_{j,i}$ is active for $c_1 \cdot t_m$ consecutive rounds from round $t+8$ ahead is at least the probability that $m^+_{j,i}$ fired in round $t+8$ and $\frac{c\log(1/\delta')}{2}$ neurons in $M^+_{j,i}$ fires $c_1 t_m$ consecutive rounds starting round $t+8$. By union bound this probability is greater than

$$1 - 3 \cdot (\delta/3\ell) = 1 - \delta/\ell.$$

Thus, we conclude that the probability all 0.8ℓ modules $M_{j,i}$ s.t $a_{j,i}$ fired in round $t+6$ are active for $c_1 \cdot t_m$ consecutive rounds is greater than $1-\delta$. ◁

We are now ready to prove the correctness of the sequential mapping step.

[6] up too ± 3 rounds, but since we assume persistence its ok.

Proof of Theorem 14

Proof. We start by proving the 3 main properties of the network. Given a pattern $\overline{Z}$ introduced in round t we will show:

(1) $\overline{Z}$ is mapped to one of the outputs $q_1, \ldots, q_k$ in round $t+6$

(2) For any pattern $\overline{Z}'$ which is *close* to $\overline{Z}$ and was introduced within a span of $c_1 \cdot t_m$ rounds from t, $\overline{Z}$ and $\overline{Z}'$ are mapped to the same output neuron.

(3) For any pattern $\overline{Z}'$ which is *far* from $\overline{Z}$ and was introduced within a span of $c_1 \cdot t_m$ rounds from t, $\overline{Z}$ and $\overline{Z}'$ are mapped to a different output neuron.

By induction on the order of arrival of the patterns. Let $\overline{Z}$ be the first pattern arrived in round 0. We show that $\overline{Z}$ is mapped to the first (available) neuron q_1 in round 6. For every $i \in \sup(\overline{Z})$ the potential function of the association neuron $a_{1,i}$ is given by:

$$pot(a_{1,i}, t) = 2\ell(z_i)^{t-1} + \ell(m^+_{1,i})^{t-1} - \sum_{j \neq i}(m^-_{1,j})^{t-1} - (19/10)\ell - 1 \ .$$

Since $\overline{Z}$ is the first pattern seen, no neuron has fired in round zero and $\mathrm{pot}(a_{1,i}, 1) = (1/10)\ell - 1 > 1$, and w.h.p. each $a_{1,i}$ for $i \in \sup(\overline{Z})$ fires in round 1.

Since q_1 is the first output, no preceding output neuron inhibits it, and its potential is:

$$\mathrm{pot}(q_1, t) = \sum_{i=1}^{r}(2 \cdot a_{1,i})^{t-1} - \sum_{i=2}^{k} 3\ell h_i - \ell + 1 \ .$$

By Claim 13, w.h.p. each input pattern $\overline{Z}$ (to the sequential mapping network) has at least 0.98ℓ non-zero entries (and at most ℓ). Therefore, at least 0.98ℓ association neurons $a_{1,i}$ excite q_1 in round 2. Recall that the history neurons h_i fire only if at least $1/2$ of the corresponding memory modules are active in the previous round. Hence w.h.p. in round 1, no history neuron fires.

We conclude that q_1 fires in round 2 w.h.p. By Claim 20 every memory module $M_{i,j}$ becomes active only after having q_i firing for 5 consecutive rounds (due to the delay chain C^M_i). For that reason, no memory module fires before round 5. Since the memory neurons are not active, $a_{1,i}$ keeps firing in rounds 1 to 6, and q_1 keeps firing in rounds 2 to 7. Since q_1 is connected to q^-_1 via a delay chain C^I_1 of length 3, starting round 5 (and as long as q_1 fires), the inhibitor q^-_1 inhibits all other output neuron q_i for $i \geq 2$. Thus, for every $i \geq 2$ the potential of q_i in round 6 is at most $\ell - 2\ell - 1/2\ell + 1 < -1$. As a result, for $i \geq 2$ neuron q_i does not fire starting round 6.

We next argue that at this point, no memory modules are yet active and consequently the history neurons are inactive as well. This is due to the fact that the delay in the inhibition of q_i by q_1 is *shorter* then the delay chain C^M_i that starts at q_i and ends at the memory modules. Thus q_i is inhibited before its memory modules are activated. We conclude that if $\overline{Z}$ is observed, starting from round 6, the output neuron q_1 is the only active output neuron, and $\overline{Z}$ is *mapped* to q_1.

Assume the claim holds for the first $i-1$ presented patterns, we next consider the i^{th} pattern $\overline{Z}$ presented in round t.

- We first show that for every $\overline{Z}'$ that is far from $\overline{Z}$ introduced in round $t' \in [t - c_1 \cdot t_m, t-1]$, the pattern $\overline{Z}$ will be mapped to a different output. By the induction assumption, $\overline{Z}'$ is mapped in round $t'+6$ to some output neuron, q_j. Since $\overline{Z}'$ was introduced within $c_1 \cdot t_m$ rounds, by Claim 23 at least 0.8ℓ many memory modules $M_{j,i}$ for $i \in \sup(\overline{Z}')$ are active it round t.

Let $\overline{X}$, $\overline{X}'$ be the inputs corresponding to $\overline{Z}$ and $\overline{Z}'$ respectively. From Lemma 12, $\|\sup(\overline{Z}) \cap \sup(\overline{Z'})\| \leq 0.1\ell$. By Observation 21, the number of active memory modules $M_{j,i}$ in round t is at most ℓ. Thus, the number of active memory modules $M_{j,i}$ in round t for $i \in \sup(\overline{Z})$ is at most $0.2\ell + 0.1\ell = 0.3\ell$.

For every association neuron $a_{j,i}$ whose memory module $M_{j,i}$ is inactive in round t, there are at least 0.8ℓ memory modules that inhibit it. Therefore its potential is $2\ell - 8/10\ell - (19/10)\ell + 1 < -1$, and w.h.p. it does not fire in round $t+1$. Overall, at most 0.3ℓ association neurons $a_{j,i}$ start firing from round $t+1$ and as long as the pattern persists. We conclude that q_j will stop firing from round $t+2$.

- Next we show that two close patterns $\overline{Z}'$ and $\overline{Z}$, introduced within a span of $c_1 \cdot t_m$ rounds are mapped to the same output. First note that by Claim 19, all patterns that are close to $\overline{Z}$ are close to each other. Hence, by the induction assumption, all patterns close to $\overline{Z}$ introduced within the last $c_1 t_m$ rounds were mapped to the same output. We now consider a pattern $\overline{Z}'$ close to $\overline{Z}$ introduced in round $t' \in [t - c_1 \cdot t_m, t - 10]$. By the induction assumption, the pattern $\overline{Z}'$ was mapped to output q_j in round $t' + 6$. By claim 23, 0.8ℓ many memory modules $M_{j,i}$ are active in round $t' + 7 < t$ onward (i.e., for $\Theta(t_m)$ rounds). Combining with Lemma 12 because $\overline{Z}$ is close to $\overline{Z}'$, at least 0.7ℓ many memory modules $M_{j,i}$, $i \in \sup(\overline{Z})$ are active in round t. Since there are at most ℓ many active memory modules associated with q_j in round t, the potential of the association neuron $a_{j,i}$ for which the memory neuron is active in round $t+1$ is at least $2\ell + \ell - (\ell - 1) - 1.9\ell - 1 = 0.1\ell > 1$. We have that at least 0.7ℓ many association neuron $a_{j,i}$ fire in round $t+1$, leading to the firing of q_j in round $t+2$.

  Next, because at least 0.8ℓ memory modules $M_{j,i}$ are active from round $t+3$ ahead, the history neuron h_j fires, and by that inhibits all output neurons q_i for $i \leq j-1$ starting from round $t+4$. Recall that every inhibitor neuron q_i^- for $i \leq j-1$ starts firing only after the delay chain C_i^I fired, i.e. after 3 rounds that q_i fired. Hence the history neuron h_j inhibits every q_i for $i \leq j-1$, just before q_i^- starts firing. We next show that no other q_i fires for $i \geq j+1$. This holds since q_j^- inhibits any such q_i in round $t+6$ via the delay chain C_j^I. Finally, we show that q_j continues firing as long the pattern persists. Because at least 0.5ℓ of the association neurons $a_{j,i}$ are firing, and no preceding inhibitor q_i^- is currently firing (thanks to the history neuron h_j), it remains to show that no other history neuron h_i for $i \neq j$ inhibits q_j. By the induction assumption, all patterns close to $\overline{Z}$ were mapped to q_j. Hence if for some other output q_i for $i \neq j$, at least 0.5ℓ of its associated memory modules are active, by Observation 17 at least 0.9ℓ memory modules are active. Since the pattern $\overline{Z}''$ that was mapped to q_i is far from $\overline{Z}$, we have that at most 0.2ℓ many memory modules $M_{i,i'}$ for $i' \in sup(\overline{Z})$ are active, thus q_i does not fire and consequently h_i does not fire.
- We now consider the case of a newly presented pattern, i.e., no close pattern to it has been presented in the last $\Theta(t_m)$ rounds. We will show that in such a case, $\overline{Z}$ will be mapped to the left-most available output q_j, where by available we mean that no memory module $M_{j,i}$ is active in round t. Let $q_{i_1}, q_{i_2} \ldots q_{i_s}$ be the available output neurons in round t. Hence all the association neurons $a_{i_1,i}$ for $i \in \sup(\overline{Z})$ start firing in round $t+1$. This is because no memory module $M_{i_1,j}$ is active. Thus, in round $t+2$ the output neuron q_{i_1} starts firing. As for the unavailable neurons q_j, by Observation 17 at least 0.9ℓ memory modules M_{ji} are active and by Observation 22 they are associated with a pattern $\overline{Z''}$ which was mapped to q_j. The pattern $\overline{Z''}$ is far from $\overline{Z}$ ($\overline{Z}$ is a new pattern) and therefore at most 0.2ℓ memory module $M_{j,i}$ for $i \in \sup(\overline{Z})$ are active in round t. Hence, at most 0.2ℓ association neurons $a_{j,i}$ fire starting round $t+1$ and w.h.p. q_j will

not fire starting round $t+2$ (and also no history neuron inhibits q_{i_1}). Since we assume persistence, and due to the delay in the activation of the memory modules, q_{i_1} fires also in rounds $t+3$ and $t+4$, and in round $t+5$ the inhibitor $q^-_{i_1}$ starts firing, inhibiting all the successive output neurons q_j for $j \geq i_1+1$.

In order to finish the proof of Theorem 14 we will prove th following Lemma.

▶ **Lemma 24** (Reset, Clearance of Memory). *Let $\overline{Z}$ be a pattern last introduced in round t and mapped to q_j. If no close pattern $\overline{Z}'$ is introduced in rounds $[t, t+c_2 \cdot t_m]$, then q_j is released in some round $\tau \leq t+c_2 t_m$, i.e., all memory modules $M_{j,i}$ stop firing with probability greater than $1-\delta$.*

Proof. Let $\overline{Z}$ be a pattern last introduced in round t and mapped to q_j. By Observation 17 if in some round τ there are less than 0.9ℓ memory modules corresponding to q_j firing, after 3 round 0 memory modules are active and w.h.p. q_j is released. As long as there are at least 0.9ℓ memory modules firing, since all patterns introduced in rounds t to $t+c_2 t_m$ are far from $\overline{Z}$ by the same arguments used in Lemma 14 starting from round $t+1$ less than 0.5ℓ association neurons associated with q_j fire and q_j will not fire for $c_2 t_m$ consecutive rounds starting from round $t+2$ (as long as it is not already released). Thus, from round $t+7$ ahead w.h.p. all neurons in the delay chain C^M_j do not fire.

Therefore, the probability neuron $m \in M^+_{j,i}$ fires in round $\tau \in [t+8, t+c_2 t_m]$ given that f_m did not fire in round $\tau-1$ is at most $\Theta(\frac{\delta}{\log 1/\delta \cdot n \cdot k \cdot \Delta \cdot t_m})$. Moreover, by union bound the probability that there exists a neuron $m \in M^+_{j,i}$ that fired in some round $\tau \in [t+8, t+c_2 t_m]$ given that f_m did not fire in round $\tau-1$ is at most $\delta/2\ell$.

Next we calculate the probability at least half of the neurons $m \in M^+_{j,i}$ fire for $c_2 t_m$ consecutive rounds. Because the delay chain C^M_j do not fire starting round $t+7$ the potential of each neuron $m \in M^+_{j,i}$ in round $t' \in [t+7, t+c_2 t_m]$ is bounded by $\lambda \cdot (\chi+2) = \lambda(\log(t_m-1)+2)$. Hence, the probability $m \in M^+_{j,i}$ fires in round t' is at most $1 - \frac{1}{e^2(t_m-1)+1} < 1 - \frac{1}{e^2 t_m}$. We conclude that the probability a neuron $m \in M^+_{j,i}$ fires for $c_2 t_m$ consecutive rounds is at most e^{c_2/e^2} which for $c_2 > e^2 log(3)$ is less than $1/3$. Using Chernoff bound and a sufficient large c (constant depending on c_2) the probability that at least $(c/2)\log(1/\delta')$ neurons in $M^+_{j,i}$ fire for $c_2 \cdot t_m$ consecutive rounds starting round $t+7$ is at most $\delta/2\ell$.

If $m^+_{j,i}$ fires for $c_2 t_m$ consecutive rounds starting round $t+8$, be Observation 18 at each round at least $1/2$ of the neurons in $M^+_{j,i}$ fired. Given that no neuron $m \in M^+_{j,i}$ fires in round $\tau \in [t+8, t+c_2 t_m]$ unless f_m fired in round $\tau-1$, the head neuron $m^+_{j,i}$ fires for $c_2 t_m$ consecutive rounds only if at least $1/2$ of the neurons in $M^+_{j,i}$ fires for $c_2 t_m$ consecutive rounds. Thus we conclude that $m^+_{j,i}$ fired for $c_2 t_m$ consecutive rounds starting round $t+8$ with probability at most $\delta/(2\ell) + \delta/(2\ell) = \delta/\ell$. Note that by Observation 21 at most ℓ memory neurons M_{ij} are active at each round and using union bound we conclude that with probability at least $1-\delta$ the output neuron q_j is release in round $\tau < t+c_2 t_m$ ◀

This concludes Theorem 14 and therefore also Theorem 3. ◀

References

1 Jayadev Acharya, Arnab Bhattacharyya, and Pritish Kamath. Improved bounds for universal one-bit compressive sensing. In *2017 IEEE International Symposium on Information Theory (ISIT)*, pages 2353–2357, 2017.

2 Zeyuan Allen-Zhu, Rati Gelashvili, Silvio Micali, and Nir Shavit. Sparse sign-consistent Johnson–Lindenstrauss matrices: Compression with neuroscience-based constraints. *PNAS*, 2014.

3 Cornelia I Bargmann and Eve Marder. From the connectome to brain function. *Nature Methods*, 10(6):483–490, 2013.

4 Ella Bingham and Heikki Mannila. Random projection in dimensionality reduction: applications to image and text data. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 245–250. ACM, 2001.

5 Burton H Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970.

6 Petros T Boufounos and Richard G Baraniuk. 1-bit compressive sensing. In *42nd Annual Conference on Information Sciences and Systems (CISS 2008).*, pages 16–21. IEEE, 2008.

7 Christos Boutsidis, Anastasios Zouzias, and Petros Drineas. Random projections for k-means clustering. In *Advances in Neural Information Processing Systems 23 (NIPS)*, 2010.

8 Emmanuel J Candes and Terence Tao. Near-optimal signal recovery from random projections: Universal encoding strategies? *IEEE transactions on information theory*, 52(12):5406–5425, 2006.

9 Sophie JC Caron, Vanessa Ruta, LF Abbott, and Richard Axel. Random convergence of olfactory inputs in the Drosophila mushroom body. *Nature*, 497(7447):113, 2013.

10 Chi-Ning Chou, Kai-Min Chung, and Chi-Jen Lu. On the algorithmic power of spiking neural networks. *arXiv preprint*, 2018. `arXiv:1803.10375`.

11 Kenneth L Clarkson and David P Woodruff. Low rank approximation and regression in input sparsity time. In *Proceedings of the 45th Annual ACM Symposium on Theory of Computing (STOC)*, 2013.

12 Michael B Cohen, Sam Elder, Cameron Musco, Christopher Musco, and Madalina Persu. Dimensionality reduction for k-means clustering and low rank approximation. In *Proceedings of the 47th Annual ACM Symposium on Theory of Computing (STOC)*, 2015.

13 Graham Cormode and Shan Muthukrishnan. An improved data stream summary: the count-min sketch and its applications. *Journal of Algorithms*, 55(1):58–75, 2005.

14 Robert Coultrip, Richard Granger, and Gary Lynch. A cortical model of winner-take-all competition via lateral inhibition. *Neural Networks*, 5(1):47–54, 1992.

15 Anirban Dasgupta, Ravi Kumar, and Tamás Sarlós. Fast locality-sensitive hashing. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1073–1081. ACM, 2011.

16 Sanjoy Dasgupta, Timothy C Sheehan, Charles F Stevens, and Saket Navlakha. A neural data structure for novelty detection. *Proceedings of the National Academy of Sciences*, 115(51):13093–13098, 2018.

17 Sanjoy Dasgupta, Charles F Stevens, and Saket Navlakha. A neural algorithm for a fundamental computing problem. *Science*, 358(6364):793–796, 2017.

18 Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on Computational geometry*, pages 253–262. ACM, 2004.

19 David L Donoho. Compressed sensing. *IEEE Transactions on information theory*, 52(4):1289–1306, 2006.

20 Surya Ganguli and Haim Sompolinsky. Compressed sensing, sparsity, and dimensionality in neuronal information processing and data analysis. *Annual Review of Neuroscience*, 2012.

21 Simon S Haykin. *Neural networks and learning machines*, volume 3. Pearson, 2009.

22 Yael Hitron and Merav Parter. Counting to Ten with Two Fingers: Compressed Counting with Spiking Neurons. *ESA*, 2019. `arXiv:1902.10369`.

23 John J Hopfield, David W Tank, et al. Computing with neural circuits- A model. *Science*, 233(4764):625–633, 1986.

24 Laurent Jacques, Jason N Laska, Petros T Boufounos, and Richard G Baraniuk. Robust 1-bit compressive sensing via binary stable embeddings of sparse vectors. *IEEE Transactions on Information Theory*, 59(4):2082–2102, 2013.

25 Robert T Knight. Contribution of human hippocampal region to novelty detection. *Nature*, 383(6597):256, 1996.

26 Christof Koch and Shimon Ullman. Shifts in selective visual attention: towards the underlying neural circuitry. In *Matters of intelligence*, pages 115–141. Springer, 1987.

27 Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 2015.

28 Robert A. Legenstein, Wolfgang Maass, Christos H. Papadimitriou, and Santosh Srinivas Vempala. Long Term Memory and the Densest K-Subgraph Problem. In *9th Innovations in Theoretical Computer Science Conference, ITCS 2018, January 11-14, 2018, Cambridge, MA, USA*, pages 57:1–57:15, 2018.

29 Andrew C Lin, Alexei M Bygrave, Alix De Calignon, Tzumin Lee, and Gero Miesenböck. Sparse, decorrelated odor coding in the mushroom body enhances learned odor discrimination. *Nature neuroscience*, 17(4):559, 2014.

30 Adi Livnat and Christos Papadimitriou. Evolution and learning: used together, fused together. A response to Watson and Szathmáry. *Trends in Ecology & Evolution*, 31(12):894–896, 2016.

31 Nikos K Logothetis. What we can do and what we cannot do with fMRI. *Nature*, 453(7197):869, 2008.

32 Nancy Lynch and Cameron Musco. A Basic Compositional Model for Spiking Neural Networks. *arXiv preprint*, 2018. `arXiv:1808.03884`.

33 Nancy Lynch, Cameron Musco, and Merav Parter. Computational Tradeoffs in Biological Neural Networks: Self-Stabilizing Winner-Take-All Networks. In *Proceedings of the 8th Conference on Innovations in Theoretical Computer Science (ITCS)*, 2017.

34 Nancy Lynch, Cameron Musco, and Merav Parter. Neuro-RAM Unit with Applications to Similarity Testing and Compression in Spiking Neural Networks. In *Proceedings of the 31st International Symposium on Distributed Computing (DISC)*, 2017.

35 Nancy Lynch, Cameron Musco, and Merav Parter. Spiking Neural Networks: An Algorithmic Perspective. In *5th Workshop on Biological Distributed Algorithms (BDA 2017)*, July 2017.

36 Wolfgang Maass. Networks of spiking neurons: the third generation of neural network models. *Neural Networks*, 10(9):1659–1671, 1997.

37 Wolfgang Maass. On the computational power of winner-take-all. *Neural computation*, 12(11):2519–2535, 2000.

38 Christos H Papadimitriou and Santosh S Vempala. Cortical learning via prediction. In *Conference on Learning Theory*, pages 1402–1422, 2015.

39 Christos H Papadimitriou and Santosh S Vempala. Random Projection in the Brain and Computation with Assemblies of Neurons. In *10th Innovations in Theoretical Computer Science Conference (ITCS 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

40 Narender Ramnani and Adrian M Owen. Anterior prefrontal cortex: insights into function from anatomy and neuroimaging. *Nature Reviews. Neuroscience*, 5(3):184, 2004.

41 Charan Ranganath and Gregor Rainer. Cognitive neuroscience: Neural mechanisms for detecting and remembering novel events. *Nature Reviews Neuroscience*, 4(3):193, 2003.

42 Tamas Sarlos. Improved approximation algorithms for large matrices via random projections. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2006.

43 Olaf Sporns, Giulio Tononi, and Rolf Kötter. The human connectome: a structural description of the human brain. *PLoS Computational Biology*, 1(4):e42, 2005.

44 Leslie G Valiant. *Circuits of the Mind*. Oxford University Press on Demand, 2000.

45 Leslie G Valiant. Memorization and association on a realistic neural model. *Neural computation*, 17(3):527–555, 2005.

46 Leslie G Valiant. Capacity of Neural Networks for Lifelong Learning of Composable Tasks. In *Proceedings of the 58th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 367–378, 2017.

47 Santosh S Vempala. *The random projection method*, volume 65. American Mathematical Society, 2005.

A Additional Proofs: Random Projection

We first prove Lemma 5, that a Chi-squared distribution is nearly uniform within a constant number of standard deviations from its mean.

▶ **Lemma 5.** *Let $\mathcal{D}_p$ be the Chi-squared distribution with p degrees of freedom. For any c with $1 \leq c < p^{1/2}$ there are constants c_ℓ, c_u (depending on c) such that, for any interval $[r_1, r_2] \subseteq \left[p - cp^{1/2}, p + cp^{1/2}\right]$, we have:*

$$\frac{c_\ell(r_2 - r_1)}{p^{1/2}} \leq \Pr_{x \sim \mathcal{D}_p}[x \in [r_1, r_2]] \leq \frac{c_u(r_2 - r_1)}{p^{1/2}}$$

That is, $\mathcal{D}_p$ is roughly uniform on the range $\left[p - cp^{1/2}, p + cp^{1/2}\right]$.

Proof. It is well known that $\mathcal{D}_p$ has mean p, density $d(x) = \frac{1}{2^{p/2}\Gamma(p/2)} x^{p/2-1} e^{-x/2}$. Since we assume $p^{1/2} > c \geq 1$ we have $p \geq 2$ and the distribution has mode $p - 2$. Additionally, we have $p - cp^{1/2} > 0$. So for $x \in \left[p - cp^{1/2}, p + cp^{1/2}\right]$ we can bound:

$$d(x) \leq d(p-2) = \frac{1}{2^{p/2}\Gamma(p/2)}(p-2)^{p/2-1}e^{-p/2+1} \leq \frac{1}{\Gamma(p/2)} \cdot \left(\frac{p}{2e}\right)^{p/2-1}$$

By Stirling's approximation, $\Gamma(p/2) \geq \sqrt{\frac{2\pi}{p/2}}\left(\frac{p}{2e}\right)^{p/2}$ which gives:

$$d(x) \leq \sqrt{\frac{p}{4\pi}} \cdot \frac{2e}{p} = \frac{e}{\sqrt{\pi} \cdot p^{1/2}}. \tag{2}$$

On the other side, since $p - cp^{1/2} > 0$, and since the density of the Chi-squared distribution is monotonically decreasing as x moves further from the mode $p - 2$ either left or right:

$$d(x) \geq \min(d(p - cp^{1/2}), d(p + cp^{1/2})). \tag{3}$$

We lower bound each term in the minimum.

$$\begin{aligned} d(p - cp^{1/2}) &= \frac{1}{2^{p/2}\Gamma(p/2)}(p - cp^{1/2})^{p/2-1}e^{-p/2+(c/2)p^{1/2}} \\ &= \frac{1}{2\Gamma(p/2)}\left(\frac{p}{2e}\right)^{p/2-1} \cdot \left(1 - \frac{c}{p^{1/2}}\right)^{p/2-1} \cdot e^{(c/2)p^{1/2}-1} \end{aligned}$$

Again using Stirling's approximation, and a similar argument to the proof of (2), for some constant c_1, $\frac{1}{2\Gamma(p/2)}\left(\frac{p}{2e}\right)^{p/2-1}$ is lower bounded by $\frac{c_1}{p^{1/2}}$. Thus,

$$\begin{aligned} d(p - cp^{1/2}) &\geq \frac{c_1}{p^{1/2}} \cdot \left(1 - \frac{c}{p^{1/2}}\right)^{p/2-1} \cdot e^{(c/2)p^{1/2}-1} \\ &\geq \frac{c_1}{p^{1/2}} \cdot \left(1 - \frac{c}{p^{1/2}}\right)^{\left(\frac{p^{1/2}}{c}-1\right)\cdot(c/2)p^{1/2}} \cdot e^{(c/2)p^{1/2}-1} \cdot \left(1 - \frac{c}{p^{1/2}}\right)^{(c/2)p^{1/2}-1} \\ &\geq \frac{c_1}{p^{1/2}} \frac{1}{e^{(c/2)p^{1/2}}} \cdot e^{(c/2)p^{1/2}-1} \cdot \left(1 - \frac{c}{p^{1/2}}\right)^{(c/2)p^{1/2}-1} \\ &\geq \frac{c_1}{ep^{1/2}} \cdot \left(1 - \frac{c}{p^{1/2}}\right)^{\left(\frac{p^{1/2}}{c}-1\right)\cdot(c^2/2)+(c^2/2-1)} \\ &\geq \frac{c_1 \cdot e^{c^2/2}}{ep^{1/2}} \cdot \left(1 - \frac{c}{p^{1/2}}\right)^{c^2/2-1} \geq \frac{c'}{p^{1/2}} \end{aligned} \tag{4}$$

for some constant c' that depends on c. We give a similar bound for $p + cp^{1/2}$.

$$\begin{aligned} d(p+cp^{1/2}) &= \frac{1}{2\Gamma(p/2)}\left(\frac{p}{2e}\right)^{p/2-1}\cdot\left(1+\frac{c}{p^{1/2}}\right)^{-p/2-1}\cdot e^{(c/2)p^{1/2}-1} \\ &\geq \frac{c_1}{p^{1/2}}\cdot\left(1+\frac{c}{p^{1/2}}\right)^{-p/2-1}\cdot e^{(c/2)p^{1/2}-1} \\ &= \frac{c_1}{p^{1/2}}\cdot\left(1+\frac{c}{p^{1/2}}\right)^{-\frac{p^{1/2}}{c}\cdot(c/2)p^{1/2}}\cdot e^{(c/2)p^{1/2}-1}\cdot\left(1+\frac{c}{p^{1/2}}\right)^{-1} \\ &\geq \frac{c}{ep^{1/2}}\cdot\left(1+\frac{c}{p^{1/2}}\right)^{-1} \geq \frac{c'}{p^{1/2}} \end{aligned} \tag{5}$$

for some c'. Combining (4) and (5) with (3) and (2) gives that there exist constants c_ℓ, c_u such that for all $x \in [p - cp^{1/2}, p + cp^{1/2}]$,

$$\frac{c_\ell}{p^{1/2}} \leq d(x) \leq \frac{c_u}{p^{1/2}}.$$

Thus for any r_1, r_2:

$$\frac{c_\ell(r_2 - r_1)}{p^{1/2}} \leq \Pr_{x\sim\mathcal{D}_p}[x \in [r_1, r_2]] \leq \frac{c_u(r_2 - r_1)}{p^{1/2}},$$

completing the lemma. ◀

We next give a complete proof of Lemma 7.

A.1 Proof of Lemma 7

Since each $[A_b\overline{X}](i)$ is a Chi-squared random variable with p degrees of freedom, which has median $\leq p$, each $[A_b\overline{X}](i)$ is upper bounded by $p \leq \tau = p + 2p^{1/2}$ with probability $\geq 1/2$. Thus, by Lemma 5 applied with $c = 2$, conditioned on $[A_b\overline{X}](i) \leq p + 2p^{1/2}$, there is some c_ℓ with:

$$\Pr\left[[A_b\overline{X}](i) \in \left[p, p + 2p^{1/2}\right] \middle| [A_b\overline{X}](i) \leq p + 2p^{1/2}\right] \geq \frac{c_\ell \cdot 2p^{1/2}}{p^{1/2}} = 2c_\ell.$$

Thus, for large enough constant c_1 and $m = c_1$, with probability at least $\frac{199}{200}$, we have $i_{1,b}(\overline{X}) \neq 0$ and $[A_b\overline{X}](i_{1,b}(\overline{X})) \geq p$. Call this event $\mathcal{E}_1$. Condition on the event that $\mathcal{E}_1$ occurs and, in particular, that $[A_b\overline{X}](i_{1,b}(\overline{X})) = x$ for any $x \in [p, p + 2p^{1/2}]$. Call this event $\mathcal{E}_{1,x}$. Then for all $j \neq i_{1,b}(\overline{X})$, $[A_b\overline{X}](j)$ is an independent Chi-squared random variable with p degrees of freedom conditioned on either 1) $[A_b\overline{X}](j) \leq x$ or 2) $[A_b\overline{X}](j) \geq p + 2p^{1/2}$. Since $[A_b\overline{X}](j) \leq p \leq x$ with probability at least $1/2$, this conditioning at most doubles the density at any one value. Thus, by Lemma 5,

$$\Pr\left[[A_b\overline{X}](j) \in \left[x - \frac{p^{1/2}}{c_2 m}, x\right] \middle| \mathcal{E}_{1,x}\right] \leq \frac{2c_u \cdot \frac{p^{1/2}}{c_2 m}}{p^{1/2}}.$$

By a union bound, we thus have:

$$\Pr\left[\exists j :\ [A_b\overline{X}](j) \in \left[x - \frac{p^{1/2}}{c_2 m}, x\right] \middle| \mathcal{E}_{1,x}\right] \leq \frac{2c_u \cdot \frac{p^{1/2}}{c_2}}{p^{1/2}} = \frac{2c_u}{c_2}.$$

Setting c_2 sufficiently large ensures that this quantity is bounded by $\frac{1}{200}$. Thus, by a union bound with the probability that $\mathcal{E}_1$ occurs, with probability $\geq \frac{99}{100}$: $i_{1,b}(\overline{X}) \neq 0$ and 2) no $[A_b\overline{X}](j)$ falls in $\left[x - \frac{p^{1/2}}{mc_2}, x\right] = \left[[A_b\overline{X}](i_{1,b}(\overline{X})) - \frac{p^{1/2}}{mc_2}, [A_b\overline{X}](i_{1,b}(\overline{X}))\right]$. This completes the proof.

A.2 Proof of Lemma 8

We first use the relative distance assumption to give a basic claim:

▷ Claim 25. Write $\overline{X}_1, \overline{X}_2$ as $\overline{X}_1 = \chi + \delta_1$ and $\overline{X}_2 = \chi + \delta_2$ where $\chi \in \{0,1\}^n$ is the common vector with $\chi(i) = 1$ iff $\overline{X}_1(i) = \overline{X}_2(i) = 1$. Note that since $\|X_1\| = \|X_2\| = p$ we have $\|\delta_1\| = \|\delta_2\|$. Letting $\Delta = \mathcal{RD}(\overline{X}_1, \overline{X}_2)$,

$$\frac{\|\delta_1\|}{p} = \frac{\Delta}{2}.$$

Proof. We can write:

$$\Delta = \mathcal{RD}(\overline{X}_1, \overline{X}_2) = \frac{\|\overline{X}_1 - \overline{X}_2\|}{p} = \frac{\|\delta_1 - \delta_2\|}{p} = \frac{\|\delta_1\| + \|\delta_2\|}{p}.$$

The claim follows since $\|\delta_1\| = \|\delta_2\|$. ◁

▷ Claim 26. For $i \in [2]$ and $j \in [m] \cup 0$ let $\mathcal{E}_j$ be the event that $j = i_{1,b}(\overline{X}_1)$. With probability $\geq 999/1000$ over the choice of $\mathrm{A}_b\,\chi$, for all j we have:

$$\Pr[\mathcal{E}_j \;\mid\; \mathrm{A}_b\,\chi] \leq \frac{1}{16}.$$

Proof. Let $\Delta = \mathcal{RD}(\overline{X}_1, \overline{X}_2)$ and assume for simplicity that $\Delta \leq 1$ (we will latter see that it is easy to remove this assumption). By Claim 25, $\|\delta_1\| = \|\delta_2\| \leq \frac{p}{2}$ and thus $\|\chi\| \geq \frac{p}{2}$. For a constant c_3 (to be set later) sub-divide the range $\left[\|\chi\| - c_3 p^{1/2}, \|\chi\| + c_3 p^{1/2}\right]$ into $\frac{1}{\Delta^{1/2}}$ subranges of width:

$$2c_3 p^{1/2}\Delta^{1/2} = 2\sqrt{2}c_3\|\delta_1\|^{1/2},$$

where the equality follows from Claim 25. By Lemma 5 (applied with the constant c in the Lemma set to c_3) for any $i \in [m]$, $[A_b\chi](i)$ falls into each range with probability $\Theta(c_3 \cdot \Delta^{1/2})$. Thus, by a standard Chernoff bound, for $m = \frac{c_1 \log 1/\Delta}{\sqrt{\Delta}}$ for sufficiently large c_1, with probability $1999/2000$ over the choice of A_b, at least c_4 indices of $A_b\chi$ fall within each bucket where c_4 is a constant to be set later. Note that c_1 depends on c_3, c_4. Call the event that c_4 indices fall into each bucket $\mathcal{E}_{full-buckets}$. Additionally, as argued in Lemma 7, for sufficiently large m, the maximum value $[A_b\bar{X}_1](i_{1,b}(\bar{X}_1))$ below $p + p^{1/2}$ satisfies $[A_b\bar{X}_1](i_{1,b}(\bar{X}_1)) \geq p$ with probability at least $1999/2000$. Thus, with probability $1999/2000$ over the choice of A_b,

$$\Pr\left[[A_b\bar{X}_1](i_{1,b}(\bar{X}_1)) \geq p \middle| A_b\chi\right] \geq 1999/2000. \tag{6}$$

Let $\mathcal{E}_{good}$ be the event that both $\mathcal{E}_{full-buckets}$ and (6) hold. $\mathcal{E}_{good}$ holds with probability $\geq 999/1000$ over the choice of A_b. First note that conditioning on $\mathcal{E}_{good}$, $\Pr[\mathcal{E}_0 \big|\, \mathrm{A}_b\,\chi] \leq \frac{1}{2000}$, easily giving the claim for $j = 0$. We now consider $j \in [m]$. We consider any bucket,

$$R = \left\{j : [A_b\chi](j) \in \left[r, r + 2\sqrt{2}c_3\|\delta_1\|^{1/2}\right]\right\},$$

where r is be some integer multiple of $2\sqrt{2}c_3\|\delta_1\|^{1/2}$. Roughly, since each index in R has a very similar value in $A_b\chi$, each has nearly the same likelihood of being the largest entry in A_bX_1 below $\tau = p + 2p^{1/2}$. Since $\mathcal{E}_{full-buckets}$ occurs, there are at least c_4 of these indices

and thus if c_4 is large, none has very high probability of being the largest entry. Formally, we will show that, assuming $\mathcal{E}_{good}$ holds, for each $j \in R$,

$$\Pr[\mathcal{E}_j \mid \mathrm{A}_b\, \chi] \leq \frac{1}{16}. \tag{7}$$

Since this bound holds for all buckets in the range $[\|\chi\| - c_3 p^{1/2}, \|\chi\| + c_3 p^{1/2}]$, it will give the claim after arguing that no index with $A_b\chi$ falling outside this range is likely to have $\mathcal{E}(1, j)$ occur either.

Indices in Buckets. Each entry of $A_b\delta_1$ is identically distributed as an independent Chi-squared random variable with $\|\delta_1\|$ degrees of freedom. Additionally, $A_b\delta_1$ is independent of $A_b\chi$ since δ_1 and χ have disjoint supports. Consider $j \in R$ with $\Pr[[A_b\bar{X}_1](j) \geq \tau \big| A_b\chi] \geq 15/16$. In this case, since $\mathcal{E}(1, j)$ can only hold if $[A_b\bar{X}_1](j) \leq \tau$, (7) trivially holds.

Next consider $j \in R$ with $\Pr[[A_b\bar{X}_1](j) \geq \tau \big| A_b\chi] \leq 15/16$. By Lemma 6 there is some c with:

$$\Pr\left[[A_b\delta_1](j) \geq \|\delta_1\| + c\|\delta_1\|^{1/2}\right] = \frac{1}{64},$$

or equivalently since $[A_b\overline{X}_1](j) = [A_b\chi](j) + [A_b\delta_1](j)$:

$$\Pr\left[[A_b\overline{X}_1](j) \geq [A_b\chi](j) + \|\delta_1\| + c\|\delta_1\|^{1/2} \big| A_b\chi\right] = \frac{1}{64},$$

Setting $r_2 = \min\left([A_b\chi](j) + \|\delta_1\| + c\|\delta_1\|^{1/2}, \tau\right)$ we thus have that

$$\Pr[[A_b\overline{X}_1](j) \in [r_2, \tau] \big| A_b\chi] \leq \frac{1}{64}. \tag{8}$$

Additionally, by Lemma 5 there is some r_1 with $r_2 - r_1 = \Theta(\|\delta_1\|^{1/2})$ such that:

$$\Pr\left[[A_b\overline{X}_1](j) \in [r_1, r_2] \big| A_b\chi\right] = \frac{1}{32}.$$

Since for all $j' \in R$, $|[A_b\chi](j) - [A_b\chi](j')| \leq 2\sqrt{2}c_3\|\delta_1\|^{1/2} = O(\|\delta_1\|^{1/2})$ we have $r_2 = [A_b\chi](j') + O(\|\delta_1\|^{1/2})$ and thus again by Lemma 5, for all $j' \in R$:

$$\Pr\left[[A_b\overline{X}_1](j') \in [r_1, r_2] \big| A_b\chi\right] = \Omega(1).$$

If we set the constant c_4 large enough, since assuming $\mathcal{E}_{good}$, $|R| \geq c_4$ we have:

$$\Pr\left[\exists\, j' \in R \setminus j :\ [A_b\overline{X}_1](j') \in [r_1, r_2] \big| A_b\chi\right] \geq \frac{31}{32}.$$

If this event holds, we can only have $\mathcal{E}(1, j)$ occur if $[A_b\overline{X}_1](j)$ falls in $[r_2, \tau]$, which by (8) occurs with probability $\leq \frac{1}{64}$ conditioned on $A_b\chi$. Thus by a union bound we have:

$$\Pr[\mathcal{E}_j \mid \mathrm{A}_b\, \chi] \leq \frac{1}{16},$$

giving (7) in this case.

Indices Outside Buckets. We now consider indices not falling in any bucket: that is, j with $[A_b\chi](j) \leq \|\chi\| - c_3 p^{1/2}$ or $[A_b\chi](j) \geq \|\chi\| + c_3 p^{1/2}$. For the later, to have $\mathcal{E}_b(1,j)$ occur we must have $[A_b\overline{X}_1](j) \leq \tau = p + 2p^{1/2}$ and thus $[A_b\delta_1](j) \leq \|\delta_1\| - (c_3 - 2)p^{1/2} \leq \|\delta_1\| - (c_3 - 2)\sqrt{2}\|\delta_1\|$. By Lemma 6, this occurs with probability $< 1/16$ for all $\|\delta_1\|$ as long as we set c_3 large enough. Similarly, for j with $[A_b\chi](j) \leq \|\chi\| - c_3 p^{1/2}$, with probability $\geq 15/16$, we will have $[A_b\delta_1](j) \leq \|\delta_1\| + c_3 p^{1/2}$ and thus $[A_b\overline{X}_1](j) \leq p$. Since assuming $\mathcal{E}_{good}$, the maximum value of $A_b\overline{X}_1$ bounded by $\leq \tau$ is $\geq p$, if $[A_b\overline{X}_1](j) \leq p$, $\mathcal{E}_b(1,j)$ will not occur. Thus completes the argument in this case, giving that for all j with $[A_b\chi](j) \leq \|\chi\| - c_3 p^{1/2}$ or $[A_b\chi](j) \geq \|\chi\| + c_3 p^{1/2}$, $\Pr[\mathcal{E}_b(1,j) \mid \mathrm{A}_b\,\chi] \leq \frac{1}{16}$.

Removing Bound on Δ. Finally, we note that we can remove the assumption that $\Delta \leq 1$. If $\Delta \geq 1$ we can simply have χ encompass some of the non-shared entries in $\overline{X}_1$ until $\|\chi\| \geq \frac{p}{2}$ and $\|\delta_1\| \leq \frac{p}{2}$ as desired. The bound will go through as argued up to constants, since we will still have $\frac{\|\delta\|_1}{p} = \Theta(\Delta)$ as in Claim 25 (note that we always have $\Delta \leq 2$). ◁

We can now complete the proof of Lemma 8. We have:

$$\begin{aligned}\Pr[i_{1,b}(\overline{X}_1) = i_{1,b}(\overline{X}_2) \mid \mathrm{A}_b\,\chi] &= \sum_{j=0}^{m} \Pr\left[i_{1,b}(\overline{X}_1) = i_{1,b}(\overline{X}_2) = j \mid \mathrm{A}_b\,\chi\right] \\ &= \sum_{j=0}^{m} \Pr\left[i_{1,b}(\overline{X}_1) = j \mid \mathrm{A}_b\,\chi\right] \cdot \Pr\left[i_{1,b}(\overline{X}_2) = j | \,\mathrm{A}_b\,\chi\right] \end{aligned} \tag{9}$$

where the second line follows from the fact that $\mathrm{A}_b\,\overline{X}_1$ and $\mathrm{A}\,\overline{X}_2$ are independent conditioned on $\mathrm{A}_b\,\chi$ since δ_1, δ_2 are disjoint vectors. By Claim 26, with probability $\geq 999/1000$ over the choice of $\mathrm{A}_b\,\chi$ we can bound (9) by:

$$\Pr[i_{1,b}(\overline{X}_1) = i_{1,b}(\overline{X}_2) \mid \mathrm{A}_b\,\chi] \leq \sum_{j=0}^{m} \Pr\left[i_{1,b}(\overline{X}_2) = j \mid \mathrm{A}_b\,\chi\right] \cdot 1/16 = 1/16 \tag{10}$$

where the last line follows simply since $\sum_{j=0}^{m} \Pr\left[i_{1,b}(\overline{X}_2) = j \mid \mathrm{A}_b\,\chi\right] = 1$. Since (10) holds with probability $\geq 999/1000$ over the choice of $\mathrm{A}_b\,\chi$, overall $\Pr[i_{1,b}(\overline{X}_1) = i_{1,b}(\overline{X}_2)] \leq 1/16 + 1/1000$.

Applying Claim 7 and a union bound gives that $i_{1,b}(\overline{X}_1) \neq i_{1,b}(\overline{X}_2)$ and the gaps between the largest and second largest entries of $\mathrm{A}_b\,\overline{X}_1$ and $\mathrm{A}_b\,\overline{X}_2$ (bounded by τ) are both at least $\geq \frac{p^{1/2}}{c_2 \cdot m}$ (or there is at most one such entry), and $[\mathrm{A}_b\,\overline{X}_1](i_1(\overline{X}_1)), [\mathrm{A}_b\,\overline{X}_2](i_1(\overline{X}_2)) \geq p$ with probability $\geq 1 - (1/16 + 1/1000) - 2/100 = .9165$, giving the lemma.

A.3 Proof of Lemma 10

We now give the deferred proof of Lemma 10, which shows that two close inputs are likely to have the same intermediate neuron with the maximum potential $\leq \tau$ in each bucket. We restate the lemma below.

▶ **Lemma 10.** *Let $\overline{X}_1, \overline{X}_2 \in \{0,1\}^n$ be two vectors with $\mathcal{RD}(\overline{X}_1, \overline{X}_2) \leq \Delta/\alpha$. Consider our construction with bucket size $m = \frac{c_1 \log(1/\Delta)}{\sqrt{\Delta}}$. Then for sufficiently large constants c_1, c_2 and $\alpha = O(\log(1/\Delta)^4)$, for any $b \in [\ell]$, defining $i_{1,b}(\cdot)$ and $i_{2,b}(\cdot)$ as in Lemma 7, with probability ≥ 0.97:*

- $i_{1,b}(\overline{X}_1) = i_{1,b}(\overline{X}_2)$.
- *For both $j = 1, 2$: $i_{1,b}(\overline{X}_j) \neq 0$, $[A_b\overline{X}_j](i_{1,b}(\overline{X}_j)) \geq p$, and $[A_b\overline{X}_j](i_{1,b}(\overline{X}_j)) - [A_b\overline{X}_j](i_{2,b}(\overline{X}_j)) \geq \frac{p^{1/2}}{c_2 \cdot m}$ or $i_{2,b}(\overline{X}_j) = 0$.*

Proof. By Lemma 7, with probability $\geq 99/100$, for all $i \in [m] \setminus i_{1,b}(\overline{X}_1)$ with $[\mathrm{A}_b\, \overline{X}_1](i) \leq \tau$:

$$[\mathrm{A}_b\, \overline{X}_1](i_{1,b}(\overline{X}_1)) - [\mathrm{A}_b\, \overline{X}_1](i) = \Omega\left(\frac{p^{1/2}}{m}\right), \tag{11}$$

By a similar argument, with probability $\geq 99/100$, for all $i \in [m]$,

$$|\tau - (\mathrm{A}_b\, \overline{X}_1)_i| = \Omega\left(\frac{p^{1/2}}{m}\right). \tag{12}$$

Additionally, by standard sub-exponential concentration (as used in Lemma 6) with probability $\geq 99/100$, for both $j = 1, 2$ and all $i \in m$ we have $[\mathrm{A}_b\, \delta_j](i) \in \|\delta_i\| \pm O(\log m \cdot \|\delta_i\|^{1/2})$. Note that $\log m = O(\log(1/\Delta))$. Additionally, by Claim 25, since $\mathcal{RD}(\overline{X}_1, \overline{X}_2) \leq \Delta/\alpha$ for $\alpha = O(\log(1/\Delta)^4)$, we have for both $i = 1, 2$, $\frac{\|\delta_i\|}{p} \leq \frac{\Delta}{2\alpha} = O\left(\frac{\Delta}{\log(1/\Delta)^4}\right)$. This gives that

$$O(\log m \cdot \|\delta_1\|^{1/2}) = O\left(\frac{\Delta^{1/2} p^{1/2}}{\log(1/\Delta)}\right) = O\left(\frac{p^{1/2}}{m}\right).$$

So for both $j = 1, 2$ and all $i \in m$, $(\mathrm{A}_b\, \delta_j)_i \in \|\delta_i\| \pm O\left(\frac{p^{1/2}}{m}\right)$. So by (11) we have for all $i \neq i_{1,b}(\overline{X}_1)$ with $[\mathrm{A}_b\, \overline{X}_1](i) \leq \tau$:

$$[\mathrm{A}_b\, \overline{X}_2](i_{1,b}(\overline{X}_1)) - [\mathrm{A}_b\, \overline{X}_1](i) =$$
$$[\mathrm{A}_b\, \overline{X}_1](i_{1,b}(\overline{X}_1)) - [\mathrm{A}_b\, \delta_1](i_{1,b}(\overline{X}_1)) + [\mathrm{A}_b\, \delta_2](i_{1,b}(\overline{X}_1)) - [\mathrm{A}_b\, \overline{X}_1](i) = \Omega\left(\frac{p^{1/2}}{m}\right).$$

By (12) we also have,

$$[\mathrm{A}_b\, \overline{X}_2](i_{1,b}(\overline{X}_1)) = [\mathrm{A}_b\, \overline{X}_1](i_{1,b}(\overline{X}_1)) - [\mathrm{A}_b\, \delta_1)](i_{1,b}(\overline{X}_1)) + [\mathrm{A}_b\, \delta_2](i_{1,b}(\overline{X}_1)) \leq \tau.$$

and similarly, for all $i \neq i_{1,b}(\overline{X}_1)$ with $[\mathrm{A}_b\, \overline{X}_1](i) \geq \tau$:

$$[\mathrm{A}_b\, \overline{X}_2](i) \geq \tau.$$

That is, $i_{1,b}(\overline{X}_1)$ is the largest entry of $\mathrm{A}_b\, \overline{X}_2$ under τ, and thus $i_{1,b}(\overline{X}_1) = i_{1,b}(\overline{X}_2)$.

Applying Lemma 7 and a union bound gives the second claim with overall probability $1 - 1/100 - 1/100 - 1/100 = 97/100$. ◀

B Detailed Analysis of the Sparsification Step via WTA

Proof of Claim 13

Proof. Let $i = \arg\max_{j:\overline{Y}(j) \leq \tau} \overline{Y}(j)$. For every neuron $j \in \{1, \ldots, m\}$ in the input vector $\overline{Y}$, let $\overline{R}(j)$ be the random variable that counts the number of rounds in which j fires in a window of $T = \Theta(m^2 \log m)$ rounds. By the construction described above in which all j with $\overline{Y}(j) \geq \tau$ are inhibited with very strong weight, $\overline{R}(j) = 0$ w.h.p. for all such j. Thus we focus on j with $\overline{Y}(j) \leq \tau$. We show that if $\overline{Y}(i) - \overline{Y}(j) = \Omega\left(\frac{p^{1/2}}{m}\right)$ for $j \neq i$, then $\overline{R}(i) \gg \overline{R}(j)$ with probability at least $1 - \Theta(1/m)$.

First, let $\overline{P}$ be the vector of firing probabilities of each intermediate neuron induced by the potentials in $\overline{Y}$ (ignoring the entries that have been zero'd out since $\overline{Y}(j) \geq \tau$). By (1) we have $\overline{P}(i) = \frac{1}{1+e^{-\overline{Y}(i)}}$. Letting $s(x) = 1/(1 + e^{-x})$, we have $s'(x) \in \left[\frac{1}{2}, \frac{3}{4}\right]$ for $x \in [0, 1]$

and can see that if $\overline{Y}(i) - \overline{Y}(j) = \Omega(1/m)$, then also $s(\overline{Y}(i)) - s(\overline{Y}(j)) = \Omega(1/m)$. That is, a gap of $\Omega(1/m)$ between $\overline{Y}(i)$ and $\overline{Y}(j)$ translates to a gap of $\Omega(1/m)$ between the firing probabilities $\overline{P}(i)$ and $\overline{P}(j)$. To ensure that $\overline{Y}(i), \overline{Y}(j)$ are in $[0,1]$ we can simply rescale the weights of the random connection matrix A by $\frac{1}{2p}$ and shift them by p by adding a bias of p to each intermediate neuron. By Corollary 9, before this shift and scaling, $\overline{Y}(i) \in [p, p + 2p^{1/2}]$, so afterwards, $\overline{Y}(i) \in [0,1]$. For all $j \neq i$, since by Corollary 9 we had $\overline{Y}(i) - \overline{Y}(j) = \Omega\left(\frac{p^{1/2}}{m}\right)$ we still have $\overline{Y}(i) - \overline{Y}(j) = \Omega\left(\frac{1}{m}\right)$ as required and thus $\overline{P}(i) - \overline{P}(j) = \Omega\left(\frac{1}{m}\right)$.

By Chernoff bound, with probability of at least $1 - c/m$,

$$\overline{R}(i) \geq T \cdot \overline{P}(i) - \sqrt{T \cdot \overline{P}(i) \cdot c \log m} \quad \text{and} \quad \overline{R}(j) \leq T \cdot \overline{P}(j) + \sqrt{T \cdot \overline{P}(j) \cdot c \log m)} \ .$$

Hence, with probability $1 - 2c/m$ we get that

$$\begin{aligned} \overline{R}(i) - \overline{R}(j) &\geq T \cdot (\overline{P}(i) - \overline{P}(j)) - \sqrt{T \cdot \overline{P}(i) \cdot c \log m)} - \sqrt{T \cdot \overline{P}(j) \cdot c \log m.} \\ &\geq T \cdot (\overline{P}(i) - \overline{P}(j)) - 2\sqrt{T \cdot \overline{P}(i) \cdot \log m} = \Omega(T/m) - O(\sqrt{T \cdot \log m}) \\ &= \Omega(T/m) \ , \end{aligned}$$

by taking $T = c' \cdot m^2 \log m$ for a sufficiently large constant c'.

Since the incoming weight of each neuron $y_{i,j}$ is $\overline{R}(i) - \overline{R}(j) = \omega(1)$, we get that $y_{i,j}$ fires with probability of $1 - \Theta(1/m)$. By doing a union bound over all $m - 1$ neurons, and taking large enough constants, we get that with probability at least $99/100$, all neurons $y_{i,j}$ fire for every $j \neq i$. Hence, z_i is the only firing neuron in $\overline{Z}$. ◀

Recall that in Step (1), every input vector $\overline{X}_i$ is projected into ℓ vectors $\overline{Y}_{i,b} = \mathrm{A}_b \cdot \overline{X}_i$ for every $b \in \{1, \ldots, m\}$. On each such vector $\overline{Y}_{i,b}$ we apply the WTA circuit and get a vector $\overline{Z}_{i,b}$. Let $\overline{Z}_i = \overline{Z}_{i,1} \circ \overline{Z}_{i,2} \circ \ldots \circ \overline{Z}_{i,\ell}$ for $\ell = O(\log(t_m/\delta))$, where $\circ$ denotes vector concatenation.

We conclude this section by showing that the relative gap between input patterns $\overline{X}_i, \overline{X}_j$ is reflected in their output vectors of the WTA circuit. By combining Claim 13 with Cor. 9 and 11, we prove Lemma 12 which completes the correctness of Step (II).

Proof of Lemma 12

Proof. First observe that for every input $\overline{X}_i$, there are at most ℓ non-zero entries in $\overline{Z}_i$ since the threshold gates fire only if there is a sufficient gap in the firing rates. (I) For a fixed pair $\overline{X}_i, \overline{X}_j$ of far patterns, let $B_{i,j}$ be the set of all buckets b where $\arg\max_{r \in [m]: \overline{Y}_{i,b}(r) \leq \tau} \overline{Y}_{i,b}(r) \neq \arg\max_{r \in [m]: \overline{Y}_{j,b}(r) \leq \tau} \overline{Y}_{j,b}(r)$ and the gap between largest and second largest entries in both vectors $\overline{Y}_{i,b}$ and $\overline{Y}_{j,b}$ is $\Omega(p^{1/2}/m)$. By Cor. 9, with probability $1 - \delta$, for every pair of far patterns $\overline{X}_i, \overline{X}_j$, $|B_{i,j}| \geq 0.9 \cdot \ell$.

By Claim 13, if $\overline{Y}_{i,b}$ has a desired gap between the largest entry and other entries then, with probability $p = 99/100$, $\overline{Z}_{i,b}$ has exactly one winning entry corresponding to $\arg\max(\overline{Y}_{i,b})$. In expectation the vectors $\overline{Z}_{i,b}$ differ in $p \cdot |B_{i,j}|$ buckets. Thus by applying Chernoff bound overall k^2 pairs, in 0.9ℓ of the buckets, the WTA picks a distinct winner for the $\overline{X}_i$ and $\overline{X}_j$ patterns. Thus, $\mathrm{supp}(\overline{Z}_i) \setminus \mathrm{supp}(\overline{Z}_j) \geq 0.9\ell$.

(II) For a fixed pair $\overline{X}_i, \overline{X}_j$ of close patterns, let $B_{i,j}$ be the set of all buckets b where $\arg\max_{r \in [m]: \overline{Y}_{i,b}(r) \leq \tau} \overline{Y}_{i,b}(r) = \arg\max_{r \in [m]: \overline{Y}_{j,b}(r) \leq \tau} \overline{Y}_{j,b}(r)$ and the gap between largest and second largest entries in both vectors $\overline{Y}_{i,b}$ and $\overline{Y}_{j,b}$ is $\Omega(p^{1/2}/m)$. By Cor. 11 with probability $1 - \delta$, for every pair of close patterns $\overline{X}_i, \overline{X}_j$, $|B_{i,j}| \geq 0.91 \cdot \ell$. By applying Claim 13 and Chernoff bound overall k^2 pairs, in at least $0.9 \cdot \ell$ of the buckets, the selected winner is the same with probability of $1 - \delta$, implying that $\mathrm{supp}(\overline{Z}_i) \cap \mathrm{supp}(\overline{Z}_j) \geq 0.9 \cdot \ell$. ◀

Signed Tropical Convexity

Georg Loho
London School of Economics and Political Science, UK
g.loho@lse.ac.uk

László A. Végh
London School of Economics and Political Science, UK
l.vegh@lse.ac.uk

Abstract

We establish a new notion of tropical convexity for signed tropical numbers. We provide several equivalent descriptions involving balance relations and intersections of open halfspaces as well as the image of a union of polytopes over Puiseux series and hyperoperations. Along the way, we deduce a new Farkas' lemma and Fourier-Motzkin elimination without the non-negativity restriction on the variables. This leads to a Minkowski-Weyl theorem for polytopes over the signed tropical numbers.

2012 ACM Subject Classification Mathematics of computing → Combinatorial optimization

Keywords and phrases tropical convexity, signed tropical numbers, Farkas' lemma

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.24

Funding Supported by the European Research Council (ERC) Starting Grant ScaleOpt, No. 757481.

Acknowledgements We thank Daniel Dadush for insisting on balanced numbers and for inspiring discussions on the tropical Fourier-Motzkin elimination. We thank Xavier Allamigeon as well as Stéphane Gaubert for communicating their related work. We are grateful to Matthias Schymura for helping to get an intuition for the unintuitive line segments. Furthermore, we thank Mateusz Skomra and an anonymous referee for pointing out the NP-completeness stated in Theorem 4.21.

1 Introduction

Tropical convexity is an important notion with applications in several branches of mathematics. It arises from the usual definition of convexity by replacing $+$ with max and $\cdot$ with $+$. This notion has been studied for several years involving different approaches from extremal algebra [36], idempotent semirings [18], max-algebra [16], convex analysis [15], discrete geometry [20], matroid theory [22]. So far, it was mainly studied in $\mathbb{T}_{\max} = \mathbb{R} \cup \{-\infty\}$. Indeed, this is essentially a restriction to the tropical non-negative orthant, as $r \geq -\infty$ for all $r \in \mathbb{T}_{\max}$, where $-\infty$ is the tropical zero element. We remedy this restriction by introducing a notion of tropical convexity involving all orthants. We give our main points of motivation for our generalization.

Mean payoff games are equivalent to feasibility of a tropical linear inequalities

$$\bigoplus_{j \in J_i} a_{ij} + x_j \geq \bigoplus_{j \in [n] \setminus J_i} a_{ij} + x_j \quad \forall j \in [m] \ , \tag{1}$$

where $a_i, x \in \mathbb{T}_{\max}^n$, for $i \in [n]$, see [3]. This problem is in NP ∩ co-NP, but no polynomial-time algorithm is known [26]. Furthermore, the latter feasibility problem is intimately related to the feasibility problem for classical linear inequality systems [33, 6]. This connection has recently lead to a construction showing that certain interior point methods are not strongly polynomial [7]. The tropical linear feasibility problem is also a special scheduling problem [31] and it can be considered as a particular disjunctive programming problem [13].

In the context of mean payoff games, signs can be used to represent the two sides of the inequality (1). This system can be concisely written as $A \odot x \geq \mathbb{0}, x \geq \mathbb{0}$, where $\mathbb{0}$ represents the vector of $-\infty$'s, and $A \in \mathbb{T}_{\pm}^{m \times n}$ is a matrix with $A_{ij} = a_{ij}$ if $j \in J_i$ and $A_{ij} = \ominus a_{ij}$ if $j \notin J_i$. Here, $\ominus a_{ij}$ represents a tropically negative number (note that 5 and -5 are tropically positive, while $\ominus 5$ and $\ominus -5$ are tropically negative).

Our main motivation is to extend some of the basic concepts and tools from convex geometry and polyhedral combinatorics to signed tropical numbers. Significant work has been done already in this direction, however, some fundamentals are still missing: in particular, a satisfactory notion of convexity for signed tropical numbers has not been given prior to this work.

Such an investigation is motivated both by algorithmic questions as well as in the context of recent developments in tropical geometry. Solving mean payoff games in polynomial time is a major open question. Developing tropical versions of polyhedral combinatorics tools may lead to new approaches to tackle this problem. For example, there is a relatively recent class of polynomial-time algorithms for linear programming are naturally formulated as deciding if the origin is in the convex hull of a set of points, see, e.g., [17]. Our convexity notion provides an analogous formulation for the tropical linear feasibility problem in terms of the signed convex hull of the coefficient vectors.

Signed tropical numbers have been used in important recent results in tropical geometry, for example, in the tropicalization of the simplex method [6]. The study of real tropicalization of semialgebraic sets [27] follows a similar spirit. Another approach to extend from $\mathbb{T}_{\max}^d$ involving signs is to deduce the structure of a variety by "unfolding" it from the positive orthant into the other orthants, which is formalized by the patchworking introduced in [34] that has several applications in algebraic geometry.

Furthermore, separation theorems like Farkas' lemma for linear programming have their easiest formulation in terms of separation from the origin leading to powerful generalizations to oriented matroids, see [11]. Our approach allows to formulate an analogous theory for tropical linear programming. This gives new possibilities for studying tropical normal fans and tropical hyperplane arrangements.

Arithmetic operations can be naturally extended from $\mathbb{T}_{\max}$ to the set of signed tropical numbers $\mathbb{T}_{\pm}$. There is however a critical case of degeneracy, namely, adding a positive and a negative number of the same absolute value. In the context of the formulation (1), this corresponds to allowing the same variable x_j on both sides with the same a_{ij} term. Introducing *balanced numbers* are a standard way to carry out such additions: $5 \oplus (\ominus 5) = \bullet 5$, thus obtaining the *symmetrized semiring*. Unfortunately, it is not possible to order this structure, and hence, it is desirable to study the geometry restricted to the signed numbers $\mathbb{T}_{\pm}$.

1.1 Our Contributions

Tropical convexity has been well-studied for $\mathbb{T}_{\max}$ (see [6]). Balanced numbers constitute an important challenge in extending this notion to signed tropical numbers. Consider for example the points $(1, 1)$ and $(\ominus 1, \ominus 1)$ in $\mathbb{T}_{\pm}^2$. It is natural to include all points (a, a) and $(\ominus a, \ominus a)$ for $a \leq 1$ as well as $(-\infty, -\infty)$ as the convex combinations of these points. Our convexity notion will additionally include every point of the form (a, b), $(\ominus a, \ominus b)$, $(\ominus a, b)$, and $(a, \ominus b)$ for $a, b \leq 1$; thus, the convex combination of these two points in $\mathbb{T}_{\pm}^2$ will be a rectangle rather than a line segment. We obtain these combinations by "resolving" the balanced combination $(\bullet 1, \bullet 1)$ of these two points.

Our definition of signed tropical convexity (Definition 3.1) just arises from the usual definition of tropical convexity by replacing equality "=" with the balance relation "$\bowtie$" first introduced in [1]. We provide multiple arguments justifying why this is the right

definition. A standard way to introduce the tropical semiring is via Puiseux lifts. It turns out (Theorem 3.14) that this construction yields the tropicalization of the union of all possible lifts. This notion of signed tropical convexity also comes up naturally in the context of hyperoperations (Section 3.3). Instead of balanced numbers, hyperoperations have a multi-valued addition of signed numbers. Defining convexity for hyperoperations coincides with our notion. However, the advantage of using balanced numbers is to maintain a finite representation for computations.

Certain properties of the signed tropical convex hull are surprising and, compared to usual convexity, harder to deal with. For example, in contrast to the classical notion, there is no unique minimal generating set of a convex set: the convex hulls of the points $\{(1,1),(\ominus 1,\ominus 1)\}$ and of $\{(1,\ominus 1),(\ominus 1,1)\}$ coincide. The duality of signed tropical convex hulls and tropical linear inequality systems is reflected in the dual notions of non-negative kernel (22) and open tropical cones (23). We formalize a new version of Farkas' lemma (Theorem 4.6) for signed tropically convex sets. We deduce it in a geometric way from new versions of Fourier-Motzkin elimination for signed numbers (Theorem 4.14 and Corollary 4.15).

There is a curious difference between open and closed signed tropical halfspaces in our framework. Whereas open halfspaces are always convex, closed halfspaces are typically not. In fact, even signed tropical hyperplanes are convex in very special cases only (Example 3.8). Whereas we show that the convex hull of a set of points coincides with the intersection of all open halfspaces containing these points (Theorem 5.1), the analogous statement is not true for closed tropical halfspaces (Remark 5.3).

Nevertheless, we can derive a Minkowski-Weyl theorem (Theorem 5.4): for every finite set of points, their convex hull can be obtained as the intersection of finitely many closed halfspaces, and conversely, if the intersection of a finite set of closed halfspaces is convex (which is not always the case), it can be obtained as the convex hull of a finite set of points. The proof of the first direction is based on a version of Fourier-Motzkin elimination for non-strict inequalities. However, the elimination procedure may create balanced coefficients that have to be resolved by signed numbers. Such a transformation can be easily obtained for the case of strict inequalities (Theorem 4.12), but becomes rather challenging in the non-strict case. In fact, our proof (Proposition 5.7) only shows existence, but does not even yield a finite algorithm.

Finally, we relate our notion to the known concept of tropical convexity over $\mathbb{T}_{\geq 0}$. We show that the signed tropical convex hull can be obtained as the union of unsigned hulls in each orthant (Theorem 6.4).

1.2 Related Work

Our notion of signed tropical convexity heavily relies on the concept of the symmetrized tropical semiring $\mathbb{S}$, which goes back to [1], and was further developed in [4, 32], among others. Signed numbers arise in the context of tropical convexity in [6], however only as coefficients for an inequality system. The technically difficult aspects are the necessary properties of equality and order relations. While [4] also developed different notions replacing orders or equalities, they do not provide all necessary concepts to deal with the new notion of tropical convexity. The relations $\bowtie$, $\vDash$ and $>$ we use also appear in the context of hyperfields in [27], where images of semi-algebraic sets are studied. The duality of the tropical analog of polar cones in [23] can be considered as a predecessor of our duality in Section 4.1. Infeasibility certificates for linear inequality systems were deduced from the duality of mean payoff games in [25, 9]. A tropical version of Fourier-Motzkin elimination was established in [8]. The latter results rely on the (tropical) non-negativity of the variables and cannot be transferred directly to our setting, as we discuss also in Remark 4.18 and Remark 4.7. The tropicalizations of

polytopes [21] or more general semialgebraic sets [27] leads to the image of a single object. However, our construction naturally leads to the tropicalization of a union of polytopes arising as the convex hull of lifts of points. This is in some sense dual to the representation established in [27], where all satisfied equations and inequalities are needed to describe the tropicalization of a single object. Parallel to our work, similar structures for signed numbers are developed in [5, 2].

2 Signed Numbers and Orderings

We introduce the necessary terminology for our purposes. For a recent comprehensive introduction to signed numbers and the symmetrized semiring, see [4].

2.1 Signed Numbers

We define the signed tropical numbers $\mathbb{T}_\pm$ by glueing two copies of $(\mathbb{R} \cup \{-\infty\})$ at $-\infty$. One copy is declared the *non-negative tropical numbers* $\mathbb{T}_{\geq \mathbb{0}}$ (this is often denoted by $\mathbb{T}_{\max}$ in the literature), the other copy forms the *non-positive tropical numbers* $\mathbb{T}_{\leq \mathbb{0}}$. Most of the time, we denote $-\infty$ by $\mathbb{0}$ as it is the tropical zero element. The elements in $\mathbb{T}_{\leq \mathbb{0}} \setminus \{\mathbb{0}\}$ are marked by the symbol $\ominus$. The signed tropical numbers $\mathbb{T}_\pm$ have a natural norm $|\,.\,|$ which maps each element of $\mathbb{T}_{\geq \mathbb{0}}$ to itself and removes the sign of an element in $\mathbb{T}_{\leq \mathbb{0}}$. This gives rise to the order

$$x \leq y \qquad \Leftrightarrow \qquad \begin{cases} x \in \mathbb{T}_{\leq \mathbb{0}} \text{ and } y \in \mathbb{T}_{\geq \mathbb{0}} \\ x \leq y \text{ for } x, y \in \mathbb{T}_{\geq \mathbb{0}} \\ |x| \geq |y| \text{ for } x, y \in \mathbb{T}_{\leq \mathbb{0}} \end{cases} . \tag{2}$$

Furthermore, we obtain the strict order $x < y \Leftrightarrow x \leq y \wedge x \neq y$. The tropical signed space $\mathbb{T}_\pm^d$ is the union of 2^d orthants which are copies of $\mathbb{T}_{\geq \mathbb{0}}^d$ glued along their boundary.

2.2 Balanced Numbers

To develop the technical tools for dealing with signed numbers, we use the *symmetrized semiring* $\mathbb{S}$ which forms a semiring containing $\mathbb{T}_\pm$, introduced in [1]. This semiring is constructed with a third copy of $\mathbb{R} \cup \{\mathbb{0}\}$ by glueing again at $\mathbb{0}$. We denote the third copy, the *balanced numbers*, by $\mathbb{T}_\bullet$ and mark the elements by the symbol $\bullet$. Unfortunately, the symmetrized semiring $\mathbb{S}$ cannot be ordered. We extend the norm $|\,.\,|$ in such a way that it removes the $\bullet$ from an element in $\mathbb{T}_\bullet$ and leaves the corresponding element in $\mathbb{T}_{\geq \mathbb{0}}$. The complementary map tsgn from $\mathbb{S}$ to $\{\oplus, \ominus, \bullet, \mathbb{0}\}$ remembers only in which of the sets an elements lies: positive tropical numbers $\mathbb{T}_{>\mathbb{0}} = \mathbb{T}_{\geq \mathbb{0}} \setminus \{\mathbb{0}\}$, negative tropical numbers $\mathbb{T}_{<\mathbb{0}} = \mathbb{T}_{\leq \mathbb{0}} \setminus \{\mathbb{0}\}$, balanced non-zero tropical numbers $\mathbb{T}_\bullet \setminus \{\mathbb{0}\}$ or the tropical zero $\{\mathbb{0}\}$.

Next, we define the binary operations of the semiring. For $x, y \in \mathbb{S}$, we define the addition by

$$x \oplus y = \begin{cases} \operatorname{argmax}_{x,y}(|x|, |y|) & \text{if } |\chi| = 1 \\ \bullet \operatorname{argmax}_{x,y}(|x|, |y|) & \text{else} \end{cases} . \tag{3}$$

where $\chi = \{\operatorname{tsgn}(\xi) \mid \xi \in (\operatorname{argmax}(|x|, |y|))\}$. Note that we omit the sign for elements in $\mathbb{T}_{\geq \mathbb{0}}$. For the multiplication we set

$$x \odot y = (\operatorname{tsgn}(x) * \operatorname{tsgn}(y))\,(|x| + |y|) \ , \tag{4}$$

where the $*$-multiplication table is the usual multiplication of $\{-1, 1, 0\}$ for $\{\ominus, \oplus, \bullet\}$ with the additional specialty that multiplication with $\mathbb{0}$ yields $\mathbb{0}$.

The operations $\oplus$ and $\odot$ extend to vectors and matrices componentwise. Observe that the operations agree with the usual max-tropical operations on $\mathbb{T}_{\geq \mathbb{0}}$.

We can also consider $\ominus$ as a unary selfmap of the semiring; to this extent, we set

$$\ominus x = \begin{cases} \ominus x & \text{if } x \in \mathbb{T}_{>\mathbb{0}} \\ |x| & \text{if } x \in \mathbb{T}_{<\mathbb{0}} \\ x & \text{if } x \in \mathbb{T}_{\bullet} \end{cases} .$$

The map $\ominus \colon \mathbb{S} \to \mathbb{S}$ is a semiring homomorphism. In particular, this justifies to write $a \ominus b$ for $a \oplus (\ominus b)$.

Furthermore, the absolute value fulfills $|a \oplus b| = |a| \oplus |b|$ by definition of the addition.

▶ **Example 2.1.** Using the definitions, we see that -5 is positive, $\ominus 6$ and $\ominus -6$ are negative, $\bullet 3$ is balanced. Furthermore, the absolute value of -5 is $|-5| = -5$, of $\ominus 6$ is $|\ominus 6| = 6$, and $|\bullet 3| = 3$. Some simple sums are $3 \oplus (\ominus 3) = \bullet 3$, $-3 \oplus 5 = 5$, $-3 \oplus (\ominus 5) = \ominus 5$, $\bullet 2 \oplus 4 = 4$, $\bullet - 3 \oplus \ominus - 5 = \bullet - 3$. Finally some simple products are $\bullet 3 \odot 5 = \bullet 8$, $\ominus 4 \odot -6 = \ominus - 2$, $\ominus 1 \odot \ominus 1 = 2$, $\bullet 3 \odot \mathbb{0} = \mathbb{0}$, $\ominus 4 \odot \mathbb{0} = \mathbb{0}$.

2.3 Extending the Order

As already mentioned, the semiring $\mathbb{S}$ cannot be ordered in a consistent way with respect to its binary operations. However, we will equip it with some binary relations, which partly fulfill the tasks of an order. They occur under a different terminology in [27]; see 3.3.

2.3.1 Signed Order

Even if $\mathbb{S}$ cannot be ordered totally, we can extend the ordering from $\mathbb{T}_{\pm}$ partially by setting

$$x > y \quad \Leftrightarrow \quad x \ominus y \in \mathbb{T}_{>\mathbb{0}} \ . \tag{5}$$

This is equivalent to

$$x > y \quad \Leftrightarrow \quad \begin{cases} x > y & \text{for } x, y \in \mathbb{T}_{\pm}, \text{ see (2)} \\ x > |y| & \text{for } x \in \mathbb{T}_{\pm}, y \in \mathbb{T}_{\bullet} \\ \ominus |x| > y & \text{with } x \in \mathbb{T}_{\bullet}, y \in \mathbb{T}_{\pm} \end{cases} . \tag{6}$$

Note that there are pairs in $\mathbb{T}_{\pm} \times \mathbb{T}_{\bullet}$ and in $\mathbb{T}_{\bullet} \times \mathbb{T}_{\pm}$ which are not comparable. In particular, the signed numbers

$$\{x \in \mathbb{T}_{\pm} \mid x \not< a \text{ and } x \not> a\} \ ,$$

which are incomparable to $a \in \mathbb{T}_{\bullet}$ via "$<$", form the interval

$$\mathcal{U}(a) := [\ominus |a|, |a|] := \{x \in \mathbb{T}_{\pm} \mid \ominus |a| \leq x \leq |a|\} \ . \tag{7}$$

We also denote the set incomparable to a signed element $a \in \mathbb{T}_{\pm}$, which is only the singleton $\{a\}$, by $\mathcal{U}(a)$. We extend this to vectors by setting $\mathcal{U}(v) = \prod_{i \in [d]} \mathcal{U}(v_i)$. Note that also no pair in $\mathbb{T}_{\bullet} \times \mathbb{T}_{\bullet}$ is comparable.

The relation (6) gives rise to a non-strict relation

$$x \geq y \quad \Leftrightarrow \quad x > y \text{ or } x = y \ . \tag{8}$$

which turns out to be a partial order in Corollary 4.9.

Observe that the ordering is compatible with the reflection map, in the sense that

$$x \geq y \qquad \Leftrightarrow \qquad \ominus y \geq \ominus x \ . \tag{9}$$

A useful property of strict inequalities is that they can be added together.

▶ **Lemma 2.2.** *For $a, b, c, d \in \mathbb{S}$, we have the implication*

$$a < b \text{ and } c < d \qquad \Rightarrow \qquad a \oplus c < b \oplus d \ . \tag{10}$$

Proof. By definition, we first get $b \ominus a > \mathbb{0}$ and $d \ominus c > \mathbb{0}$. As addition is closed in $\mathbb{T}_{>\mathbb{0}}$, this yields $b \ominus a \oplus d \ominus c > \mathbb{0}$. The claim follows from (5). ◀

▶ Remark 2.3. In general, the strict and non-strict partial order "$<$" and "$\leq$" on $\mathbb{S}$ is not compatible with addition. The inequality $3 < 4$ does not imply $3 \oplus 5 < 4 \oplus 5$, and $3 \leq 4$ does not imply $\ominus 4 = 3 \ominus 4 \leq 4 \ominus 4 = \bullet 4$. This is the main motivation for introducing the relation "$\vDash$" below, which is not an ordering (as it lacks transitivity) but it is compatible with the addition.

An advantage of strict inequalities is the validity of

$$a \oplus b > c \Leftrightarrow a > c \ominus b \ .$$

The analogous reformulation

$$a \oplus b \geq c \Leftrightarrow a \geq c \ominus b \ .$$

is wrong in general. For example, $2 \oplus 5 \geq 5$ but 2 is incomparable with $5 \ominus 5 = \bullet 5$. However, such reformulations hold for the relation "$\vDash$", which we show in Lemma 2.6(a).

2.3.2 Balanced Relations

The balance relation "Δ" was introduced in [1]; we will use the notation $\bowtie$ in this paper. We define

$$x \bowtie y \quad \Leftrightarrow \quad x \ominus y \in \mathbb{T}_\bullet \ .$$

The following characterizations are immediate from the definitions. For more properties of $\bowtie$, we refer to [1, §IV].

▶ **Lemma 2.4.** *Let $a, b \in \mathbb{S}$.*
(a) *$a \bowtie b$ is equivalent to $(a \in \mathbb{T}_\bullet, |a| \geq |b|) \vee (b \in \mathbb{T}_\bullet, |b| \geq |a|) \vee (a = b)$.*
(b) *If $b \in \mathbb{T}_\pm$, then $a \bowtie b$ is equivalent to $b \in \mathcal{U}(a)$.*

▶ Remark 2.5. Note that $\bowtie$ is *not* an equivalence relation, as the example

$$1 \bowtie \bullet 6, \quad \bullet 6 \bowtie 3, \quad \text{but } 1 \not\bowtie 3$$

shows.

We introduce the binary relation

$$x \vDash y \qquad \Leftrightarrow \quad x > y \text{ or } x \bowtie y \quad \Leftrightarrow \qquad x \ominus y \in \mathbb{T}_{\geq \mathbb{0}} \cup \mathbb{T}_\bullet \ . \tag{11}$$

Note that $a \bowtie b$ is equivalent to $(a \vDash b) \vee (a \dashv b)$. Remark 2.5 shows that $\vDash$ is *not* a partial order.

Recall from Remark 2.3 that bringing terms to the other side of a non-strict inequality with "$\geq$" is not valid in general. The next lemma shows, among other simple properties, that "$\vDash$" is compatible with the semiring operations; the proof is deferred to the Appendix.

▶ **Lemma 2.6.** *Let* $a, b, c, d \in \mathbb{S}$.
(a) $a \oplus c \vDash b \Leftrightarrow a \vDash b \ominus c$
(b) $a \vDash b \quad \wedge \quad c \vDash d \quad \Rightarrow \quad a \oplus c \vDash b \oplus d$.
(c) *If* $c \in \mathbb{T}_{\pm}$, *then* $b \vDash c$ *and* $c \vDash a$ *imply* $b \vDash a$.
(d) $a \vDash b$ *implies* $c \odot a \oplus d \vDash c \odot b \oplus d$ *for* $c \in \mathbb{T}_{\geq \mathbb{0}}$ *and* $c \odot b \oplus d \vDash c \odot a \oplus d$ *for* $c \in \mathbb{T}_{\leq \mathbb{0}}$.

3 Tropical Convexity of Signed Numbers

3.1 Signed Tropical Convex Combinations

Let us recall the notation that for a matrix $A \in \mathbb{T}_{\geq \mathbb{0}}^{d \times n}$, and a vector $x \in \mathbb{T}_{\geq \mathbb{0}}^{n}$, we denote by $A \odot x \in \mathbb{T}_{\geq \mathbb{0}}^{n}$ the tropical matrix product. The *tropical convex hull* $\operatorname{tconv}(A)$ of the columns of a matrix $A \in \mathbb{T}_{\geq \mathbb{0}}^{d \times n}$, studied in [15, 18, 20], is defined as

$$\operatorname{tconv}(A) = \left\{ A \odot x \;\middle|\; x \in \mathbb{T}_{\geq \mathbb{0}}^{n}, \bigoplus_{j \in [n]} x_j = 0 \right\} \subseteq \mathbb{T}_{\geq \mathbb{0}}^{d} \ . \tag{12}$$

In this definition it is essential that all columns of A lie in the non-negative orthant $\mathbb{T}_{\geq \mathbb{0}}^{d}$. For general matrices in $\mathbb{T}_{\pm}^{d \times n}$, the product $A \odot x$ may contain balanced entries. We now extend the notion of the tropical convex hull to $\mathbb{T}_{\pm}^{d}$. Note that we switch freely between a matrix and its set of columns.

▶ **Definition 3.1** (Inner hull). *The* (signed) tropical convex hull *of the columns of the matrix* $A \in \mathbb{T}_{\pm}^{d \times n}$ *is defined as*

$$\operatorname{tconv}(A) = \left\{ z \in \mathbb{T}_{\pm}^{d} \;\middle|\; z \bowtie A \odot x, x \in \mathbb{T}_{\geq \mathbb{0}}^{n}, \bigoplus_{j \in [n]} x_j = 0 \right\} \subseteq \mathbb{T}_{\pm}^{d} \ . \tag{13}$$

Such a set is a (signed) tropical polytope. *The tropical convex hull of an arbitrary set* $M \subseteq \mathbb{T}_{\pm}^{d}$ *is the union*

$$\operatorname{tconv}(M) = \bigcup_{V \subseteq M, V \text{ finite}} \operatorname{tconv}(V) \ .$$

A subset $M \subseteq \mathbb{T}_{\pm}^{d}$ *is* tropically convex *if* $M = \operatorname{tconv}(M)$.
This hull construction generalizes (12) because if $A \in \mathbb{T}_{\geq \mathbb{0}}^{d \times n}$ then $A \odot x \in \mathbb{T}_{\pm}^{d}$. In this case, Lemma 2.4(a) implies that $z \bowtie A \odot x$ holds only for $z = A \odot x$.

Using Lemma 2.4(b), we can write (13) equivalently as

$$\operatorname{tconv}(A) = \bigcup \left\{ \mathcal{U}(A \odot x) \;\middle|\; x \in \mathbb{T}_{\geq \mathbb{0}}^{n}, \bigoplus_{j \in [n]} x_j = 0 \right\} \subseteq \mathbb{T}_{\pm}^{d} \ . \tag{14}$$

▶ **Example 3.2.** The combinations of the tropical convex hull depicted in Figure 1, where balanced numbers occur, can be calculated via

$$(-3) \odot \begin{pmatrix} 3 \\ 3 \end{pmatrix} \oplus \begin{pmatrix} \ominus 1 \\ \ominus 0 \end{pmatrix} = \begin{pmatrix} \ominus 1 \\ \bullet 0 \end{pmatrix} , \qquad (-2) \odot \begin{pmatrix} 3 \\ 3 \end{pmatrix} \oplus \begin{pmatrix} \ominus 1 \\ \ominus 0 \end{pmatrix} = \begin{pmatrix} \bullet 1 \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} 3 \\ 3 \end{pmatrix} \oplus (-1) \odot \begin{pmatrix} \ominus 4 \\ \ominus 2 \end{pmatrix} = \begin{pmatrix} \bullet 3 \\ 3 \end{pmatrix} , \qquad (-1) \odot \begin{pmatrix} 3 \\ 3 \end{pmatrix} \oplus \begin{pmatrix} \ominus 4 \\ \ominus 2 \end{pmatrix} = \begin{pmatrix} \ominus 4 \\ \bullet 2 \end{pmatrix} \ .$$

A more precise way, how these points can be used to determine the signed tropical convex hull, via the tropical convex hull of the intersection with each orthant is given in Theorem 6.4.

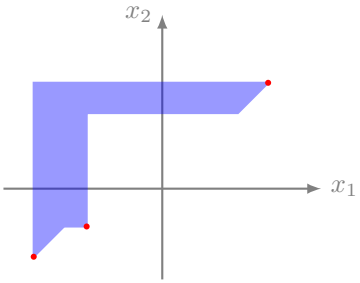


Figure 1 The signed tropically convex hull of $\{(3,3),(\ominus 1,\ominus 0),(\ominus 4,\ominus 2)\}$. We omit labels for the axes as the origin is $(-\infty,-\infty)$ and therefore infinitely far away.

▶ Remark 3.3. There is no unique minimal generating set in the usual sense as one can see from $\operatorname{tconv}((0,0),(\ominus 0,\ominus 0)) = \operatorname{tconv}((0,\ominus 0),(\ominus 0,0))$.

We now derive some elementary properties of this convexity notion. The following are immediate from the definition, as (14) is just a componentwise construction.

▶ **Proposition 3.4.**
(a) *The intersection of tropically convex sets is tropically convex.*
(b) *The coordinate projection of tropically convex sets is tropically convex.*

Next, we show that convexity follows already by showing the containment of line segments (Proposition 3.6), and that tconv(.) is a closure operator, i.e., the convex hull of a set is a tropically convex set (Proposition 3.7). The following technical lemma will be needed for these proofs. The next three proofs are deferred to the Appendix.

▶ **Lemma 3.5.**
(a) *Let $a \in \mathbb{S}$, $b \in \mathbb{T}_{\pm}$, and $z \in \mathcal{U}(a \oplus b)$. Then there exists an $a' \in \mathcal{U}(a)$ such that $z \in \mathcal{U}(a' \oplus b)$.*
(b) *If $a \in \mathcal{U}(x)$, $b \in \mathcal{U}(y)$, and $c \in \mathbb{T}_{\pm}$, then $\mathcal{U}(c \odot a \oplus b) \subseteq \mathcal{U}(c \odot x \oplus y)$.*

▶ **Proposition 3.6.** *An arbitrary subset $M \subseteq \mathbb{T}_{\pm}^d$ is tropically convex if and only if the tropical convex hull $\operatorname{tconv}(\{p,q\})$ is contained in M for all $p,q \in M$.*

▶ **Proposition 3.7.** *For any matrix $A \in \mathbb{T}_{\pm}^{d\times n}$, the convex hull $\operatorname{tconv}(A)$ is tropically convex. Consequently, $\operatorname{tconv}(\operatorname{tconv}(A)) = \operatorname{tconv}(A)$.*

▶ **Example 3.8.** A *(signed) tropical hyperplane* is of the form

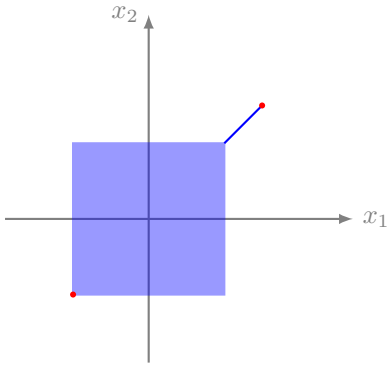
$$\operatorname{Hyp}(a) = \left\{ x \in \mathbb{T}_{\pm}^d \;\middle|\; a \odot x \in \mathbb{T}_{\bullet}^d \right\} \quad .$$

It is easy to see that this set is tropically convex if $|\operatorname{supp}(a)| = 1$, where $\operatorname{supp}(a) = \{i \in [d] \mid a_i \neq \mathbb{0}\}$. Therefore, using Proposition 3.4(a), we see that for a subset $I \subseteq [d]$ and a point $b \in \mathbb{T}_{\pm}^d$, the set

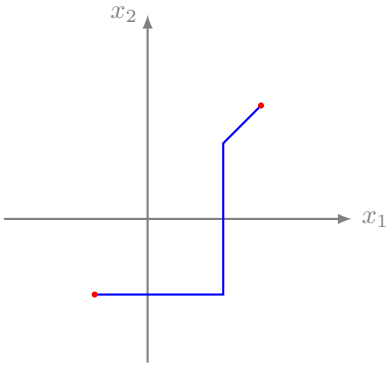
$$\left\{ z \in \mathbb{T}_{\pm}^d \;\middle|\; z_i = b_i \text{ for all } i \in I \right\}$$

is tropically convex.

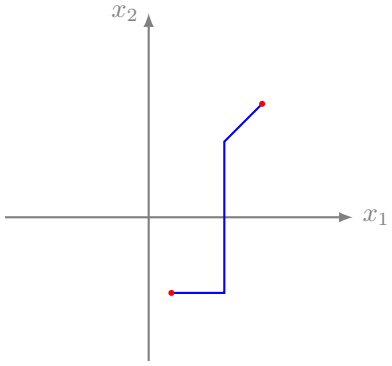
On the other hand, if $|\operatorname{supp}(a)| > 1$, then $\operatorname{Hyp}(a)$ is *not* tropically convex. To see this, let us assume that $\operatorname{supp}(a) \supseteq \{1,2\}$. Then $p = (\ominus a_2, a_1, \mathbb{0}, \ldots, \mathbb{0}), q = (a_2, \ominus a_1, \mathbb{0}, \ldots, \mathbb{0}) \in \operatorname{Hyp}(a)$. Because of $p \oplus q = (\bullet a_2, \bullet a_1, \mathbb{0}, \ldots, \mathbb{0})$, the point $(a_2, a_1, \mathbb{0}, \ldots, \mathbb{0})$ is contained in $\operatorname{tconv}(p,q)$. However, it is not an element of $\operatorname{Hyp}(a)$.



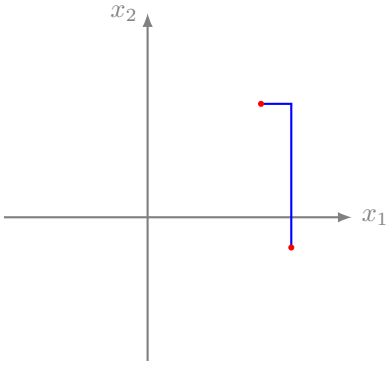
(a) $\mathrm{tconv}((0,0),(\ominus-2,\ominus-2))$.



(b) $\mathrm{tconv}((0,0),(\ominus-3,\ominus-2))$.



(c) $\mathrm{tconv}((0,0),(-3,\ominus-1))$.



(d) $\mathrm{tconv}((0,0),(2,\ominus-1))$.

Figure 2 Several tropical line segments in the plane.

▶ **Example 3.9.** For a vector $(a_0,a_1,\dots,a_d)\in\mathbb{T}_\pm^{d+1}$ we define the *open signed (affine) tropical halfspace*

$$\mathcal{H}^+(a)=\left\{x\in\mathbb{T}_\pm^d \;\middle|\; a\odot\begin{pmatrix}0\\x\end{pmatrix}>\mathbb{O}\right\}\ . \tag{15}$$

An open signed tropical halfspace is tropically convex. Let $c\in\mathbb{T}_\pm^d, c_0\in\mathbb{T}_\pm$, $p,q\in\mathbb{T}_\pm^d$ and $\lambda,\mu\in\mathbb{T}_{\leq\mathbb{O}}$ with $\lambda\oplus\mu=0$. For p and q contained in the halfspace, we have $c\odot p\oplus c_0>\mathbb{O}$ and $c\odot q\oplus c_0>\mathbb{O}$, and by Lemma 2.2,

$$c\odot(\lambda\odot p\oplus\mu\odot q)\oplus c_0=\lambda\odot(c\odot p\oplus c_0)\oplus\mu\odot(c\odot q\oplus c_0)>\mathbb{O}\ . \tag{16}$$

If $\lambda\odot p\oplus\mu\odot q$ has a balanced component $b\in\mathbb{T}_\bullet$, then the value of $c\odot(\lambda\odot p\oplus\mu\odot q)\oplus c_0$ cannot depend on this component as it is positive. Hence, we can replace that component by an element in $\mathcal{U}(b)$ and preserve the inequality (16).

▶ Remark 3.10. Let $\mathbb{P}\subset\mathbb{T}_\pm^{d\times d}$ be the set of permutation matrices with 0 as one and $\mathbb{O}$ as zero, and let $\mathbb{D}\subset\mathbb{T}_\pm^{d\times d}$ be the set of matrices with diagonal entries from $\mathbb{T}_\pm$ and $\mathbb{O}$ else. Their union generates the multiplicative *group of signed tropical transformations* $\mathbb{ST}$. This group is the natural group of transformations which leaves the combinatorial structure of a subset of $\mathbb{T}_\pm^d$ unchanged. It provides a useful tool to simplify technical constructions.

▶ **Example 3.11.** We want to describe the line segment $\mathrm{tconv}(p,q)$ for two points $p,q\in\mathbb{T}_\pm^d$. By suitable scaling with elements from $\mathbb{ST}$, we can assume that $p=(0,\dots,0)$, and that the entries of q are ordered by increasing absolute value.

Analogous to the description in [20], one obtains a piecewise-linear structure where the breakpoints are determined by the absolute values of the components of q. As an additional phenomenon, the line segments flip to another orthant at each tropically negative entry of q. If the sign changes in ℓ coordinates at once, the line segment has dimension ℓ. We visualize several examples for the two-dimensional case in Figure 2.

▶ **Definition 3.12** (Conic hull). *The* signed tropical conic hull *of the columns of A is*

$$\operatorname{tcone}(A) = \bigcup_{\lambda \in \mathbb{T}^n_{\geq \mathbb{0}}} \mathcal{U}(A \odot \lambda) \ . \tag{17}$$

The definition together with Proposition 3.7 yields the following.

▶ **Corollary 3.13.** *The conic hull of a subset of $\mathbb{T}^d_\pm$ is tropically convex.*

3.2 Image of Puiseux Lifts

The aim of this section is to relate our concept of convexity over $\mathbb{T}_\pm$ to convexity over $\mathbb{R}$. To achieve this, we move to another ordered field, the field of real Puiseux series $\mathbb{K} = \mathbb{R}\{\{t\}\}$. This has proven to be a helpful concept in the study of tropical numbers with signs, see [35, 6, 27]. It is formed by formal Laurent series with exponents in $\mathbb{R}$ and coefficients in $\mathbb{R}$. The exponent sequence is strictly decreasing and it has no accumulation point. This ordered field is equipped with a non-archimedean valuation val which maps all non-zero elements to their leading exponent and zero to $\mathbb{0} = -\infty$. Additionally, the map $\operatorname{sgn}\colon \mathbb{K} \to \{\ominus, \mathbb{0}, \oplus\}$ yields the sign of an element. This gives rise to the *signed valuation* $\operatorname{sval}\colon \mathbb{K} \to \mathbb{T}_\pm$ which maps an element $k \in \mathbb{K}$ to $\operatorname{sgn}(k)\operatorname{val}(k)$. It is enough to think of Puiseux series as polynomials in t with arbitrary exponents and coefficients in $\mathbb{R}$.

The *tropicalization* of structured sets over $\mathbb{K}$, i.e., the study of the image of a subset of $\mathbb{K}^d$ is a technique which is widely used in tropical geometry. We introduced a concept purely on the tropical side. We will see in Theorem 3.14, that signed tropically convex sets are not the image of the valuation of a single convex hull but of a whole union, ranging over the fibers of tropical points.

In some sense, this is complementary to the main result in [27]. While they consider semialgebraic sets over $\mathbb{K}$ in general, polytopes, i.e., the convex hull of finitely many points in $\mathbb{K}^d$, can be considered as a special case. They show that one has to tropicalize all semialgebraic relations fulfilled by a set to describe its image under the signed valuation map.

Recall that for our concept of tropical convexity over $\mathbb{T}_\pm$ the image of a single polytope under the signed valuation may not be tropically convex as the Example B.1 shows. It is subject to further work to study the special case of polytopes (as semialgebraic sets) from [27] and to see which properties such a notion of signed tropical polytopes could provide.

Note that the next statement is valid for more general fields with a non-trivial non-archimedean valuation val which is surjective onto $\mathbb{T}_{\geq \mathbb{0}}$. The proof is given in the Appendix.

▶ **Theorem 3.14.** *The signed hull* $\operatorname{tconv}(A)$ *is the union of the signed valuations for all possible lifts*

$$\operatorname{tconv} A = \bigcup_{\operatorname{sval}(\mathbf{A})=A} \operatorname{sval}(\operatorname{conv}(\mathbf{A})) \ .$$

▶ Remark 3.15. Theorem 3.14 generalizes [21, Proposition 2.1], since val is a semiring homomorphism from $\mathbb{K}_{\geq 0}$ to $\mathbb{T}_{\max} = \mathbb{T}_{\geq \mathbb{0}}$.

▶ **Corollary 3.16.** *The tropical convex hull is the union of the convex hulls of the lifts, i.e.,*

$$\operatorname{tconv}(A) = \operatorname{sval}(\operatorname{conv}(\operatorname{sval}^{-1}(A))) \ .$$

3.3 Convex Hull from Hyperoperation

As we shall see in Proposition 3.17, our signed tropical convex hull from Definition 3.1 is a natural generalization of the classical convex hull as image of a simplex to hyperoperations. In recent years, hyperfields found their way into matroid theory due to the work [12] building on [30] and [35]. While hyperfields have been used in tropical geometry [27] from an algebraic point of view, they were not used to describe intrinsically defined geometric objects before.

We introduce the necessary notions for hyperfields to define a signed convex hull and compare our binary operations with hyperfield operations (20). Let us briefly introduce the *real plus-tropical hyperfield* $\mathbb{H}$, see [35]. It has the multiplicative group $(\mathbb{T}_\pm, \odot)$ and its additive hyperoperation on $\mathbb{T}_\pm$ is given by

$$x \boxplus y = \begin{cases} \operatorname{argmax}_{x,y}(|x|,|y|) & \text{if } \chi \subseteq \{+,\mathbb{0}\} \text{ or } \chi = \{-\} \\ [\ominus|x|, |x|] & \text{else} \quad . \end{cases} \tag{18}$$

We see that the latter addition for non-balanced numbers $x, y \in \mathbb{T}_\pm$ differs from the Definition in (3) in that it has a multi-valued result in the powerset of $\mathbb{T}_\pm$. A balanced outcome $z \in \mathbb{T}_\bullet$ is replaced with the interval $\mathcal{U}(z) = [-|z|, |z|]$. One can extend the operations again componentwise and use the symbol $\boxdot$ for the product of two matrices or vectors. In particular, the operation $\boxdot$ agrees with $\odot$ on $\mathbb{T}_\pm$. The addition is set-valued in $\mathbb{H}$ if and only if it would be balanced in $\mathbb{S}$. It agrees with $\oplus$ on $\mathbb{T}_{\geq \mathbb{0}}$.

We recall the order relations used in [27] for the multiplicative real tropical hyperfield. Note that they use the multiplication $\odot =$"$\cdot$" instead of our approach with "+".

A polynomial over the real tropical hyperfield is a formal expression

$$F(x) = \boxplus_{d_1,\dots,d_n \in \mathbb{Z}} c_{d_1,\dots,d_n} x_1^{d_1} \cdots x_n^{d_n}$$

which can be evaluated at an element $\zeta \in \mathbb{H}^n$. This yields a subset

$$F(\zeta) = \boxplus_{d_1,\dots,d_n \in \mathbb{Z}} c_{d_1,\dots,d_n} \zeta_1^{d_1} \cdots \zeta_n^{d_n} \subseteq \mathbb{H}^n \quad . \tag{19}$$

Note that we mainly deal with linear polynomials, where the exponent vector $(d_1, \dots, d_n) \in \mathbb{Z}^d$ is just a unit vector.

One can define the sets

$$\begin{aligned} \{F = 0\} &:= \{(x_1,\dots,x_n) \in \mathbb{T}_\pm^n \colon \mathbb{0} \in F(x_1,\dots,x_n)\} \\ \{F \geq 0\} &:= \{(x_1,\dots,x_n) \in \mathbb{T}_\pm^n \colon F(x_1,\dots,x_n) \cap \mathbb{T}_{\geq \mathbb{0}} \neq \emptyset\} \\ \{F > 0\} &:= \{(x_1,\dots,x_n) \in \mathbb{T}_\pm^n \colon F(x_1,\dots,x_n) \in \mathbb{T}_{>\mathbb{0}}\} \quad . \end{aligned} \tag{20}$$

Observe that $F = 0$ is indeed equivalent to $F \geq 0 \wedge -F \leq 0$ due to the structure of the set (19). Translating (20) to the symmetrized semiring $\mathbb{S}$ yields the relations "$\bowtie$", "$\vDash$" and "$>$".

To motivate the next construction, we consider a tropical polytope generated by $V \in \mathbb{T}_{\geq \mathbb{0}}^{d \times k}$ as the image of the tropical standard simplex in the sense that

$$\operatorname{tconv}(V) = \{V \odot \lambda \mid \bigoplus_{\ell \in [k]} \lambda_\ell = 0\} \quad .$$

For a matrix $A \in \mathbb{T}_\pm^{d \times n}$, we define the *balanced image* of the tropical standard simplex $\Delta_n = \{\lambda \mid \bigoplus_{\ell \in [n]} \lambda_\ell = 0\}$ by

$$A \odot \Delta_n := \left\{ A \odot x \;\middle|\; \bigoplus_{j \in [n]} x_j = 0, x \geq \mathbb{0} \right\} \subset \mathbb{S}^d \quad . \tag{21}$$

With this notion, one can write $\operatorname{tconv}(A) = \bigcup_{z \in A \odot \Delta_d} \mathcal{U}(z)$.

By using the hyperoperations in $\mathbb{H}$, we can naturally consider the image of the tropical standard simplex $\Delta_k = \{\lambda \mid \bigoplus_{\ell \in [k]} \lambda_\ell = 0\}$ with respect to matrix multiplication by $V \in \mathbb{T}_\pm^{d \times k}$ as a subset of $\mathbb{T}_\pm^d$.

▶ **Proposition 3.17.**

$$V \boxdot \Delta_d = \operatorname{tconv}(V) \ .$$

Proof. This follows directly by the definition of the set-valued addition in (18) from (13) with $\mathcal{U}(z) = [\ominus|z|, |z|]$ for $z \in \mathbb{T}_\bullet$. ◀

In Appendix B, we compare our convexity notion to $\mathbb{B}$-convexity studied in the literature [15], as well as a cancellative sum construction used in [14].

4 Farkas' Lemma and Fourier-Motzkin Elimination

4.1 Convexity and Tropical Linear Feasibility

For a matrix $A = (a_{ij}) \in \mathbb{T}_\pm^{d \times n}$, we define the *non-negative kernel*

$$\ker_+(A) = \left\{ x \in \mathbb{T}^n_{\geq \mathbb{O}} \setminus \{\mathbb{O}\} \;\middle|\; A \odot x \bowtie \mathbb{O} \right\} \tag{22}$$

This corresponds to the classical definition of a polyhedral cone in the form $Ax = 0, x \geq 0, x \neq 0$. We replace "$=$" by "$\bowtie$" and "$\geq$" by "$\vDash$". In terms of the non-negative kernel, we can express containment in the convex hull as follows.

▶ **Proposition 4.1.** *For $A \in \mathbb{T}_\pm^{d \times n}$ and $b \in \mathbb{T}_\pm^d$ we have*

$$b \in \operatorname{tconv}(A) \Leftrightarrow \mathit{ker}_+ \begin{pmatrix} A & \ominus b \\ 0 & \ominus 0 \end{pmatrix} \neq \emptyset \ .$$

▶ **Corollary 4.2.** *The origin $\mathbb{O}$ is in the convex hull $\operatorname{tconv}(A)$ if and only if the non-negative kernel $\mathit{ker}_+(A)$ is not empty.*

Proof. Setting $b = \mathbb{O}$ in Proposition 4.1 implies the equivalence with the definition from (22). ◀

We now define the *open tropical cone* as the dual to the non-negative kernel (22).

$$\operatorname{sep}_+(A) = \{y \in \mathbb{T}_\pm^d \mid y^\top \odot A > \mathbb{O}\} \ . \tag{23}$$

The name is motivated by the use of the elements of $\operatorname{sep}_+(A)$ as separators of the columns of A from the origin. Note that the condition "$> \mathbb{O}$", in particular, means that the product "$y^\top \odot A$" is comparable with $\mathbb{O}$ and, equivalently, is in $\mathbb{T}_{>\mathbb{O}}$.

We can also define $\ker_+(A)$ and $\operatorname{sep}_+(A)$ for $A \in \mathbb{S}^{d \times n}$. However, this does not provide a wider class of objects. This follows by replacing a balanced number by $\mathbb{O}$ in $\ker_+(A)$ and applying Theorem 4.12 for $\operatorname{sep}_+(A)$. We still extend the definition to these more general matrices, as it will lead to simplified arguments.

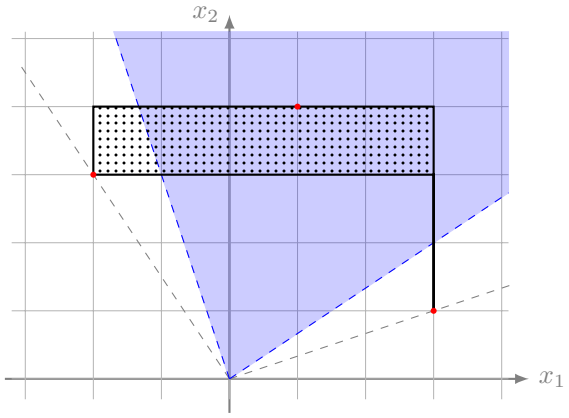


Figure 3 An open tropical cone visualized with the operations "$\oplus$" = "max" and "$\odot$" = "$\cdot$".

▶ **Example 4.3.** In the next example we work with the semiring $(\mathbb{R}_{\geq 0}, \max, \cdot)$, as it provides a more natural picture of the behaviour around the origin in the different orthants. Instead of the semiring defined on $\mathbb{R} \cup \{-\infty\}$ with operations "max" and "+", one can isomorphically use the semiring $\mathbb{R}_{>0} \cup \{0\}$ with "max" and "$\cdot$". The exponentiation map

$$\exp\colon \mathbb{R} \cup \{-\infty\} \to \mathbb{R}_{>0} \cup \{0\}$$

yields an isomorphism between these two semirings. One can further extend this mapping to $\mathbb{T}_{\pm}$ by setting

$$x \mapsto \qquad \operatorname{tsgn}(x) \exp(|x|) \ .$$

Note that the image of $\mathbb{S}$ involves a balanced version of positive numbers as it decomposes into the union $\mathbb{R}_{>0} \cup \mathbb{R}_{<0} \cup \bullet\mathbb{R}_{>0} \cup \{0\}$.

The dotted black shape in Figure 3 depicts the $(\max, \cdot)$-convex hull of the columns of the matrix $A = \begin{pmatrix} 3 & 1 & -2 \\ 1 & 4 & 3 \end{pmatrix}$. The combinations, where balanced numbers occur, are

$$\frac{1}{2} \cdot \begin{pmatrix} -2 \\ 3 \end{pmatrix} \oplus \begin{pmatrix} 1 \\ 4 \end{pmatrix} = \begin{pmatrix} \bullet 1 \\ 4 \end{pmatrix} \quad \text{and} \quad \frac{2}{3} \cdot \begin{pmatrix} 3 \\ 1 \end{pmatrix} \oplus \begin{pmatrix} -2 \\ 3 \end{pmatrix} = \begin{pmatrix} \bullet 2 \\ 3 \end{pmatrix} \ .$$

The blue shaded area is $\operatorname{sep}_+(A)$. The configuration visualizes Theorem 4.4, as $\operatorname{sep}_+(A)$ is not empty and the $(\max, \cdot)$-convex hull of A does not contain the origin.

Let us now show that weak duality holds between the non-negative kernel and the open tropical cone:

▶ **Theorem 4.4** (Weak duality). *For a matrix $A \in \mathbb{T}_{\pm}^{d \times n}$ at least one of the sets $\ker_+(A)$ and $\operatorname{sep}_+(A)$ is empty. Equivalently, for an arbitrary vector $y \in \mathbb{T}_{\pm}^{d}$ and a non-negative vector $x \in \mathbb{T}_{\geq \mathbb{0}}^{n} \setminus \{\mathbb{0}\}$ at most one of $A \odot x \in \mathbb{T}_{\bullet}^{d}$ and $y^{\top} \odot A > \mathbb{0}$ can hold.*

Proof. The claim follows from the fact that $\mathbb{T}_{\bullet}$ is a left- and right-ideal of $\mathbb{S}$, see [4, Definition 2.6]. This means that for $x \in \ker_+(A)$, the product $y^{\top} \odot A \odot x$ is in $\mathbb{T}_{\bullet}$ while $y^{\top} \odot A > \mathbb{0} \Rightarrow y^{\top} \odot A \odot x > \mathbb{0}$. ◀

▶ Remark 4.5. We give a direct proof of the former statement. We consider the product $y^\top \odot A \odot x$. Scaling the rows of A by arbitrary numbers in $\mathbb{T}_\pm$ does not change whether $x \in \ker_+(A)$, just as scaling the columns of A by a non-negative number in $\mathbb{T}_{\geq \mathbb{0}}$ does not change whether $y \in \operatorname{sep}_+(A)$. Hence, we can assume that x and y only have the entries 0 or $\mathbb{0}$. Let a_{pq} be the entry of A with maximal absolute value. For $x \in \ker_+(A)$, there is an index $r \in [n]$ such that $a_{pr} = \ominus a_{pq}$. We can assume that $a_{pq} > \mathbb{0} > a_{pr}$. From this we conclude that $y \notin \operatorname{sep}_+(A)$ since the rth column of $y^\top \odot A$ cannot be positive.

The key result of this section will be showing the appropriate version of Farkas' lemma. The proof will follow via Fourier-Motzkin elimination.

▶ **Theorem 4.6** (Farkas' lemma)**.** *For a matrix $A \in \mathbb{T}_\pm^{d\times n}$ exactly one of the sets $ker_+(A)$ and $sep_+(A)$ is nonempty.*

▶ Remark 4.7. Theorem 4.6 is similar to [25, Corollary 3.12]. Through a suitable replacement of the balanced coefficients and a careful analysis of the occuring signs, Theorem 4.6 may be deduced from [25]. Note however, that we allow for unconstrained variables in the definition of $\operatorname{sep}_+(A)$ which is not directly covered by [25, Corollary 3.12].

4.2 Technical Properties of the Order Relations

The next lemma is a version of transitivity and it is a preparation for the elimination of a variable in a system of inequalities in Section 4.3. The proofs of the next two propositions are given in the Appendix.

▶ **Proposition 4.8.** *Let $A, B \subset \mathbb{S}$ be two finite sets. There is an element $c \in \mathbb{S}$ with*

$$c \ominus a > \mathbb{0} \text{ and } b \ominus c > \mathbb{0} \text{ for all } a \in A, b \in B \tag{24}$$

if and only if

$$b \ominus a > \mathbb{0} \text{ for all } (a, b) \in A \times B \ . \tag{25}$$

Furthermore, the element c can chosen to be signed.

▶ **Corollary 4.9.** *The non-strict relation $x \geq y$ defined in* (8) *is a partial order.*

Proof. Reflexivity and antisymmetry follow directly from (8) and (6). Proposition 4.8 implies transitivity. ◀

▶ **Proposition 4.10.** *Let $A, B \subset \mathbb{S}$ be two finite sets. There is an element $c \in \mathbb{T}_\pm$ with*

$$c \ominus a \vDash \mathbb{0} \text{ and } b \ominus c \vDash \mathbb{0} \text{ for all } a \in A, b \in B \tag{26}$$

if and only if

$$b \ominus a \vDash \mathbb{0} \text{ for all } (a, b) \in A \times B \ . \tag{27}$$

To deal with geometric objects in $\mathbb{T}_\pm^d$, we will use balanced numbers because this allows for explicit calculations in the semiring $\mathbb{S}$. However, as we are only interested in the signed part of the sets. We provide a first tool to resolve balanced numbers in inequalities. While this is for strict inequalities, Proposition 5.7 provides a tool for the relation $\vDash$.

▶ **Lemma 4.11.** *For $a, b \in \mathbb{S}$, we have an equivalence of*
(1) *$a \ominus a \oplus b > \mathbb{0}$*
(2) *$a \oplus b > \mathbb{0}$ and $\ominus a \oplus b > \mathbb{0}$*
(3) *For all $c \in [\ominus|a|, |a|] = \mathcal{U}(a \ominus a)$, it holds $c \oplus b > \mathbb{0}$.*

Proof. The condition (3) clearly implies (2), as the latter is just a special case. The implication from (2) to (1) follows by adding up the positive values in (2). For the direction from (1) to (3), we use that $a \ominus a$ is balanced and, hence, incomparable. Therefore, we have $b > |a|$. Because of $|a| = |\ominus a| = |a \ominus a|$, the claim follows. ◀

Given a matrix $A \in \mathbb{S}^{d\times n}$, let us construct the matrix $B = \xi(A) \in \mathbb{T}_{\pm}^{d'\times n}$ for $d \le d' \le 2d$ as follows. We include in B all the columns of A that do not contain any balanced elements. For every column a_{*j} that contains some balanced elements, we include two columns a' and a'' in B, such that whenever $a_{ij} = \bullet\alpha$ is a balanced entry, then we set $a'_i = \oplus\alpha$ and $a''_i = \ominus\alpha$. For all other $i \in [d]$, we set $a'_i = a''_i = a_{ij}$.

▶ **Theorem 4.12.** *For every matrix $A \in \mathbb{S}^{d\times n}$, $sep_+(A) = sep_+(\xi(A))$.*

Proof. Recalling Definition 23, we see that it is enough to consider the transition from A to B column by column. An inequality arising from A (after potentially reordering entries) is of the form $y^\top \odot (u, v) > \mathbb{0}$ with $u \in \mathbb{T}_{\pm}^k$ and $v \in \mathbb{T}_{\bullet}^{d-k}$ for some $k \le d$. It either remains the same in B if it has no balanced entry or it gets transformed to two inequalities. Regrouping the summands by balanced and signed numbers, we deduce the claim from the equivalence of (1) and (3) in Lemma 4.11. ◀

4.3 Fourier-Motzkin Elimination

We derive three versions of Fourier-Motzkin elimination, which will be useful for deriving further description of signed tropically convex sets in Section 5 and 6. As the elimination process produces balanced coefficients for the inequalities, it is convenient to have an elimination procedure which can directly deal with those (Theorem 4.14). We also need to derive explicit inequalities with signed coefficients (Corollary 4.15) to describe the dual convex hull in Section 6. The version with non-strict inequalities (Theorem 4.19) will be used in constructing an exterior description by closed tropical halfspaces (Theorem 5.4).

For a subset M of $\mathbb{T}_{\pm}^d$, we define its coordinate projection $\rho_i(M)$ with $i \in [d]$ by

$$\rho_i(M) = \{(x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_d) \mid (x_1, \dots, x_d) \in M\} \ . \tag{28}$$

We fix a matrix $A = (a_{ij}) \in \mathbb{S}^{d\times n}$. The non-negative matrix $S^{(i)}$ in the group of signed tropical transformations from Remark 3.10 with the sequence $(-|a_{ij}|)_j \in \mathbb{T}^n_{\ge \mathbb{0}}$ as diagonal scales the rows of A such that the ith row of $A \odot S$ has only entries with absolute value 0 or $\mathbb{0}$. As S is non-negative, we directly obtain from the definitions in Section 4.1 the following.

▶ **Lemma 4.13.** *There is a matrix $S^{(i)} \in \mathbb{D} \subset \mathbb{T}_{\pm}^{d\times d}$ such that the entries in the ith row of $A \odot S^{(i)}$ have absolute value 0 or $\mathbb{0}$, and $ker_+(A \odot S^{(i)}) = ker_+(A)$, $sep_+(A \odot S^{(i)}) = sep_+(A)$ and $\left\{ y \in \mathbb{T}_{\pm}^d \;\middle|\; y^\top \odot A \vDash \mathbb{0} \right\} = \left\{ y \in \mathbb{T}_{\pm}^d \;\middle|\; y^\top \odot S^{(i)} \odot A \vDash \mathbb{0} \right\}$.*

We define several sets partitioning $[n]$ based on the sign of the entries in the ith row of A by

$$\begin{aligned} J^+ &= \{j \in [n] \mid a_{ij} > \mathbb{0}\}\,, & J^- &= \{j \in [n] \mid a_{ij} < \mathbb{0}\} \\ J^\bullet &= \{j \in [n] \mid a_{ij} \in \mathbb{T}_\bullet \setminus \{\mathbb{0}\}\}\,, & J^0 &= \{j \in [n] \mid a_{ij} = \mathbb{0}\} \ . \end{aligned} \tag{29}$$

Furthermore, we define $T^{(i)} = (t_{j,p}) \in \{0, \mathbb{0}\}^{n \times ((J^+ \cup J^\bullet) \times (J^\bullet \cup J^-) \cup J^0)}$ as the incidence matrix

$$t_{j,p} = \begin{cases} 0 & j = k \text{ or } j = \ell \text{ for } p = (k,\ell) \in (J^+ \cup J^\bullet) \times (J^\bullet \cup J^-) \\ 0 & j = p \text{ for } p \in J^0 \\ \mathbb{0} & \text{else} \end{cases} . \tag{30}$$

We denote by A_{-i} the matrix obtained from A by removing the last row.

4.3.1 Strict Inequalities

▶ **Theorem 4.14** (Fourier–Motzkin for strict inequalities with balanced numbers)**.** *The ith coordinate projection of the open tropical cone $\rho_i(sep_+(A))$ for the matrix $A \in \mathbb{S}^{d \times n}$ is the open tropical cone $sep_+(A_{-i} \odot S^{(i)} \odot T^{(i)})$.*

Proof. By using an appropriate scaling matrix $S^{(i)}$, we can assume by Lemma 4.13 that the absolute value of each entry in the ith row of A is either 0 or $\mathbb{0}$ and that this does not affect $\rho_i(\mathrm{sep}_+(A))$.

To simplify notation, we additionally assume that $i = d$. Using Lemma 4.11 and ordering the inequalities according to the partition from (29), we get the system

$$\begin{aligned} y_d \oplus (y_1, \dots, y_{d-1}) \odot (a_{1,j}, \dots, a_{d-1,j})^\top &> \mathbb{0} \text{ for } j \in J^+ \cup J^\bullet \\ \ominus y_d \oplus (y_1, \dots, y_{d-1}) \odot (a_{1,j}, \dots, a_{d-1,j})^\top &> \mathbb{0} \text{ for } j \in J^- \cup J^\bullet \end{aligned} \tag{31}$$

Proposition 4.8 implies that (31) has a solution $(y_1, \dots, y_{d-1}, y_d) \in \mathbb{T}_\pm^d$ if and only if

$$(y_1, \dots, y_{d-1}) \odot A_{-d} \odot T > \mathbb{0} \tag{32}$$

has a solution $(y_1, \dots, y_{d-1}) \in \mathbb{T}_\pm^{d-1}$. ◀

We can resolve the balanced entries which can occur in the product with $T^{(i)}$ by means of Theorem 4.12.

▶ **Corollary 4.15** (Fourier–Motzkin for strict inequalities with signed numbers)**.** *The ith coordinate projection of the open tropical cone $\rho_i(sep_+(A))$ is $sep_+\left(\xi(A_{-i} \odot S^{(i)} \odot T^{(i)})\right)$.*

▶ **Example 4.16.** We use the matrix

$$A = \begin{pmatrix} 3 & \ominus 1 & \ominus 4 \\ 3 & \ominus 0 & \ominus 2 \end{pmatrix} ,$$

from Figure 1.

Eliminating the first row yields

$$\begin{pmatrix} 3 & \ominus 0 & \ominus 2 \end{pmatrix} \odot \begin{pmatrix} -3 & \mathbb{0} & \mathbb{0} \\ \mathbb{0} & -1 & \mathbb{0} \\ \mathbb{0} & \mathbb{0} & -4 \end{pmatrix} \odot \begin{pmatrix} 0 & 0 \\ 0 & \mathbb{0} \\ \mathbb{0} & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 \end{pmatrix}$$

This shows that $\mathrm{sep}_+^+(A)$ is not empty (as all entries of the remaining matrix are positive signed numbers), which can also be seen in Figure 1 using the duality in Theorem 4.6.

▶ Remark 4.17. The crucial difference to classical Fourier–Motzkin elimination happens in the treatment of balanced numbers occuring in the calculation. While classically in each step several variables could be eliminated at the same time we have to deal with their balanced left-overs. For strict inequalities, Lemma 4.11 provides a tool to resolve them by introducing two inequalities instead of one. We will see how to resolve them for non-strict inequalities in Proposition 5.7.

▶ Remark 4.18. The classical technique of Fourier-Motzkin for polytopes, see e.g. [19] has already been successfully adapted to tropical linear inequality systems in [8]. In the latter work, an algorithmic scheme to determine a projection of a tropical inequality system is described. In our Theorem 4.14, we do not have the non-negatively constrained variables but allow arbitrary elements of $\mathbb{T}_{\pm}$. Classically, one can represent an unconstrained variable as the difference of a pair of non-negative variables. Applying this technique to a system of the form $y^\top \odot A > \mathbb{0}$ with unconstrained $y \in \mathbb{T}_{\pm}^n$ for a matrix $A \in \mathbb{T}_{\pm}^{d\times n}$ yields the system

$$(u^\top, v^\top) \odot \begin{pmatrix} A \\ \ominus A \end{pmatrix} \; > \; \mathbb{0} \text{ with } u, v \in \mathbb{T}^d_{\geq \mathbb{0}} \; . \tag{33}$$

Reordering terms with coefficients in $\mathbb{T}_{<\mathbb{0}}$ to the other side of the inequality yields a system which allows to apply [8, Theorem 11]. However, the differences of non-negative variables are harder to resolve as there is no cancellation but it results in balanced entries. This makes our direct approach more tractable for unconstrained variables. Furthermore, we are also interested in the structure of the resulting inequalities in Section 5, whence our approach is more suitable for this setting.

The elimination procedure derived in the last section allows to prove the desired separation in $\mathbb{T}_{\pm}^d$.

Proof of Theorem 4.6. At first, we show the claim for $d = 1$. The set $\operatorname{sep}_+(A)$ is non-empty if and only if either all entries are positive or all entries are negative. Otherwise, we can select a balanced entry or a pair of entries with opposite sign by multiplication from the right.

If $\operatorname{sep}_+(A)$ is not empty, then Theorem 4.4 tells us that $\ker_+(A)$ is empty. So, we assume that $\operatorname{sep}_+(A)$ is empty. As the scaling of the columns of A does not change the sets $\operatorname{sep}_+(A)$ or $\ker_+(A)$, we can assume that the absolute value of the entries in the last row of A is 0 or $\mathbb{0}$. Let T be the matrix from (30). Then Theorem 4.14 shows that $\operatorname{sep}_+(A_{-d} \odot T)$ is empty. By induction, there is an element z in $\ker_+(A_{-d} \odot T)$. We show that $T \odot z \in \ker_+(A)$. By definition of T, the elements in the dth row of $A \odot T$ are all in $\mathbb{T}_\bullet$. This implies that the dth row of $A \odot T \odot z$ is in $\mathbb{T}_\bullet$. Additionally, the choice of z yields $A_{-d} \odot T \odot z \in \mathbb{T}_\bullet^{d-1}$. This finishes the proof. ◀

4.3.2 Non-strict Inequalities

While we mainly considered the strict inequalities as separators from the origin, in light of Theorem 4.6, we develop the elimination theory for non-strict inequalities with a view towards the representation of convex sets as intersection of closed halfspaces in Theorem 5.4. Therefore, we add another coordinate to treat affine halfspaces, which only occurred in Example 3.9 so far. Next, we derive the analogous statement to Theorem 4.14 for the relation "$\vDash$" instead of "$>$".

Note that this is substantially more subtle than the case of strict inequalities. We require the coefficient matrix of the inequalities to consist of signed numbers only. However, as each elimination step can result in balanced coefficients, we again need a method to resolve those. While this was rather easy for strict inequalities as shown in Theorem 4.12, we need to develop a more technical and less explicit machinery in Proposition 5.7.

We recall the matrices $S^{(i)}$ from Lemma 4.13 and $T^{(i)}$ from (30) with d replaced by $d+1$ and $J^\bullet = \emptyset$.

▶ **Theorem 4.19** (Fourier–Motzkin for non-strict inequalities with signed numbers). *The ith coordinate projection of the set*

$$\left\{ y \in \mathbb{T}_\pm^d \;\middle|\; (0, y^\top) \odot A \vDash \mathbb{0} \right\} \tag{34}$$

for $A \in \mathbb{T}_\pm^{(d+1)\times n}$ is the set

$$\left\{ z \in \mathbb{T}_\pm^{d-1} \;\middle|\; (0, z^\top) \odot A_{-i} \odot S^{(i)} \odot T^{(i)} \vDash \mathbb{0} \right\} \quad . \tag{35}$$

Proof. Using Lemma 4.13, we can assume that the absolute value of each entry in the ith row of A is either 0 or $\mathbb{0}$ by using an appropriate scaling matrix $S^{(i)}$.

To simplify notation, we additionally assume that $i = d$. Ordering the inequalities according to the distinction from (29), we get the system

$$\begin{aligned} y_d \oplus (0, y_1, \dots, y_{d-1}) \odot (a_{0,j}, a_{1,j}, \dots, a_{d-1,j})^\top &\vDash \mathbb{0} \text{ for } j \in J^+ \\ \ominus y_d \oplus (0, y_1, \dots, y_{d-1}) \odot (a_{0,j}, a_{1,j}, \dots, a_{d-1,j})^\top &\vDash \mathbb{0} \text{ for } j \in J^- \end{aligned} \tag{36}$$

Proposition 4.10 implies that (36) has a solution $(0, y_1, \dots, y_{d-1}, y_d) \in \mathbb{T}_\pm^{d+1}$ if and only if

$$(0, y_1, \dots, y_{d-1}) \odot A_{-d} \odot S^{(d)} \odot T^{(d)} \vDash \mathbb{0} \tag{37}$$

has a solution $(0, y_1, \dots, y_{d-1}) \in \mathbb{T}_\pm^d$. ◀

▶ **Example 4.20.** We will see how to obtain an exterior description by closed halfspaces in Theorem 5.4. To determine the exterior description of the one dimensional line segment from $\ominus 0$ to 1 in $\mathbb{T}_\pm$, one can eliminate x_1 and x_2 from the system

$$\ominus 0 \odot x_1 \oplus 1 \odot x_2 \ominus z \vDash \mathbb{0} \tag{38a}$$
$$0 \odot x_1 \ominus 1 \odot x_2 \oplus z \vDash \mathbb{0} \tag{38b}$$
$$0 \odot x_1 \oplus 0 \odot x_2 \ominus 0 \vDash \mathbb{0} \tag{38c}$$
$$\ominus 0 \odot x_1 \ominus 0 \odot x_2 \oplus 0 \vDash \mathbb{0} \tag{38d}$$
$$0 \odot x_1 \vDash \mathbb{0} \tag{38e}$$
$$0 \odot x_2 \vDash \mathbb{0} \tag{38f}$$

Eliminating x_1 yields

$$\bullet 1 \odot x_2 \bullet z \vDash \mathbb{0} \qquad \text{from (38a)\&(38b)} \tag{39a}$$
$$1 \odot x_2 \ominus z \ominus 0 \vDash \mathbb{0} \qquad \text{from (38a)\&(38c)} \tag{39b}$$
$$1 \odot x_2 \ominus z \vDash \mathbb{0} \qquad \text{from (38a)\&(38e)} \tag{39c}$$
$$\ominus 1 \odot x_2 \oplus z \oplus 0 \vDash \mathbb{0} \qquad \text{from (38d)\&(38b)} \tag{39d}$$
$$\bullet 0 \odot x_2 \bullet 0 \vDash \mathbb{0} \qquad \text{from (38d)\&(38c)} \tag{39e}$$
$$\ominus 0 \odot x_2 \oplus 0 \vDash \mathbb{0} \qquad \text{from (38d)\&(38e)} \tag{39f}$$
$$0 \odot x_2 \vDash \mathbb{0} \qquad \text{from (38f)} \tag{39g}$$

From further elimination of x_2 we get by ignoring redundant inequalities of (39)

$$\bullet z \bullet 0 \vDash \mathbb{0} \qquad \text{from (39b)\&(39d)} \tag{40a}$$
$$\ominus(-1) \odot z \oplus 0 \vDash \mathbb{0} \qquad \text{from (39b)\&(39f)} \tag{40b}$$
$$(-1) \odot z \oplus (-1) \vDash \mathbb{0} \qquad \text{from (39g)\&(39d)} \tag{40c}$$
$$0 \vDash \mathbb{0} \qquad \text{from (39g)\&(39f)} \tag{40d}$$

This yields the exterior description $z \vDash \ominus 0$ and $z \vDash \ominus 1$.

4.4 On the Complexity of Signed Tropical Inequality Systems

For (regular) linear programs, solving all standard forms such as $Ax = b$, $x \geq 0$ or $Ax \leq b$ are polynomial-time equivalent. For example, consider a system in the second form with $A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^m$. By using variables $x', x'' \in \mathbb{R}^n$ and $s \in \mathbb{R}^m$, we can write a system in the first form as $(A, -A, I_m)(x', x'', s) = b$, $(x', x'', s) \geq 0$. From a feasible solution (x', x'', s), we can recover a feasible solution $x = x' - x''$ to the original system.

The situation for tropical LP is different. Consider systems of the form $A \odot x \bowtie b$, $x \geq \mathbb{0}$, $x \in \mathbb{T}^d_\pm$ and $A \odot x \trianglerighteq b$, $x \in \mathbb{T}^d_\pm$; in both cases, we need to find a nonzero solution. The first system asks for $\ker_+ \begin{pmatrix} A & \ominus b \\ 0 & \ominus 0 \end{pmatrix} \neq \emptyset$, and is essentially equivalent to solving mean payoff games (1). Deciding whether this system is feasible is in the complexity class NP∩coNP; this can be seen from Farkas' lemma (Theorem 4.6), and an additional argument showing that if one of the respective systems is feasible, there is a solution of bit complexity bounded in the bit complexity of the input.

It turns out that systems of the form $A \odot x \trianglerighteq b$, $x \in \mathbb{T}^d_\pm$ actually describe a more general problem. We thank Mateusz Skomra for making the connection with the negative cycles in mixed graphs [10] and an anonymous referee for pointing us to a reduction leading to [29] as well as a hinting towrads a direct proof which we give here.

▶ **Theorem 4.21.** *The feasibility problem for systems of the form $A \odot x \trianglerighteq b$, $x \in \mathbb{T}^d_\pm$ is NP-complete.*

Proof. We give a reduction from SAT. For this, we fix an arbitrary SAT formula $\mathcal{F}$ in conjunctive normal form on d variables. To each clause $(l_1 \vee \cdots \vee l_k)$, composed of the literals $l_1, \ldots, l_k$, we associate an inequality $\sigma_1 \odot x_1 \oplus \cdots \oplus \sigma_k \odot x_k \vDash 0$ where σ_i is 0 or $\ominus 0$ depending if the corresponding literal in the clause is positive or negative. This yields a linear inequality system where the number of inequalities equals the number of clauses in the formula.

For a satisfying assignment $s \in \{\texttt{True}, \texttt{False}\}^d$, we construct a vector $x \in \{0, \ominus 0\}^d$ by replacing `True` with 0 and `False` with $\ominus 0$. As at least one literal yields `True` for the assignment s, each inequality becomes $0 \vDash 0$ or $\bullet 0 \vDash 0$, which are both true.

On the other hand, let $x \in \mathbb{T}^d_\pm$ be a solution of the associated tropical linear inequality system. We claim that then also $(\operatorname{tsgn}(x_1)\, 0, \ldots, \operatorname{tsgn}(x_d)\, 0)$ is a solution. Indeed, at least one term $\sigma_i \odot x_i$ of each inequality $\sigma_1 \odot x_1 \oplus \cdots \oplus \sigma_k \odot x_k \vDash 0$ has to be 0 or bigger (with respect to $\geq$). Hence, we can just set the absolute value of that component to 0 without changing the validity of the inequality. Each component which does not contribute to any of the left hand sides can take any value in the interval $[\ominus 0, 0]$ without violating any of the inequalities. Now, we can again use the correspondence between $(0, \ominus 0)$ and $(\texttt{True}, \texttt{False})$ to construct a satisfying assignment for $\mathcal{F}$.

Note that the problem is in NP as a solution can be checked to fulfill the inequalities in polynomial time. Hence, the problem is NP-complete. ◀

Recall that we do not obtain a variant of Farkas' lemma for the system $A \odot x \operatorname{\rotatebox[origin=c]{180}{$\vDash$}} b$, $x \in \mathbb{T}_{\pm}^d$ via the above Fourier-Motzkin elimination. The construction analogous to the classical case would transform it to $(A, \ominus A, I_m) \odot (x', x'', s) \bowtie b$, $(x', x'', s) \geq \mathbb{0}$. However, the solution $x = x' \ominus x''$ may contain balanced entries, and it may not be possible to resolve these entries to get a feasible solution.

▶ **Example 4.22.** We consider the system

$$\begin{pmatrix} 0 & 0 \\ \ominus 0 & 0 \\ 0 & \ominus 0 \\ \ominus 0 & \ominus 0 \end{pmatrix} \odot \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \operatorname{\rotatebox[origin=c]{180}{$\vDash$}} \begin{pmatrix} \ominus 0 \\ \ominus 0 \\ \ominus 0 \\ \ominus 0 \end{pmatrix} .$$

It does not have a signed solution. However the extended system

$$\begin{pmatrix} 0 & 0 & \ominus 0 & \ominus 0 & 0 & \mathbb{0} & \mathbb{0} & \mathbb{0} \\ \ominus 0 & 0 & 0 & \ominus 0 & \mathbb{0} & 0 & \mathbb{0} & \mathbb{0} \\ 0 & \ominus 0 & \ominus 0 & 0 & \mathbb{0} & \mathbb{0} & 0 & \mathbb{0} \\ \ominus 0 & \ominus 0 & 0 & 0 & \mathbb{0} & \mathbb{0} & \mathbb{0} & 0 \end{pmatrix} \odot \begin{pmatrix} x_1' \\ x_2' \\ x_1'' \\ x_2'' \\ s_1 \\ s_2 \\ s_3 \\ s_4 \end{pmatrix} \bowtie \begin{pmatrix} \ominus 0 \\ \ominus 0 \\ \ominus 0 \\ \ominus 0 \end{pmatrix} , \qquad x', x'', s \geq \mathbb{0}$$

has the solution $(0, 0, 0, 0, 0, 0, 0, 0)$.

Note that one can use [29, Theorem 1.2] to show that also systems of the form $A \odot x \bowtie b$, $x \in \mathbb{T}_{\pm}^d$ are NP-complete. This is in stark contrast to the intuition from classical equations. While systems of linear equations with the usual arithmetic are solvable in strongly polynomial time, it gets harder when we add non-negativity constraints. The latter is just linear programming and, hence, solvable in weakly polynomial time. Adding a non-negativity constraint to the system of balances above yields a tropical linear programming which is in NP ∩ co-NP.

5 Description by Halfspaces

An important property of classical polytopes is the duality between the representation as convex hull and as intersection of finitely many halfspaces. This is more subtle for tropical polytopes over $\mathbb{T}_{\pm}$. While we establish a description as intersection of open tropical halfspaces (Theorem 5.1) containing a set of points, we need additional properties to formulate a Minkowski-Weyl theorem (Theorem 5.4). All proofs are deferred to Section 5.1.

We saw already in Examples 3.9 that open halfspaces are convex. The *outer hull* of a set $M \subseteq \mathbb{T}_{\pm}^d$ is the intersection of its containing open halfspaces

$$\bigcap_{M \subseteq \mathcal{H}^+(a)} \mathcal{H}^+(a) \ . \tag{41}$$

This construction yields the same as our key Definition 3.1.

▶ **Theorem 5.1.** *The outer and the inner hull of a matrix $A \in \mathbb{T}_{\pm}^{d \times n}$ coincide, i. e.,* $\operatorname{tconv}(A) = \bigcap_{A \subseteq \mathcal{H}^+(v)} \mathcal{H}^+(v)$*, where we identify A with the set of its columns.*

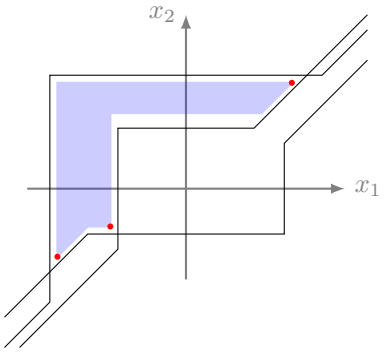


Figure 4 Approximation of a triangle by open halfspaces.

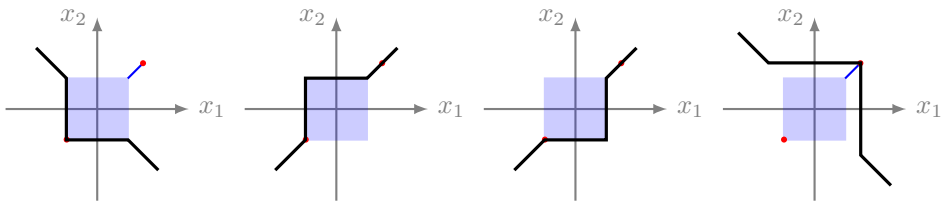


Figure 5 Exterior description of $\operatorname{tconv}((\ominus 1, \ominus 1), (2, 2))$ by closed halfspaces.

As the latter does not give a finite description, one is usually more interested in closed halfspaces. For a vector $(a_0, a_1, \dots, a_d) \in \mathbb{T}_{\pm}^{d+1}$, we define a *closed (signed) tropical halfspace* by

$$\overline{\mathcal{H}}^{+}(a) = \left\{ x \in \mathbb{T}_{\pm}^{d} \;\middle|\; a \odot \begin{pmatrix} 0 \\ x \end{pmatrix} \in \mathbb{T}_{\geq \mathbb{0}} \cup \mathbb{T}_{\bullet} \right\} . \tag{42}$$

▶ **Lemma 5.2.** *The closed signed tropical halfspace $\overline{\mathcal{H}}^{+}(a)$ is the topological closure of the open signed halfspace $\mathcal{H}^{+}(a)$.*

Observe that the former statement is wrong for inequalities with balanced numbers as coefficients.

▶ Remark 5.3. Closed tropical halfspaces are not as suitable for the hull construction in Equation (41) as open tropical halfspaces. The inner hull of $M = \{(\ominus 1, 1), (1, \ominus 1)\}$ should contain the origin $\mathbb{0}$. However, the closed halfspace $x_1 \oplus x_2 \geq \ominus 0$ contains those two points but not the origin. Taking the analogous intersection as in (41) with closed tropical halfspaces for M yields again M. Note that this is the same as the intersection of the balanced image $M \odot \Delta_2$ with $\mathbb{T}_{\pm}^{2}$.

Example 3.8 shows that such a closed signed tropical halfspace is in general not tropically convex. However, a finite intersection of such halfspaces can be tropically convex, as Figure 5 shows.

Hence, to arrive at a Minkowski-Weyl theorem for tropical polytopes over $\mathbb{T}_{\pm}$, one has to adapt the condition in the characterization of finite intersections of closed tropical halfspaces.

▶ **Theorem 5.4.** *For each finite set $V \subset \mathbb{T}_{\pm}^{d}$, there are finitely many closed tropical halfspaces H such that $\operatorname{tconv}(V)$ is the intersection of the halfspaces.*

For each finite set H of closed halfspaces, whose intersection M is tropically convex, there is a finite set of points $V \in \mathbb{T}_{\pm}^{d}$ such that $M = \operatorname{tconv}(V)$.

▶ **Corollary 5.5.** *A tropically convex set is the intersection of its containing closed halfspaces.*

▶ Remark 5.6. The crucial difference with Theorem 5.1 is that for open tropical halfspaces the generators are enough, while for closed tropical halfspaces, we have to take the whole set into account.

5.1 Proofs

To prove that the inner hull and the outer hull coincide, we mainly have to use the separation results from Section 4.

Proof of Theorem 5.1. The inclusion $\operatorname{tconv}(A) \subseteq \bigcap_{A \subseteq \mathcal{H}^+(v)} \mathcal{H}^+(v)$ follows by combining Proposition 3.4.a and Example 3.9.

For the other inclusion, assume that there is a point

$$z \in \bigcap_{A \subseteq \mathcal{H}^+(v)} \mathcal{H}^+(v) \setminus \operatorname{tconv}(A) \ .$$

By Proposition 4.1, we get that $\ker_+(B) = \emptyset$, where

$$B = \begin{pmatrix} 0 & \ominus 0 \\ A & \ominus z \end{pmatrix} \ .$$

Theorem 4.6 implies that $\operatorname{sep}_+(B) \neq \emptyset$. Hence, there is a separator $(u_0, \bar{u}) = (u_0, u_1, \dots, u_d) \in \mathbb{T}_\pm^{d+1}$ with $u_0 \oplus \bar{u}^\top \odot a^{(j)} > \mathbb{O}$ for all columns $a^{(j)}$ of A, and $\ominus u_0 \oplus \ominus \bar{u}^\top \odot z > \mathbb{O} \Leftrightarrow u_0 \oplus \bar{u}^\top \odot z < \mathbb{O}$. This means that the columns of A lie in the halfspace $\mathcal{H}^+(u)$ but z does not. This contradicts the choice of z in $\bigcap_{A \subseteq \mathcal{H}^+(v)} \mathcal{H}^+(v)$. ◀

Showing that a closed tropical halfspace is obtained as the closure of an open tropical halfspace follows from a simple limit argument. We use the notation $[d]_0 = \{0, 1, 2, \dots, d\}$.

Proof of Lemma 5.2. Let $z \in \mathbb{T}^d_{\geq \mathbb{O}}$ be an element of $\overline{\mathcal{H}^+}(a) \setminus \mathcal{H}^+(a)$. Set

$$J = \operatorname{argmax} \{ |c_j \odot z_j| \mid j \in [d]_0 \} \ ,$$

where $z_0 = 0$, and let $\ell \in J$ with $c_\ell \odot z_\ell > \mathbb{O}$. Denoting the kth unit vector $(0, \dots, 0, 1, 0, \dots, 0)$ in $\mathbb{R}^d$ by e_k for $k \in [d]$ and $e_0 = (-1, \dots, -1)$, we define a sequence

$$y^{(m)} = z + \frac{1}{m} \cdot e_\ell \ .$$

The sequence converges to z but each element of the sequence is an element of $\mathcal{H}^+(a)$. ◀

We will use the Fourier-Motzkin version for the relation "$\vDash$" (Theorem 4.19) to deduce an exterior description of a tropical polytope. However, the system describing the projection in (35) may contain balanced coefficients. We address this issue in the next statement. It is an existence argument statement that a balanced coefficient can be replaced by a signed coefficient, see also Figure 6.

▶ **Proposition 5.7** (Resolving balanced coefficients)**.** *Let M be a tropically convex set and $c = (c_0, c_1, \dots, c_d) \in \mathbb{S}^d$ with $c_i \in \mathbb{T}_\bullet$ for some $i \in [d]_0$ such that*

$$M \subseteq H(c) = \left\{ x \in \mathbb{T}^d_\pm \;\middle|\; c \odot \begin{pmatrix} 0 \\ x \end{pmatrix} \vDash \mathbb{O} \right\} \ .$$

Then there is an element $b \in \mathcal{U}(c_i)$ such that M is contained in

$$\left\{ x \in \mathbb{T}^d_\pm \;\middle|\; (c_0, \dots, c_{i-1}, b, c_{i+1}, \dots, c_d) \odot \begin{pmatrix} 0 \\ x \end{pmatrix} \vDash \mathbb{O} \right\} \ .$$

Proof. If $i = 0$, then we can set $b = |c_i|$.

Assuming without loss of generality that $i = d$, we set $c_{-d} = (c_0, c_1, \dots, c_{d-1})$. Fix $u \in \mathbb{T}_{\pm}^{d-1}, v \in \mathbb{T}_{\pm}$ such that $(u, v) \in M$. We define $w = w(u) = c_{-d} \odot \begin{pmatrix} 0 \\ u \end{pmatrix}$.

If $v \in \mathbb{T}_{<\mathbb{0}}$, then

$$\lambda(u, v) = \operatorname{argmax} \{ \lambda \in \mathcal{U}(c_d) \mid w \oplus \lambda \odot v \vDash \mathbb{0} \} \quad . \tag{43}$$

Let

$$\underline{\lambda} = \min \{ \lambda(u, v) \mid (u, v) \in M, v < \mathbb{0} \} \quad . \tag{44}$$

If $v \in \mathbb{T}_{>\mathbb{0}}$, then

$$\lambda(u, v) = \operatorname{argmin} \{ \lambda \in \mathcal{U}(c_d) \mid w \oplus \lambda \odot v \vDash \mathbb{0} \} \quad . \tag{45}$$

Let

$$\overline{\lambda} = \max \{ \lambda(u, v) \mid (u, v) \in M, v > \mathbb{0} \} \quad . \tag{46}$$

We derive a contradiction to the convexity of M, if $\overline{\lambda} > \underline{\lambda}$.

Let (p, q) and (r, s) attain $\underline{\lambda}$ and $\overline{\lambda}$, respectively. In particular, we have $q < \mathbb{0}$ and $s > \mathbb{0}$. The inequality $\overline{\lambda} > \underline{\lambda}$ implies that $\overline{\lambda} > \ominus|c_d|$ and that $\underline{\lambda} < |c_d|$. We can assume that $w(p)$ and $w(r)$ are signed numbers, as we otherwise can replace them by their absolute value without changing the admissible values of λ in (43) and (45).

Now, the construction of $\underline{\lambda}$ in (43) implies that $w(p) = \ominus\underline{\lambda} \odot q$ and (45) yields $w(r) = \ominus\overline{\lambda} \odot s$. This implies

$$w(r) \odot s^{\odot -1} = \ominus\overline{\lambda} < \ominus\underline{\lambda} = w(p) \odot q^{\odot -1} \quad . \tag{47}$$

We consider the point in the convex combination of (p, q) and (r, s) given by

$$z = (\ominus q^{\odot -1} \oplus s^{\odot -1})^{\odot -1} \odot (\ominus q^{\odot -1} \odot p \oplus s^{\odot -1} \odot r, \mathbb{0}) \quad .$$

Then

$$\begin{aligned} c \odot \begin{pmatrix} 0 \\ z \end{pmatrix} &= c \odot \begin{pmatrix} 0 \\ (\ominus q^{\odot -1} \oplus s^{\odot -1})^{\odot -1} \odot (\ominus q^{\odot -1} \odot p \oplus s^{\odot -1} \odot r) \\ \mathbb{0} \end{pmatrix} = \\ &= (\ominus q^{\odot -1} \oplus s^{\odot -1})^{\odot -1} \odot \begin{pmatrix} \ominus q^{\odot -1} \odot c_{-d} \odot \begin{pmatrix} 0 \\ p \end{pmatrix} \oplus s^{\odot -1} \odot c_{-d} \odot \begin{pmatrix} 0 \\ r \end{pmatrix} \\ \mathbb{0} \end{pmatrix} = \\ &= (\ominus q^{\odot -1} \oplus s^{\odot -1})^{\odot -1} \odot \begin{pmatrix} \ominus q^{\odot -1} \odot w(p) \oplus s^{\odot -1} \odot w(r) \\ \mathbb{0} \end{pmatrix} \end{aligned}$$

By (47), this implies $c \odot \begin{pmatrix} 0 \\ z \end{pmatrix} < \mathbb{0}$. As M is convex, this yields $z \in M \subseteq H(c)$, a contradiction.

So, we can conclude that $\overline{\lambda} \leq \underline{\lambda}$. Fix any b in the interval $[\overline{\lambda}, \underline{\lambda}] \neq \emptyset$. Let $(u, v) \in M$ with $v < \mathbb{0}$. Then

$$(c_{-d}, b) \odot \begin{pmatrix} 0 \\ u \\ v \end{pmatrix} = w(u) \oplus b \odot v \vDash w(u) \oplus \underline{\lambda} \odot v \vDash w(u) \oplus \lambda(u, v) \odot v \vDash \mathbb{0} \quad ,$$

where we use $b \leq \underline{\lambda}$, (44) and Lemma 2.6.d. The proof for $v > \mathbb{0}$ goes analogously. ◀

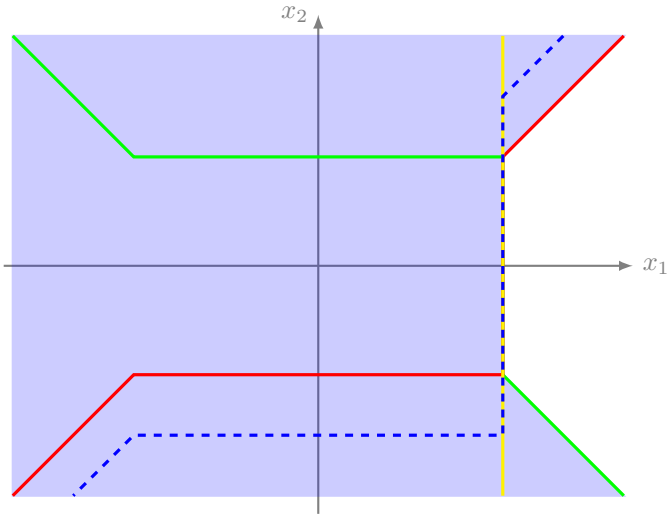


Figure 6 Finding a containing halfspace without balanced coefficients.

▶ **Example 5.8.** A pathological example for the last statement arises from resolving the tautological relation $\bullet(-1) \odot x \oplus \bullet(-1) \odot y \oplus \bullet 0 \vDash \mathbb{0}$. One obtains the chain of relations

$$\begin{aligned} \bullet(-1) \odot x \oplus \bullet(-1) \odot y \oplus \bullet 0 \vDash \mathbb{0} &\Leftrightarrow \\ \bullet(-1) \odot x \oplus \bullet(-1) \odot y \oplus 0 \vDash \mathbb{0} &\Leftrightarrow \\ \bullet(-1) \odot x \oplus \mathbb{0} \odot y \oplus 0 \vDash \mathbb{0} &\Leftrightarrow \\ \mathbb{0} \odot x \oplus \mathbb{0} \odot y \oplus 0 \vDash \mathbb{0} \ . & \end{aligned}$$

▶ Remark 5.9. Indeed, any value of $b \in \mathcal{U}(c_d)$ in the former proof could occur as Figure 6 shows. Any part of the dashed line without opposite (with respect to the origin) points is a tropical line segment.

▶ **Example 5.10.** The shaded area in Figure 6 shows the feasible region

$$\left\{ (x,y) \in \mathbb{T}_{\pm}^2 \;\middle|\; \ominus(-1) \odot x_1 \oplus (\bullet 2) \odot x_2 \oplus 0 \vDash \mathbb{0} \right\} \ .$$

The red line marks the inequality $\ominus(-1) \odot x_1 \oplus 2 \odot x_2 \oplus 0 \vDash \mathbb{0}$, while the green line marks the inequality $\ominus(-1) \odot x_1 \oplus (\ominus 2) \odot x_2 \oplus 0 \vDash \mathbb{0}$. The yellow line corresponds to the inequality $\ominus(-1) \odot x_1 \oplus 0 \vDash \mathbb{0}$. The blue dashed line interpolates between these three possible extreme closed halfspaces contained in the feasible region.

Our proof of Theorem 5.4 is based on eliminating variables from the canonical exterior description (13). For using those halfspaces, we need to show the additional requirement of tropical convexity for their intersection.

▶ **Lemma 5.11.** *The set*

$$\left\{ (x,z) \in \mathbb{T}_{\pm}^{n+d} \;\middle|\; A \odot x \bowtie z,\ x \geq \mathbb{0} \right\} \tag{48}$$

is tropically convex.

Proof. It is enough to show that, for fixed $a \in \mathbb{T}_{\pm}^n$, the set

$$H = \left\{ (x,z) \in \mathbb{T}_{\pm}^{n+1} \;\middle|\; a \odot x \bowtie z,\ x \geq \mathbb{0} \right\}$$

is tropically convex, then the claim follows from Proposition 3.4(a). Let $(p,q),(r,s) \in H$, and $\lambda \in \mathbb{T}_{\geq \mathbb{0}}$, $\lambda \leq 0$. We need to show that $\mathcal{U}(p \oplus \lambda \odot r, q \oplus \lambda \odot s) \subseteq H$.

Note that since $p, r \geq \mathbb{0}$, we have $p \oplus \lambda \odot r \in \mathbb{T}_{\pm}$ and therefore $\mathcal{U}(p \oplus \lambda \odot r) = \{p \oplus \lambda \odot r\}$. By Lemma 2.4(b), we have that $q \in \mathcal{U}(a \odot p)$ and $s \in \mathcal{U}(a \odot r)$. Using Lemma 3.5(b), we see that

$$\mathcal{U}(q \oplus \lambda \odot s) \subseteq \mathcal{U}((a \odot p) \oplus \lambda \odot (a \odot r)) = \mathcal{U}(a \odot (p \oplus \lambda \odot r)) \ ,$$

completing the proof. ◀

Proof of Theorem 5.4. Equation (13) provides a description by halfspaces involving additional variables. The convex hull of V is the set of those $z \in \mathbb{T}_{\pm}^d$ for which there is an $x \in \mathbb{T}_{\pm}^n$ with

$$\begin{pmatrix} A & \ominus \boldsymbol{I}_d & \mathbb{0} \\ \ominus A & \boldsymbol{I}_d & \mathbb{0} \\ 0 & \mathbb{0}_d & \ominus 0 \\ \ominus 0 & \mathbb{0}_d & 0 \\ \boldsymbol{I}_n & \mathbb{0}_d & \mathbb{0} \end{pmatrix} \odot \begin{pmatrix} x \\ z \\ 0 \end{pmatrix} \vDash \mathbb{0} \ , \tag{49}$$

where $\boldsymbol{I}$ is a tropical identity matrix with 0 on the diagonal and $\mathbb{0}$ elsewhere.

By Lemma 5.11 and the tropical convexity of $\left\{ x \in \mathbb{T}_{\pm}^n \;\middle|\; \bigoplus_{j \in [n]} x_j = 0, x \geq \mathbb{0} \right\}$, the set of $(x,z) \in \mathbb{T}_{\pm}^{n+d}$ fulfilling (49) is the intersection of tropically convex sets and, by Proposition 3.4(a), tropically convex as well.

Hence, we can use Theorem 4.19 to successively project out the x-variables. As the inequalities arising from a projection may contain balanced coefficients, we use Proposition 5.7 to replace them by signed coefficients. This yields a description of $\operatorname{tconv}(A)$ by non-strict inequalities. Here, we use that resolving a balanced coefficient in an inequality leads to a tighter constraint.

If the intersection of closed tropical halfspaces M is tropically convex, then its intersection with any orthant is tropically convex. By the tropical Minkowski-Weyl theorem in the non-negative orthant [24], each of the parts in the orthants are finitely generated. Taking the tropical convex hull of the union of all these generators yields M, as M is tropically convex. ◀

6 Connection with Tropical Convexity in $\mathbb{T}_{\max}$

The intersection of a signed tropically convex set with an orthant is just a tropically convex set over $\mathbb{T}_{\max}$. This allows to study signed tropical convex sets through the existing theory of unsigned tropically convex sets with the hull (12) in each orthant of $\mathbb{T}_{\pm}^d$. The proofs of the main statements are deferred to Section 6.1; these are based on the duality between the non-negative kernels and open tropical cones.

We fix a (finite) set $M \subseteq \mathbb{T}_{\pm}^d$ and interpret M as a matrix whose columns list the points. We will augment M by iteratively intersecting line segments between points in M with the coordinate hyperplanes to get a matrix $\zeta(M)$.

To make the construction of $\zeta(M)$ formal, we start with a slight modification of (30). We fix $A = (a_{ij}) \in \mathbb{T}_{\pm}^{d \times n}$ and a row index $i \in [d]$.

For two columns u and v of A with $u_i > \mathbb{O}$ and $v_i < \mathbb{O}$, we set

$$\begin{aligned}\lambda^+(u,v) &= \ominus v_i^{\odot -1} \odot (u_i^{\odot -1} \oplus \ominus v_i^{\odot -1})^{\odot -1} \text{ and}\\ \lambda^-(u,v) &= u_i^{\odot -1} \odot (u_i^{\odot -1} \oplus \ominus v_i^{\odot -1})^{\odot -1} \ .\end{aligned} \tag{50}$$

These scalars fulfill $\lambda^+(u,v) \oplus \lambda^-(u,v) = 0$ and $\lambda^+(u,v), \lambda^-(u,v) \geq \mathbb{O}$.

we define $T^{(i)} = (t_{j,p}) \in \{0, \mathbb{O}\}^{n \times ((J^+ \cup J^\bullet) \times (J^\bullet \cup J^-) \cup J^0)}$, using the decomposition from (29) for A, as the incidence matrix

$$t_{j,p} = \begin{cases} \lambda^+(a^{(k)}, a^{(\ell)}) & j = k \text{ for } p = (k,\ell) \in (J^+ \cup J^\bullet) \times (J^\bullet \cup J^-) \\ \lambda^-(a^{(k)}, a^{(\ell)}) & j = \ell \text{ for } p = (k,\ell) \in (J^+ \cup J^\bullet) \times (J^\bullet \cup J^-) \\ 0 & j = p \text{ for } p \in J^0 \\ \mathbb{O} & \text{else} \end{cases} \ . \tag{51}$$

In Theorem 4.14, we could also have used the matrix from (51) instead of multiplying with both $S^{(i)}$ and $T^{(i)}$. This is technically slightly more complicated but has the advantage that the resulting columns of the product with A are tropical convex combinations of the original columns. This motivates the following.

▶ **Definition 6.1.** *The matrix $\zeta_i(A)$ is the* ith elimination matrix *of A and it arises from $\xi(A \odot T)$ by replacing the ith row with $\mathbb{O}$.*

This allows to describe generators for the intersection of a tropical convex hull with a coordinate hyperplane $H_i := \left\{ x \in \mathbb{T}_\pm^d \;\middle|\; x_i = \mathbb{O} \right\}$ for $i \in [d]$.

▶ **Proposition 6.2.** *The intersection $\operatorname{tconv}(A) \cap H_i$ is generated by $\zeta_i(A)$.*

Additionally, we get generators for the intersection with an orthant.

▶ **Proposition 6.3.** *The intersection $\operatorname{tconv}(A) \cap \mathbb{T}^d_{\geq \mathbb{O}}$ is generated by*

$$\left(A \cup \bigcup_{i \in [d]} (\operatorname{tconv}(A) \cap H_i) \right) \cap \mathbb{T}^d_{\geq \mathbb{O}} \ .$$

Now, we have to combine all possible ways of descending in a chain of intersections of coordinate hyperplanes. Each permutation σ in the symmetric group on d elements S_d gives rise to a sequence of matrices $\zeta_{\sigma(1)}(M)$, $\zeta_{\sigma(2)}(\zeta_{\sigma(1)}(M))$ until $\zeta_{\sigma(d)}(\dots(\zeta_{\sigma(2)}(\zeta_{\sigma(1)}(M)))\dots)$. We denote the concatenation of these d matrices by $\zeta_\sigma(M)$. The concatenation of the matrices for all $\sigma \in S_d$ forms the matrix $\zeta(M)$.

▶ **Theorem 6.4.** *The convex hull $\operatorname{tconv}(M)$ of M is the union*

$$\bigcup_{O \text{ closed orthant of } \mathbb{T}^d_\pm} \operatorname{tconv}(O \cap \zeta(M)) \ . \tag{52}$$

Proof. By Proposition 6.3, we know that $\operatorname{tconv}(M)$ is generated by the projections on the boundary of the orthants and the generators in the interior. Iteratively applying Proposition 6.2 to the intersection of coordinate hyperplanes yields the claim. ◀

▶ **Example 6.5.** Looking at the points from Example 3.2, we see how we can determine the tropical convex hull of $\{(3,3), (\ominus 1, \ominus 0), (\ominus 4, \ominus 2)\}$. It is the union of the tropical convex hulls

$$\begin{aligned}&\operatorname{tconv}\left(\{(3,3),(\mathbb{O},3),(\mathbb{O},1)\}\right), \quad \operatorname{tconv}\left(\{(\ominus 1,0),(\mathbb{O},1),(\mathbb{O},3),(\ominus 4,\mathbb{O})\}\right)\\ &\operatorname{tconv}\left(\{(\ominus 1,\ominus 0),(\ominus 4,\ominus 2),(\ominus 1,\mathbb{O}),(\ominus 4,\mathbb{O})\}\right) \ .\end{aligned}$$

On the other hand, to get the tropical convex hull $\operatorname{tconv}((0,0),(\ominus-2,\ominus-2))$ in Figure 2a, one needs actual multi-valued cancellation.

We get

$$\zeta_1\left(\begin{pmatrix}0\\0\end{pmatrix},\begin{pmatrix}\ominus-2\\\ominus-2\end{pmatrix}\right)=\left\{\begin{pmatrix}\mathbb{O}\\-2\end{pmatrix},\begin{pmatrix}0\\\ominus-2\end{pmatrix}\right\} \text{ and}$$
$$\zeta_2\left(\begin{pmatrix}0\\0\end{pmatrix},\begin{pmatrix}\ominus-2\\\ominus-2\end{pmatrix}\right)=\left\{\begin{pmatrix}-2\\\mathbb{O}\end{pmatrix},\begin{pmatrix}\ominus-2\\\mathbb{O}\end{pmatrix}\right\}.$$

From applying $\zeta_1(\zeta_2(.))$, we additionally obtain $\{(\mathbb{O},\mathbb{O})\}$.

For the positive orthant, this yields the generators $\{(\mathbb{O},\mathbb{O}),(\mathbb{O},-2),(-2,\mathbb{O}),(0,0)\}$. The other orthants are derived analogously.

▶ **Corollary 6.6.** *Tropically convex sets are contractible.*

Proof. The space $\mathbb{T}_{\pm}^d$ inherits the topology of $\mathbb{R}^d$ via the map $\operatorname{slog}\colon x\mapsto \operatorname{sgn}(x)\log(|x|)$, where the origin is mapped to the all-$\mathbb{O}$-point. As tropically convex sets in all orthants are contractible [20, Theorem 2], we can contract to the boundary of the orthants. The claim follows by induction on the dimension d. ◀

For the definition of the covector decomposition in $(\mathbb{R}\cup\{-\infty\})^d$ and its connection with regular subdivisions of $\Delta_d\times\Delta_n$ we refer the reader to [28].

▶ **Corollary 6.7** (Covector decomposition)**.** *The combinatorics of the tropically convex hull of a matrix $A\in\mathbb{T}_{\pm}^{d\times n}$ can be described by 2^d regular subdivisions of $\Delta_d\times\Delta_n$.*

6.1 Proofs

Proof of Proposition 6.2. Using the construction in (51) and the fact that $\ominus|z|,|z|\in\mathcal{U}(z)$ for $z\in\mathbb{T}_\bullet$, the inclusion $\zeta_i(A)\subseteq\operatorname{tconv}(A)\cap H_i$ follows from Definition 3.1. Example 3.8 and Proposition 3.4.a imply that $\operatorname{tconv}(\zeta_i(A))\subseteq\operatorname{tconv}(A)\cap H_i$.

For the other inclusion, assume that there is a point $z\in\operatorname{tconv}(A)$ with $z_i=\mathbb{O}$ which is not contained in $\operatorname{tconv}(\zeta_i(A))$. By Proposition 4.1, this implies that

$$\ker_+\begin{pmatrix}0&\ominus 0\\ \zeta_i(A)&\ominus z\end{pmatrix}=\emptyset\ .$$

The Farkas Lemma 4.6 implies that

$$\operatorname{sep}_+\begin{pmatrix}0&\ominus 0\\ \zeta_i(A)&\ominus z\end{pmatrix}\neq\emptyset\ .$$

Furthermore, we get

$$\operatorname{sep}_+(\zeta_i\begin{pmatrix}0&\ominus 0\\ A&\ominus z\end{pmatrix})\neq\emptyset$$

as, because of $z_i=\mathbb{O}$, the last column is unchanged by ζ_i and the first row remains the same due to the definition of $\lambda^+(u,v)$ and $\lambda^-(u,v)$ for (50).

However, by Corollary 4.15, then also

$$\operatorname{sep}_+\begin{pmatrix}0&\ominus 0\\ A&\ominus z\end{pmatrix}\neq\emptyset\ .$$

Using again the Farkas Lemma 4.6, this implies $z\notin\operatorname{tconv}(A)$, a contradiction. ◀

To derive the next claim, we start with a more precise description of a signed tropical line segment. There is a natural bijection between $\Delta_2 = \left\{ (\nu, \mu) \in \mathbb{T}^2_{\geq \mathbb{0}} \;\middle|\; \max(\nu, \mu) = 0 \right\}$ and $\overline{\mathbb{T}} = \mathbb{R} \cup \{-\infty, \infty\}$ given by

$$(\nu, \mu) \mapsto \mu - \nu \ .$$

We denote the inverse image of an element $\eta \in \overline{\mathbb{T}}$ with respect to this map by $\Psi(\eta)$. This leads to a parametrization of a tropical line segment for $a, b \in \mathbb{T}^d_{\pm}$ via $\operatorname{tconv}(a, b) = \left\{ L_\eta(a, b) \;\middle|\; \eta \in \overline{\mathbb{T}} \right\}$ where $L_\eta(a, b) = \Psi(\eta)_0 \odot a \oplus \Psi(\eta)_1 \odot b$. Note that $L_{-\infty}(a, b) = a$ and $L_\infty(a, b) = b$.

Proof of Proposition 6.3. Let $z \in \operatorname{tconv}(A) \cap \mathbb{T}^d_{\geq \mathbb{0}}$ be an element of $\mathcal{U}(A \odot \lambda)$ with $\lambda \in \Delta_d$. We consider the tropical line segments from z to the columns of A. For a fixed column $a^{(j)}$ of A with $a^{(j)} \notin \mathbb{T}^d_{\geq \mathbb{0}}$, there is a minimal $\eta \in \overline{\mathbb{T}}$ such that a component of $L_\eta(z, a^{(j)})$ is balanced.

Intermediate claim I: All entries of $L_\eta(z, a^{(j)})$ are either balanced or in $\mathbb{T}_{\geq \mathbb{0}}$. For an arbitrary row $i \in [d]$, the expression $\Psi(\eta)_1 \odot z_i \oplus \Psi(\eta)_2 \odot a^{(j)}_i$ is in $\mathbb{T}_{\geq \mathbb{0}}$ for $\eta = -\infty$. The claim now follows from the piecewise definition of the addition in terms of the absolute values.

Using Proposition 3.6, we see that the point $b^{(j)}$ obtained from $L_\eta(z, a^{(j)})$ by replacing all balanced entries with $\mathbb{0}$ is in $\operatorname{tconv}(A) \cap \mathbb{T}^d_{\geq \mathbb{0}}$. For $a^{(j)} \in \mathbb{T}^d_{\geq \mathbb{0}}$ we set $b^{(j)} = a^{(j)}$.

Intermediate claim II: The point z is in the convex hull of $\left\{ b^{(j)} \;\middle|\; j \in [n] \right\}$. It is enough to show that

$$\ker_+\left(\begin{pmatrix} 0 & \dots & 0 & 0 & \ominus 0 \\ a^{(1)} & \dots & a^{(n-1)} & b^{(n)} & \ominus z \end{pmatrix}\right) \neq \emptyset$$

because than we can iteratively replace the columns $a^{(i)}$ by $b^{(i)}$. Let $b^{(1)} \in \nu \odot z \oplus \mu \odot a^{(1)}$. Pick an element x scaled such that $x_1 = \mu$. Then

$$\begin{pmatrix} \nu & \mathbb{0} & \dots & \mathbb{0} & \ominus \nu \\ \nu \odot z & \mathbb{0} & \dots & \mathbb{0} & \nu \odot \ominus z \end{pmatrix} \oplus \begin{pmatrix} 0 & 0 & \dots & 0 & \ominus 0 \\ a^{(1)} & a^{(2)} & \dots & a^{(n)} & \ominus z \end{pmatrix} \odot x$$

is in $\mathbb{T}^d_\bullet$. Therefore $(0, \dots, 0)$ is in the non-negative kernel of

$$\begin{pmatrix} \nu \oplus x_1 & x_2 & \dots & x_n & \ominus(\nu \oplus x_{n+1}) \\ \nu \odot z \oplus x_1 \odot a^{(1)} & x_2 \odot a^{(2)} & \dots & x_n \odot a^{(n)} & (\nu \oplus x_{n+1}) \odot (\ominus z) \end{pmatrix} = \begin{pmatrix} 0 & x_2 & \dots & x_n & \ominus(\nu \oplus x_{n+1}) \\ \nu \odot z \oplus \mu \odot a^{(1)} & x_2 \odot a^{(2)} & \dots & x_n \odot a^{(n)} & (\nu \oplus x_{n+1}) \odot (\ominus z) \end{pmatrix} \tag{53}$$

For fixed $i \in [d]$, if $\nu \odot z_i \oplus \mu \odot a^{(1)}_i$ is not balanced or the maximum absolute value is attained somewhere else in the row, we can replace it by $b^{(1)}_i$ and $(0, \dots, 0)$ is still in the non-negative kernel.

Otherwise, $\ominus \nu \odot z_i = \mu \odot a^{(1)}_i$ and $\nu \oplus x_{n+1} = \nu$. But $(0, \dots, 0)$ is also in the non-negative kernel of

$$\begin{pmatrix} \mu & x_2 & \dots & x_n & \ominus x_{n+1} \\ \mu \odot a^{(1)} & x_2 \odot a^{(2)} & \dots & x_n \odot a^{(n)} & x_{n+1} \odot (\ominus z) \end{pmatrix} \ .$$

Since $\mu \odot a^{(1)}_i = \ominus \nu \odot z_i$ has the same sign as $\ominus x_{n+1} \odot z_i$ and we have $\nu \geq x_{n+1}$, there has to be an $\ell \in [n]$ such that $x_\ell \odot a^{(\ell)}_i = \ominus \mu \odot a^{(1)}_i = \nu \odot z_i$. Therefore, we can replace $\nu \odot z_i \oplus x_1 \odot a^{(1)}$ by $\mathbb{0}$ and $(0, \dots, 0)$ remains in the non-negative kernel.

Therefore, $(0, x_2, \dots, x_n, (\nu \oplus x_{n+1}))$ is in the non-negative kernel of

$$\begin{pmatrix} 0 & 0 & \dots & 0 & \ominus 0 \\ b^{(1)} & a^{(2)} & \dots & a^{(n)} & \ominus z \end{pmatrix} .$$

This finishes the proof of claim II. ◀

References

1 M. Akian, G. Cohen, S. Gaubert, R. Nikoukhah, and J. P. Quadrat. Linear systems in (max, +) algebra. In *29th IEEE Conference on Decision and Control*, pages 151–156 vol.1, December 1990. doi:10.1109/CDC.1990.203566.

2 Marianne Akian, Xavier Allamigeon, Stéphane Gaubert, and Sergei Sergeev. Tropical optimization with signs, in preparation.

3 Marianne Akian, Stéphane Gaubert, and Alexander Guterman. Tropical polyhedra are equivalent to mean payoff games. *Internat. J. Algebra Comput.*, 22(1):1250001, 43, 2012. doi:10.1142/S0218196711006674.

4 Marianne Akian, Stéphane Gaubert, and Alexander Guterman. Tropical Cramer determinants revisited. In *Tropical and idempotent mathematics and applications*, volume 616 of *Contemp. Math.*, pages 1–45. Amer. Math. Soc., Providence, RI, 2014. doi:10.1090/conm/616/12324.

5 Marianne Akian, Stéphane Gaubert, and Louis Rowen. Linear algebra over systems, in preparation.

6 Xavier Allamigeon, Pascal Benchimol, Stéphane Gaubert, and Michael Joswig. Tropicalizing the simplex algorithm. *SIAM J. Discrete Math.*, 29(2):751–795, 2015. doi:10.1137/130936464.

7 Xavier Allamigeon, Pascal Benchimol, Stéphane Gaubert, and Michael Joswig. Log-barrier interior point methods are not strongly polynomial. *SIAM J. Appl. Algebra Geom.*, 2(1):140–178, 2018.

8 Xavier Allamigeon, Uli Fahrenberg, Stéphane Gaubert, Ricardo D. Katz, and Axel Legay. Tropical Fourier-Motzkin elimination, with an application to real-time verification. *International Journal of Algebra and Computation*, 24(05):569–607, 2014. doi:10.1142/S0218196714500258.

9 Xavier Allamigeon, Stéphane Gaubert, and Ricardo D. Katz. Tropical polar cones, hypergraph transversals, and mean payoff games. *Linear Algebra Appl.*, 435(7):1549–1574, 2011. doi:10.1016/j.laa.2011.02.004.

10 Esther M. Arkin and Christos H. Papadimitriou. On negative cycles in mixed graphs. *Oper. Res. Lett.*, 4(3):113–116, 1985. doi:10.1016/0167-6377(85)90013-6.

11 Achim Bachem and Walter Kern. *Linear programming duality: an introduction to oriented matroids.* Berlin: Springer-Verlag, 1992.

12 Matthew Baker and Nathan Bowler. Matroids over hyperfields, 2016. arXiv:1601.01204.

13 Egon Balas. Disjunctive programming. *Ann. Discrete Math.*, 5:3–51, 1979. Discrete optimization (Proc. Adv. Res. Inst. Discrete Optimization and Systems Appl., Banff, Alta., 1977), II.

14 Walter Briec. Some remarks on an idempotent and non-associative convex structure. *J. Convex Anal.*, 22(1):259–289, 2015.

15 Walter Briec and Charles Horvath. $\mathbb{B}$-convexity. *Optimization*, 53(2):103–127, 2004. doi:10.1080/02331930410001695283.

16 Peter Butkovič, Hans Schneider, and Sergei Sergeev. Generators, extremals and bases of max cones. *Linear Algebra Appl.*, 421(2-3):394–406, 2007. doi:10.1016/j.laa.2006.10.004.

17 Sergei Chubanov. A polynomial projection algorithm for linear feasibility problems. *Math. Program.*, 153(2, Ser. A):687–713, 2015. doi:10.1007/s10107-014-0823-8.

18 Guy Cohen, Stéphane Gaubert, Jean-Pierre Quadrat, and Ivan Singer. Max-plus convex sets and functions. In *Idempotent mathematics and mathematical physics*, volume 377 of *Contemp. Math.*, pages 105–129. Amer. Math. Soc., Providence, RI, 2005. doi:10.1090/conm/377/06987.

19 Michele Conforti, Gérard Cornuéjols, and Giacomo Zambelli. *Integer programming.*, volume 271. Cham: Springer, 2014.

20 Mike Develin and Bernd Sturmfels. Tropical convexity. *Doc. Math.*, 9:1–27, 2004.

21 Mike Develin and Josephine Yu. Tropical polytopes and cellular resolutions. *Experiment. Math.*, 16(3):277–291, 2007. URL: `http://projecteuclid.org/euclid.em/1204928529`.

22 Alex Fink and Felipe Rincón. Stiefel tropical linear spaces. *J. Combin. Theory Ser. A*, 135:291–331, 2015. `doi:10.1016/j.jcta.2015.06.001`.

23 Stéphane Gaubert and Ricardo D. Katz. The tropical analogue of polar cones. *Linear Algebra Appl.*, 431(5-7):608–625, 2009. `doi:10.1016/j.laa.2009.03.012`.

24 Stéphane Gaubert and Ricardo D. Katz. Minimal half-spaces and external representation of tropical polyhedra. *J. Algebraic Combin.*, 33(3):325–348, 2011. `doi:10.1007/s10801-010-0246-4`.

25 Dima Grigoriev and Vladimir V. Podolskii. Tropical effective primary and dual Nullstellensätze. *Discrete Comput. Geom.*, 59(3):507–552, 2018. `doi:10.1007/s00454-018-9966-3`.

26 V. A. Gurvich, A. V. Karzanov, and L. G. Khachiyan. Cyclic games and finding minimax mean cycles in digraphs. *Zh. Vychisl. Mat. i Mat. Fiz.*, 28(9):1407–1417, 1439, 1988.

27 Philipp Jell, Claus Scheiderer, and Josephine Yu. Real tropicalization and analytification of semialgebraic sets, 2018. `arXiv:1810.05132`.

28 Michael Joswig and Georg Loho. Weighted digraphs and tropical cones. *Linear Algebra Appl.*, 501:304–343, 2016. `doi:10.1016/j.laa.2016.02.027`.

29 Victor Klee, Richard Ladner, and Rachel Manber. Signsolvability revisited. *Linear Algebra Appl.*, 59:131–157, 1984. `doi:10.1016/0024-3795(84)90164-2`.

30 Marc Krasner. A class of hyperrings and hyperfields. *Internat. J. Math. Math. Sci.*, 6(2):307–311, 1983. `doi:10.1155/S0161171283000265`.

31 Rolf H. Möhring, Martin Skutella, and Frederik Stork. Scheduling with AND/OR precedence constraints. *SIAM J. Comput.*, 33(2):393–415, 2004. `doi:10.1137/S009753970037727X`.

32 Louis Rowen. Algebras with a negation map, 2016. `arXiv:1602.00353`.

33 Sven Schewe. From parity and payoff games to linear programming. In *Mathematical foundations of computer science 2009*, volume 5734 of *Lecture Notes in Comput. Sci.*, pages 675–686. Springer, Berlin, 2009. `doi:10.1007/978-3-642-03816-7_57`.

34 Oleg Viro. Patchworking real algebraic varieties, 2006. `arXiv:math/0611382`.

35 Oleg Viro. Hyperfields for Tropical Geometry I. Hyperfields and dequantization, 2010. `arXiv:1006.3034`.

36 Karel Zimmermann. A general separation theorem in extremal algebras. *Ekonom.-Mat. Obzor*, 13(2):179–201, 1977.

A Missing Proofs

▶ **Lemma 2.6.** *Let* $a, b, c, d \in \mathbb{S}$.

(a) $a \oplus c \vDash b \Leftrightarrow a \vDash b \ominus c$

(b) $a \vDash b \quad \wedge \quad c \vDash d \quad \Rightarrow \quad a \oplus c \vDash b \oplus d$.

(c) *If* $c \in \mathbb{T}_{\pm}$, *then* $b \vDash c$ *and* $c \vDash a$ *imply* $b \vDash a$.

(d) $a \vDash b$ *implies* $c \odot a \oplus d \vDash c \odot b \oplus d$ *for* $c \in \mathbb{T}_{\geq \mathbb{0}}$ *and* $c \odot b \oplus d \vDash c \odot a \oplus d$ *for* $c \in \mathbb{T}_{\leq \mathbb{0}}$.

Proof.

(a) The claim follows directly from the definition and the properties of the semiring $\mathbb{S}$.

(b) Using the definition, we obtain $a \ominus b, c \ominus d \in \mathbb{T}_{\geq \mathbb{0}} \cup \mathbb{T}_{\bullet}$. This implies already $a \oplus c \ominus b \ominus d \in \mathbb{T}_{\geq \mathbb{0}} \cup \mathbb{T}_{\bullet}$.

(c) For a contradiction, assume $b \nvDash a$, that is, $b \ominus a \in \mathbb{T}_{<\mathbb{0}}$.

Case I: $|b| > |a|$. In this case, $b \in \mathbb{T}_{<0}$. Since $b \ominus c \in \mathbb{T}_{\geq 0} \cup \mathbb{T}_\bullet$, it follows that $c \in \mathbb{T}_{<0}$ and $|c| \geq |b|$, using $c \in \mathbb{T}_\pm$. We now get a contradiction to $c \ominus a \in \mathbb{T}_{\geq 0} \cup \mathbb{T}_\bullet$, since $|c| \geq |b| > |a|$.

Case 2: $|a| > |b|$. This case follows by an analogous argument. With $a \in \mathbb{T}_{>0}$, the condition $c \ominus a \in \mathbb{T}_{\geq 0} \cup \mathbb{T}_\bullet$ implies $c \in \mathbb{T}_{>0}$ and $|c| \geq |a| > |b|$. This contradicts $b \ominus c \in \mathbb{T}_{\geq 0} \cup \mathbb{T}_\bullet$.

Case 3: $|a| = |b|$. In this case, we must have $b \in \mathbb{T}_{<0}$ and $a \in \mathbb{T}_{>0}$. We thus obtain $c \in \mathbb{T}_{<0}$ as in case I, but also $c \in \mathbb{T}_{>0}$ as in case II, a contradiction.

(d) The expression $c \odot (a \ominus b) = c \odot a \ominus c \odot b$ is in $\mathbb{T}_{\geq 0}$ for $c \in \mathbb{T}_{\geq 0}$ and in $\mathbb{T}_{\leq 0}$ for $c \in \mathbb{T}_{\leq 0}$. Now, the statement follows from (b) with $d \vDash d$. ◀

▶ **Lemma 3.5.**

(a) *Let $a \in \mathbb{S}$, $b \in \mathbb{T}_\pm$, and $z \in \mathcal{U}(a \oplus b)$. Then there exists an $a' \in \mathcal{U}(a)$ such that $z \in \mathcal{U}(a' \oplus b)$.*

(b) *If $a \in \mathcal{U}(x)$, $b \in \mathcal{U}(y)$, and $c \in \mathbb{T}_\pm$, then $\mathcal{U}(c \odot a \oplus b) \subseteq \mathcal{U}(c \odot x \oplus y)$.*

Proof.

(a) If $a \in \mathbb{T}_\pm$, then $a' = a$ satisfies the requirements. For the rest of the proof, we assume $a \in \mathbb{T}_\bullet$. If $|b| > |a|$, then we can set $a' = |a|$. In this case, $a' \oplus b = a \oplus b = b \in \mathbb{T}_\pm$. Consider now the case $|a| \geq |b|$, which implies $a \oplus b = a$. Then $z \in \mathcal{U}(a \oplus b)$ if and only if $|z| \leq |a|$. For $|a| \geq |z| > |b|$, we set $a' = z$. If $|a| \geq |b| \geq |z|$, then we set $a' = \ominus b$. In both cases it is easy to see that $z \in \mathcal{U}(a' \oplus b)$.

(b) Note that $|a| \leq |x|$ and $|b| \leq |y|$, and consequently, $|c \odot a \oplus b| \leq |c \odot x \oplus y|$. If $c \odot x \oplus y$ is balanced, then the claim follows: $\mathcal{U}(c \odot x \oplus y)$ contains all $r \in \mathbb{T}_\pm$ with $|r| \leq |c \odot x \oplus y|$; this holds for all $r \in \mathcal{U}(c \odot a \oplus b)$.

Hence, assume that $c \odot x \oplus y$ is not balanced. In particular, x or y is not balanced. If both $x, y \in \mathbb{T}_\pm$, then $a = x$ and $b = y$ and thus the claim is immediate. The remaining case is when exactly one of x and y is balanced. Let us assume $y \in \mathbb{T}_\pm$; the case $x \in \mathbb{T}_\pm$ follows similarly. Now we have $b = y$, and we must also have $|y| > |c \odot x|$ as otherwise $c \odot x \oplus y$ would be balanced. Consequently, $c \odot x \oplus y = y$. On the other hand, $|a| \leq |x|$ and $b = y$ imply $|c \odot a| < |b|$, and therefore $c \odot a \oplus b = y$, and the claim follows. ◀

▶ **Proposition 3.6.** *An arbitrary subset $M \subseteq \mathbb{T}_\pm^d$ is tropically convex if and only if the tropical convex hull $\operatorname{tconv}(\{p, q\})$ is contained in M for all $p, q \in M$.*

Proof. For a tropically convex set, the tropical convex hull of all two-element subsets is contained by definition. In the converse direction, we show by induction on n that if we select any n vectors from M as the columns of a matrix $A \in \mathbb{T}_\pm^{d \times n}$, then $\mathcal{U}(A \odot x) \subseteq M$ for any $x \in \mathbb{T}_{\geq 0}^n$, $\bigoplus_{j \in [n]} x_j = 0$. The case $n = 2$ follows by the assumption; consider now $n \geq 3$ and assume that the claim holds for $n - 1$.

Let $z \in \mathcal{U}(A \odot x)$. Without loss of generality, we can assume that $x_1 = 0$. We set $s = \bigoplus_{\ell=1}^{n-1} x_\ell \odot a^{(\ell)} \in \mathbb{S}^d$, where $a^{(\ell)}$ is the ℓ-th column of A. We let $q = a^{(n)}$. Then, $A \odot x = s \oplus x_n \odot q$.

We can apply Lemma 3.5(a) to each component of z, s, and $x_n \odot q$. Thus, we obtain a vector $p \in \mathcal{U}(s)$ such that $z \in \mathcal{U}(p \oplus x_n \odot q)$. By induction, $p \in M$, and thus $z \in \operatorname{tconv}(\{p, q\}) \subseteq M$ by the assumption. This completes the proof. ◀

▶ **Proposition 3.7.** *For any matrix $A \in \mathbb{T}_\pm^{d \times n}$, the convex hull $\operatorname{tconv}(A)$ is tropically convex. Consequently, $\operatorname{tconv}(\operatorname{tconv}(A)) = \operatorname{tconv}(A)$.*

Proof. Using Proposition 3.6, it suffices to show that if $p, q \in \operatorname{tconv}(A)$, $\lambda \in \mathbb{T}_{\geq \mathbb{0}}$, $\lambda \leq 0$, then $\mathcal{U}(p \oplus \lambda \odot q) \subseteq \operatorname{tconv}(A)$.

Let $x, y \in \mathbb{T}^n_{\geq \mathbb{0}}$, $\bigoplus_{j\in[n]} x_j = 0$, $\bigoplus_{j\in[n]} y_j = 0$ such that $p \in \mathcal{U}(A \odot x)$ and $q \in \mathcal{U}(A \odot y)$. We let $z = x \oplus \lambda \odot q$; clearly, $z \in \mathbb{T}^n_{\geq \mathbb{0}}$ and $\bigoplus_{j\in[n]} z_j = 0$. From Lemma 3.5(b), we obtain that

$$\mathcal{U}(p \oplus \lambda \odot q) \subseteq \mathcal{U}\left((A \odot x) \oplus \lambda \odot (A \odot y)\right) = \mathcal{U}(A \odot z) \subseteq \operatorname{tconv}(A) \ . \qquad \blacktriangleleft$$

▶ **Proposition 4.1.** *For $A \in \mathbb{T}^{d\times n}_{\pm}$ and $b \in \mathbb{T}^d_{\pm}$ we have*

$$b \in \operatorname{tconv}(A) \Leftrightarrow \mathit{ker}_+ \begin{pmatrix} A & \ominus b \\ 0 & \ominus 0 \end{pmatrix} \neq \emptyset \ .$$

Proof. The condition $b \in \operatorname{tconv}(A)$ is equivalent to the existence of an element $x \in \mathbb{T}^n_{\geq \mathbb{0}}$ with $\bigoplus_{j\in[n]} x_j = 0$ and $A \odot x \bowtie b$.

Let (x, t) be a vector in the non-negative kernel, where $x \in \mathbb{T}^n_{\geq \mathbb{0}}$ and $t \in \mathbb{T}_{\geq \mathbb{0}}$ denotes the last component. First, we claim that $t \neq \mathbb{0}$. Indeed, $t = \mathbb{0}$ would yield $\bigoplus_{j\in[n]} x_j \bowtie \mathbb{0}$, which implies $x_j = \mathbb{0}$ for all $j \in [n]$. Thus, we obtain $(x, t) = \mathbb{0}$, a contradiction. Since $t \neq \mathbb{0}$, we can scale (x, t) such that $t = 0$. In this case, the definition of the kernel gives $A \odot x \ominus b \bowtie \mathbb{0}$ and $\bigoplus_{j\in[n]} x_j \ominus 0 \bowtie \mathbb{0}$. The latter inequality yields $\bigoplus_{j\in[n]} x_j = 0$. This is the same as the combination witnessing $b \in \operatorname{tconv}(A)$ as above. $\blacktriangleleft$

▶ **Theorem 3.14.** *The signed hull* $\operatorname{tconv}(A)$ *is the union of the signed valuations for all possible lifts*

$$\operatorname{tconv} A = \bigcup_{\operatorname{sval}(\mathbf{A})=A} \operatorname{sval}(\operatorname{conv}(\mathbf{A})) \ .$$

Proof. We start with the inclusion "$\supseteq$". Let $A \in \mathbb{T}^{d\times n}_{\pm}$ and fix a lift $\mathbf{A}$ of A, this means a matrix $\mathbf{A} \in \mathbb{K}^{d\times n}$ with $\operatorname{sval}(\mathbf{A}) = A$. For a vector $\lambda \in \mathbb{K}_{\geq 0}$ with $\sum_{j=1}^n \lambda_i = 1$ the valuation $x = \operatorname{sval}(\lambda)$ is in $\mathbb{T}^n_{\geq \mathbb{0}}$ and fulfills $\bigoplus_{j=1}^n x_i = 0$. We want to show that $b = \operatorname{sval}(\lambda_1 \cdot \mathbf{a^{(1)}} + \cdots + \lambda_n \cdot \mathbf{a^{(n)}}) \in \operatorname{tconv} A$. For each $i \in [d]$, let $c_i = \max\left\{ |\operatorname{sval}(\lambda_j \cdot \mathbf{a_i^{(j)}})| \;\middle|\; j \in [n] \right\}$. Furthermore, we define $p = A \odot x = \bigoplus_{j\in[n]} x_j \odot a^{(j)} = \bigoplus_{j\in[n]} \operatorname{sval}(\lambda_j \cdot \mathbf{a^{(j)}})$. We fix an $i \in [d]$ and we want to show that $b_i \in \mathcal{U}(p_i)$. Note that $|p_i| = c_i$. If p_i is not balanced, we already have $b_i = p_i$. Otherwise, we get $|b_i| \leq c_i$ and consequently $b_i \in [\ominus c_i, c_i]$. This finishes the proof of the inclusion "$\supseteq$".

For the other direction, we fix $b \in \mathcal{U}(A \odot x)$ for some $x \in \mathbb{T}^n_{\geq \mathbb{0}}, \bigoplus_{j\in[n]} x_j = 0$. We define

$$\lambda_j = t^{x_j} \cdot \left(\sum_{k\in[n]} t^{x_k} \right)^{-1} \qquad \text{for each } j \in [n] \ .$$

With this, we get $\lambda \geq 0$ and $\sum_{k\in[n]} \lambda_k = 1$.

For each row $i \in [d]$, we denote by J_i^+ the set of indices of the positive elements in the set $\operatorname{argmax}\left\{ a_j^{(i)} \odot x_j \;\middle|\; j \in [n] \right\}$, and by J_i^- analogously for negative elements.

We set ℓ_i to be an arbitrary index in $\operatorname{argmin}\{|J_i^+|, |J_i^-|\}$, and define

$$\mathbf{a_i^{(j)}} = \begin{cases} \operatorname{tsgn}(a_{ij}) t^{|a_{ij}|} + \alpha_i & \text{for } j = \ell_i \\ \operatorname{tsgn}(a_{ij}) t^{|a_{ij}|} & \text{else} \end{cases} ,$$

where

$$\alpha_i = \frac{1}{\lambda_{\ell_i}} \left(- \sum_{k \in [d]} \operatorname{tsgn}(a_{ik}) t^{|a_{ik}|+x_k} + \operatorname{tsgn}(b_i) t^{|b_i|} \right) \quad .$$

Note that $|b_i| \leq |a_{i\ell_i}| + x_{\ell_i}$ and $|a_{ij}| + x_j - x_{\ell_i} \leq |a_{i\ell_i}|$ for all $j \in [n]$. Therefore, $\operatorname{sval}(\mathbf{a_i^{(j)}}) = a_{ij}$ for all $i \in [d]$ and $j \in [n]$. Furthermore, we get

$$\begin{aligned} \mathbf{a_i} \cdot \lambda &= \sum_{k \in [d] \setminus \{\ell_i\}} \lambda_k \mathbf{a_i^{(k)}} + \lambda_{\ell_i} \mathbf{a_i^{(\ell_i)}} \\ &= \sum_{k \in [d]} \operatorname{tsgn}(a_{ik}) t^{|a_{ik}|+x_k} - \sum_{k \in [d]} \operatorname{tsgn}(a_{ik}) t^{|a_{ik}|+x_k} + \operatorname{tsgn}(b_i) t^{|b_i|} \\ &= \operatorname{tsgn}(b_i) t^{|b_i|} \quad . \end{aligned}$$

Hence, we have $\operatorname{sval}(\mathbf{a_i} \cdot \lambda) = b_i$ for all $i \in [d]$. This concludes the proof. ◀

▶ **Proposition 4.8.** *Let $A, B \subset \mathbb{S}$ be two finite sets. There is an element $c \in \mathbb{S}$ with*

$$c \ominus a > \mathbb{0} \text{ and } b \ominus c > \mathbb{0} \text{ for all } a \in A, b \in B \tag{24}$$

if and only if

$$b \ominus a > \mathbb{0} \text{ for all } (a, b) \in A \times B \quad . \tag{25}$$

Furthermore, the element c can chosen to be signed.

Proof. For each pair $(a, b) \in A \times B$, we add the inequalities in (24) using Lemma 2.2 and obtain $b \ominus a \oplus c \ominus c > \mathbb{0}$. This implies $b \ominus a > |c| \geq \mathbb{0}$ and, hence, (25).

For the other direction, note that $A \subset \mathbb{T}_\pm$ or $B \subset \mathbb{T}_\pm$, as two balanced elements are not comparable by ">". Because of the symmetry (9), we can assume that $B \subset \mathbb{T}_\pm$. Let β denote the minimum of B. Furthermore, we define α as the maximum of $A \cap \mathbb{T}_\pm$ and $\{|a| \mid a \in A \cap \mathbb{T}_\bullet\}$, where either of these two sets could also be empty. We obtain from (25) that $\beta > \alpha$, where we use that $b > a \Leftrightarrow b > |a|$ for $a \in \mathbb{T}_\bullet$. An arbitrary element c with $\alpha < c < \beta$ fulfills (24). As the elements in B are totally ordered, the claim for the inequalities involving B follows immediately. Distinguishing the balanced and signed elements yields the claim for the inequalities involving A. ◀

▶ **Proposition 4.10.** *Let $A, B \subset \mathbb{S}$ be two finite sets. There is an element $c \in \mathbb{T}_\pm$ with*

$$c \ominus a \vDash \mathbb{0} \text{ and } b \ominus c \vDash \mathbb{0} \text{ for all } a \in A, b \in B \tag{26}$$

if and only if

$$b \ominus a \vDash \mathbb{0} \text{ for all } (a, b) \in A \times B \quad . \tag{27}$$

Proof. The first direction from (26) to (27) follows from Lemma 2.6(a) and 2.6(c) because of $c \in \mathbb{T}_\pm$.

For the other direction, let

$$\begin{aligned} \alpha_0 &= \operatorname{argmin} \{ |a| \mid a \in A \cap \mathbb{T}_\bullet \} \quad , \\ \alpha_1 &= \max \{ a \mid a \in A \cap \mathbb{T}_\pm \} \quad , \\ \beta_0 &= \operatorname{argmin} \{ |b| \mid b \in B \cap \mathbb{T}_\bullet \} \quad , \\ \beta_1 &= \min \{ b \mid b \in B \cap \mathbb{T}_\pm \} \quad , \end{aligned} \tag{54}$$

with respect to the ordering "$\geq$". By construction, we get from (27) the relation $\beta_1 \ominus \alpha_1 \vDash \mathbb{0}$, which yields $\beta_1 \geq \alpha_1$. Furthermore, we obtain $\beta_1 \ominus \alpha_0 \vDash \mathbb{0}$ implying $\beta_1 \geq \ominus|\alpha_0|$ and $\beta_0 \ominus \alpha_1 \vDash \mathbb{0}$ implying $|\beta_0| \geq \alpha_1$. We conclude that

$$\underline{\beta} := \min(|\beta_0|, \beta_1) \geq \max(\ominus|\alpha_0|, \alpha_1) =: \overline{\alpha} \ ,$$

using also the trivial inequality $|\beta_0| \geq \ominus|\alpha_0|$. Let γ be an arbitrary element in the interval

$$\left\{ x \in \mathbb{T}_\pm \;\middle|\; \underline{\beta} \geq x \geq \overline{\alpha} \right\} \neq \emptyset.$$

By checking all possibilities arising from the list in (54), we see that the element γ fulfills $b \ominus \gamma \vDash \mathbb{0}$ and $\gamma \ominus a \vDash \mathbb{0}$ for all $a \in A, b \in B$. ◀

B Other Notions of Tropical Convexity

Parallel to the development of tropical convexity, the more general notion of $\mathbb{B}$-convexity was developed starting with [15]. The notion of $\mathbb{B}$-convexity boils down to convexity defined over the semiring $\mathbb{R}_{\geq 0}$ with operations "$\oplus$" = "max" and "$\odot$" = "$\cdot$", see [15, Theorem 2.1.1]. Taking logarithms transforms these operations to "$\oplus$" = "max" and "$\odot$" = "$+$" on $\mathbb{R} \cup \{-\infty\}$. This gives rise to a transferred version of $\mathbb{B}$-convexity on $\mathbb{T}_\pm$ by considering the images of $\mathbb{B}$-convex sets in $\mathbb{R}^d$ under the map $\operatorname{slog}\colon x \mapsto \operatorname{sgn}(x)\log(|x|)$.

The following example shows that our notion of signed tropical convexity is an even more restrictive notion than $\mathbb{B}$-convexity and $\mathbb{B}^\sharp$-convex [14].

▶ **Example B.1.** The tropical convex hull of $A = \{(\ominus 2, \ominus 1), (2, 1)\}$ is the set

$$[\ominus 2, 2] \times [\ominus 1, 1] \ .$$

However, the set $\operatorname{Co}^r(A)$ is

$$L = \{\, (2 \odot \lambda, \lambda) \mid \lambda \in [\ominus 1, 1] \} \ .$$

for all $r \in \mathbb{N}$. In particular, also $\operatorname{Co}^\infty(A)$ equals L. This implies that $\mathbb{B}(L) = L$. We depict both in Figure 7. Hence, $\operatorname{tconv}(A)$ strictly contains $\mathbb{B}(A)$. Furthermore, [14, Corollary 4.2.4] shows that L is also $\mathbb{B}^\sharp$-convex.

Interestingly, the set L is also the image under the signed valuation of the set

$$\operatorname{conv}\left(\begin{pmatrix} -t^2 \\ -t \end{pmatrix}, \begin{pmatrix} t^2 \\ t \end{pmatrix} \right) \ .$$

Here, we mean the convex hull over the Puiseux series $R\{\!\{t\}\!\}$. So L is the tropicalization of a single line segment while our hull construction yields the union of line segments whose spanning points tropicalize to A, as we saw in Section 3.2. For example, we get the set $\{\ominus 2\} \times [\ominus 1, 1] \cup [\ominus 2, 2] \times \{1\}$ as the tropicalization of $\operatorname{conv}\left(\begin{pmatrix} -2t^2 \\ -t \end{pmatrix}, \begin{pmatrix} t^2 \\ 2t \end{pmatrix} \right)$.

▶ Remark B.2. It is tempting to define a *cancellative sum* for two numbers $a, b \in \mathbb{T}_\pm$ by

$$a \overline{\oplus} b = \begin{cases} a & |a| > |b| \\ b & |b| > |a| \\ a & a = b \\ \mathbb{0} & a = \ominus b \end{cases} \ .$$

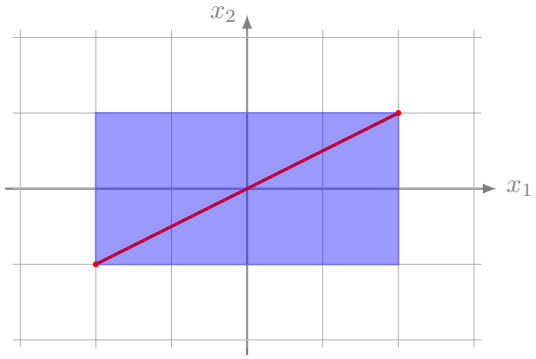


Figure 7 Distinction between $\mathbb{B}$-convex line and tropical line segment through the origin.

This can be extended componentwise to $\mathbb{T}_{\pm}^{d}$.

An iterative version of this construction is used in [14]. A conceptional drawback of the cancellative sum is that it is not associative, as the example

$$0 \,\overline{\oplus}(\ominus 0 \,\overline{\oplus}\, -1) = 0 \,\overline{\oplus} \ominus 0 = \mathbb{0} \neq -1 = \mathbb{0} \,\overline{\oplus}\, -1 = (0 \,\overline{\oplus} \ominus 0) \,\overline{\oplus}\, -1$$

shows.

▶ Remark B.3. Recall that Theorem 6.4 gave a way to determine the whole tropical convex hull from the tropical convex hull in each orthant. For sufficiently generic matrices, one can use the cancellative sum from Remark B.2. If there are no antipodal points, no balanced coefficients arise in the elimination and the map ξ used for Theorem 4.12 is just the identity. Hence, the iterative construction of a single intersection point with a coordinate hyperplane suffices.

Distributional Property Testing in a Quantum World

András Gilyén
QuSoft, CWI and University of Amsterdam, The Netherlands
gilyen@cwi.nl

Tongyang Li
Department of Computer Science, Institute for Advanced Computer Studies,
University of Maryland, USA
Joint Center for Quantum Information and Computer Science,
University of Maryland, USA
tongyang@cs.umd.edu

Abstract

A fundamental problem in statistics and learning theory is to test properties of distributions. We show that quantum computers can solve such problems with significant speed-ups. We also introduce a novel access model for quantum distributions, enabling the coherent preparation of quantum samples, and propose a general framework that can naturally handle both classical and quantum distributions in a unified manner. Our framework generalizes and improves previous quantum algorithms for testing closeness between unknown distributions, testing independence between two distributions, and estimating the Shannon / von Neumann entropy of distributions. For classical distributions our algorithms significantly improve the precision dependence of some earlier results. We also show that in our framework procedures for classical distributions can be directly lifted to the more general case of quantum distributions, and thus obtain the first speed-ups for testing properties of density operators that can be accessed coherently rather than only via sampling.

2012 ACM Subject Classification Mathematics of computing → Distribution functions; Theory of computation → Algorithm design techniques; Theory of computation → Quantum query complexity

Keywords and phrases distributional property testing, quantum algorithms, quantum query complexity

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.25

Related Version The arXiv version of the paper is available at `https://arxiv.org/abs/1902.00814`.

Funding *András Gilyén*: Supported by ERC Consolidator Grant QPROGRESS and partially supported by QuantERA project QuantAlgo 680-91-034.
Tongyang Li: Supported by IBM PhD Fellowship, QISE-NET Triplet Award (NSF DMR-1747426), and the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Quantum Algorithms Teams program.

Acknowledgements A.G. thanks Ronald de Wolf, Ignacio Cirac and Yimin Ge for useful discussion. T.L. thanks Xiaodi Wu and Nengkun Yu for useful discussion.

1 Introduction

Distributional property testing is a fundamental problem in theoretical computer science (see, e.g. [22]) aiming at determining properties of probability distributions with the least number of independent samples. It has intimate connections and applications to statistics, learning theory, and algorithm design.

The merit of distributional property testing mainly comes from the fact that the testing of many properties admits *sublinear* algorithms. For instance, given the ability to take samples from a discrete distribution p on $[n] := \{1, \ldots, n\}$, it requires $\Theta(n/\epsilon^2)$ samples to "learn" p,

i.e., to construct a distribution q on $[n]$ such that $\|p-q\|_1 \leq \epsilon$ with success probability at least $2/3$ ($\|\cdot\|_1$ being ℓ^1-distance). However, testing whether $p=q$ or $\|p-q\|_1 > \epsilon$ requires only $\Theta(\max\{\frac{n^{2/3}}{\epsilon^{4/3}}, \frac{n^{1/2}}{\epsilon^2}\})$ samples from p and q [19], which is sublinear in n and significantly smaller than the complexity of learning the entire distributions. See Section 1.4 for more examples and discussions.

In this paper, we study the impact of quantum computation on distributional property testing problems. We are motivated by the emerging topic of "quantum property testing" (see the survey of [30]) which focuses on investigating the quantum advantage in testing classical statistical properties. Quantum speed-ups have already been established for a few specific problems such as testing closeness between distributions [13, 29], testing identity to known distributions [18], estimating entropies [27], etc. In this paper we propose a generic approach for quantum distributional property testing, and illustrate its power on a few examples. This is our attempt to make progress on the question:

Can quantum computers test properties of distributions systematically and more efficiently?

1.1 Problem statements

Throughout the paper, we denote probability distributions on $[n]$ by p and q; their ℓ^α-distance is defined as $\|p-q\|_\alpha := (\sum_{i=1}^n |p_i - q_i|^\alpha)^{\frac{1}{\alpha}}$. Similarly, we denote $n \times n$ density operators[1] (i.e., quantum distributions) by ρ and σ; their ℓ^α-distance is defined via the corresponding Schatten norm.

Input models. To formulate the problems we address, we define classical and quantum access models for distributions on $[n]$. We begin with the very natural model of sampling.

▶ **Definition 1** (Sampling). *A classical distribution $(p_i)_{i=1}^n$ is accessible via* classical sampling *if we can request independent samples from the distribution, i.e., get a random $i \in [n]$ with probability p_i. A quantum distribution $\rho \in \mathbb{C}^{n\times n}$ is accessible via* quantum sampling *if we can request independent copies of the state ρ.*

Now we define a coherent analogue of the above sampling model. To our knowledge this type of query-access was only studied by a few earlier works [29, 24] and only in the special case of classical distributions. The motivation for this input model is the following: we can think about a density operator as the outcome of some physical process modeled by some black-box. Suppose that the black-box can generate samples on demand. Unlike in the classical (randomized) setting, in a quantum scenario in principle it is always possible to reverse every computational / physical process – including this black-box. If reversion is not feasible, then we get the plain sampling model; however if it is possible to reverse the (quantum) black-box then we get the purified query access model that we describe. For example, if a quantum computer produces the samples via, say, a Monte Carlo method, then the process is easily reversible. However, if the samples come from some source "outside the lab", then reversing the process might not be possible. Therefore, both input models (purified quantum query access and sampling access) are well-motivated. The surprising fact is that this subtle difference in the input models gives rise to significantly different complexities, as we show later for several problems.

[1] For readers less familiar with quantum computing, a density operator (=quantum distribution) $\rho \in \mathbb{C}^{n\times n}$ is a positive semidefinite matrix with $\mathrm{Tr}[\rho] = 1$. Please refer to the textbook [31] for more information.

▶ **Definition 2** (Purified quantum query-access). *A density operator $\rho \in \mathbb{C}^{n\times n}$, has* purified quantum query-access *if we have access to a unitary oracle U_ρ (and its inverse) acting as*[2]

$$U_\rho|0\rangle_A|0\rangle_B = |\psi_\rho\rangle_{AB} = \sum_{i=1}^{n} \sqrt{p_i}|\phi_i\rangle_A|\psi_i\rangle_B, \text{ (where } \langle\phi_i|\phi_j\rangle = \langle\psi_i|\psi_j\rangle = \text{(Kronecker) } \delta_{ij})$$

such that $\mathrm{Tr}_A(|\psi_\rho\rangle\langle\psi_\rho|) = \rho$. If $|\psi_i\rangle = |i\rangle$, then $\rho = \sum_i^n p_i|i\rangle\langle i|$ is a diagonal density operator which can be identified with the classical distribution p, so we can simply write U_p instead of U_ρ. With a slight abuse of notation sometimes we will concisely write $|\rho\rangle$ instead of $|\psi_\rho\rangle$.

We also define an even stronger input model that is considered in a series of earlier works, see, e.g., [13, 18, 27, 15].

▶ **Definition 3** (Classical distribution with discrete query-access). *A classical distribution $(p_i)_{i=1}^n$, has* discrete query-access *if we have classical / quantum query-access to a function $f\colon S \to [n]$ such that for all $i \in [n]$, $p_i = |\{s \in [S] : f(s) = i\}|/|S|$. (Typically the interesting regime is when $|S| \gg n$.) In the quantum case a query oracle is a unitary operator O acting on $\mathbb{C}^{|S|} \otimes \mathbb{C}^n$ as*

$$\mathrm{O}\colon |s, 0\rangle \leftrightarrow |s, f(s)\rangle \text{ for all } s \in S.$$

Note that if one first creates a uniform superposition over S and then makes a query, then the above oracle turns into a purified query oracle to a classical distribution as in Definition 2. Therefore all lower bounds that are proven in this model also apply to the purified query-access oracles. In fact all algorithms that the authors are aware of do this conversion, so they effectively work in the purified query-access model. Moreover, we conjecture that the two input models are equivalent when $|S| \gg n$. For this reason we only work with the purified query-access model in this work.

Another strengthening of the purified query-access model for classical distributions is when we assume access to a unitary (and its inverse) acting as $|0\rangle \mapsto \sum_{i=1}^n \sqrt{p_i}|i\rangle$. A very similar input model was thoroughly studied by [4].

▶ **Definition 4** (Classical distribution with pure-state preparation access). *A classical distribution $(p_i)_{i=1}^n$, is accessible via* pure state preparation oracle *if we have access to a unitary oracle U_{pure} (and its inverse) acting as*

$$U_{\mathrm{pure}}\colon |0\rangle \mapsto \sum_{i=1}^{n} \sqrt{p_i}|i\rangle.$$

This is strictly stronger[3] than the purified query-access model. In order to simulate purified queries we can first do a pure state query and then copy $|i\rangle$ to a second fresh ancillary

[2] $|\psi\rangle \in \mathbb{C}^n$ denotes a "ket" vector and $\langle\psi| = (|\psi\rangle)^\dagger$ stands for its conjugate transpose, called "bra" in Dirac notation; $|i\rangle = \vec{e}_i$ is the i^{th} basis vector. An ℓ^2-normalized $|\psi\rangle$ is called a pure state, and corresponds to density operator $|\psi\rangle\langle\psi|$. For $A = \mathbb{C}^k, B = \mathbb{C}^n$ and $|\phi\rangle \in A\otimes B$ we denote by $\mathrm{Tr}[|\phi\rangle\langle\phi|]_A \in B\otimes B^* = \mathbb{C}^{n\times n}$ the partial trace over A.

[3] This can be seen in various ways. We give an argument in the spirit of distributional property testing. Closeness of two unknown distributions p, q can be tolerantly tested in the squared Hellinger distance $H(p,q)^2 = \frac{1}{2}\left\|\sqrt{p} - \sqrt{q}\right\|_2^2$ to precision ϵ in query complexity $\mathcal{O}\left(1/\sqrt{\epsilon}\right)$ in the model of Definition 4 using amplitude estimation. On the other hand the classical sample complexity of testing equality to ϵ precision in this metric is $\widetilde{\Theta}(\min(n^{2/3}/\epsilon^{4/3}, n^{3/4}/\epsilon))$, as shown in [20]. The results of Chailloux [16] imply that this query complexity improves at most cubically in the model of Definition 2, showing that the input model of Definition 4 is strictly stronger.

register using, e.g., some CNOT gates. Finally, for completeness we mention that one could also consider a model similar to the above where one can only request samples of pure states of the form $\sum_{i=1}^{n} \sqrt{p_i}|i\rangle$, as studied for example in [6, 5].

We will mostly focus on the first two input models and will only use the latter strengthenings of the purified query-access model for invoking and proving lower bounds.

Property testing problems. We study three distributional properties: ℓ^α-closeness testing, independence testing, and entropy estimation. In the classical literature these are well-studied properties, and the corresponding testers motivate general algorithms for testing properties of discrete distributions [20, 1].

For brevity we only give the definitions for classical distributions; similar definitions apply to quantum density matrices if we replace vector norms by the corresponding Schatten norms.

▶ **Definition 5** (ℓ^α-closeness testing). *Given $\epsilon > 0$ and two probability distributions p, q on $[n]$,* ℓ^α-closeness testing *is to decide whether $p = q$ or $\|p-q\|_\alpha \geq \epsilon$ with success probability at least $\frac{2}{3}$. Tolerant testing: decide whether $\|p-q\|_\alpha \leq 0.99\epsilon$ or $\|p-q\|_\alpha \geq \epsilon$ with success probability at least $\frac{2}{3}$.*

▶ **Definition 6** (Independence testing). *Given $\epsilon > 0$ and a probability distribution p on $[n] \times [m]$ with $n \geq m$,* independence testing *is to decide, with success probability at least $\frac{2}{3}$, whether p is a product distribution or p is ϵ-far in ℓ^1-norm from any product distribution on $[n] \times [m]$.*

▶ **Definition 7** (Entropy estimation). *Given $\epsilon > 0$ and a density operator $\rho \in \mathbb{C}^{n \times n}$,* entropy estimation *is to estimate the Shannon / von Neumann entropy $H(\rho) = -\mathrm{Tr}[\rho \log(\rho)]$ within additive ϵ-precision with success probability at least $\frac{2}{3}$.*

1.2 New results

We give a systematic study of distributional property testing for classical / quantum distributions, and obtain the following results for the purified quantum query model of Definition 2:

- Entropy estimation of classical / quantum distributions costs $\widetilde{\mathcal{O}}\left(\frac{\sqrt{n}}{\epsilon^{1.5}}\right)$ and $\widetilde{\mathcal{O}}\left(\frac{n}{\epsilon^{1.5}}\right)$ queries respectively,[4] as we prove in Theorem 12 and Theorem 13.
- Tolerant ℓ^2-closeness testing of classical / quantum distributions costs $\tilde{\Theta}\left(\frac{1}{\epsilon}\right)$ and $\mathcal{O}\left(\min\left(\frac{\sqrt{n}}{\epsilon}, \frac{1}{\epsilon^2}\right)\right)$ queries respectively, as we prove in Theorem 14 and Theorem 15.
- ℓ^1-closeness testing of classical / quantum distributions costs $\widetilde{\mathcal{O}}\left(\frac{\sqrt{n}}{\epsilon}\right)$ and $\mathcal{O}\left(\frac{n}{\epsilon}\right)$ queries respectively, as we prove in Corollary 17.
- Independence testing of classical / quantum distributions costs $\widetilde{\mathcal{O}}\left(\frac{\sqrt{nm}}{\epsilon}\right)$ and $\mathcal{O}\left(\frac{nm}{\epsilon}\right)$ queries respectively, as we prove in Corollary 18.

For context, we compare our results with previous classical and quantum results in Table 1 and Table 2. (Note that all of our results are gate efficient, because they are based on singular value transformation and amplitude estimation, both of which have gate-efficient implementations.)

---

[4] Throughout the paper, we use $\widetilde{\mathcal{O}}$ to omit poly-logarithmic factors in $\mathcal{O}$, i.e., $\widetilde{\mathcal{O}}(T) = \mathcal{O}(T \,\mathrm{poly}(\log T))$.
[5] Recent results of [16] imply that in this model quantum speed-ups are at most cubic.

Table 1 Summary of sample and query complexity results. Our new bounds are printed in **bold**. For classical distributions with quantum query-access[5] we prove (almost) matching upper and lower bounds for ℓ^2-testing, and improve the previous best complexity $\widetilde{\mathcal{O}}\left(\sqrt{n}/\epsilon^{2.5}\right)$ for ℓ^1-testing by [29] and $\widetilde{\mathcal{O}}\left(\sqrt{n}/\epsilon^{2}\right)$ for Shannon entropy estimation by [27]. We are not aware of prior work on testing quantum distributions with purified query-access.

problem / model	ℓ^1-closeness testing	(tolerant) ℓ^2-closeness testing	Shannon / von Neumann entropy
Classical distribution sampling	$\Theta\left(\max\left\{\frac{n^{2/3}}{\epsilon^{4/3}}, \frac{n^{1/2}}{\epsilon^2}\right\}\right)$ [19]	$\Theta\left(\frac{1}{\epsilon^2}\right)$ [19]	$\Theta\left(\frac{n}{\epsilon \log n} + \frac{\log^2 n}{\epsilon^2}\right)$ [25, 42]
Classical distribution with purified query-access	$\widetilde{\boldsymbol{\mathcal{O}}}\left(\frac{\sqrt{n}}{\epsilon}\right)$	$\widetilde{\boldsymbol{\Theta}}\left(\frac{1}{\epsilon}\right)$	$\widetilde{\boldsymbol{\mathcal{O}}}\left(\frac{\sqrt{n}}{\epsilon^{1.5}}\right)$; $\tilde{\Omega}(\sqrt{n})$ [15]
Quantum state with purified query-access	$\boldsymbol{\mathcal{O}}\left(\frac{n}{\epsilon}\right)$	$\boldsymbol{\mathcal{O}}\left(\mathbf{min}\left(\frac{\sqrt{n}}{\epsilon}, \frac{1}{\epsilon^2}\right)\right)$	$\widetilde{\boldsymbol{\mathcal{O}}}\left(\frac{n}{\epsilon^{1.5}}\right)$
Quantum state sampling	$\Theta\left(\frac{n}{\epsilon^2}\right)$ [7]	$\Theta\left(\frac{1}{\epsilon^2}\right)$ [7]	$\mathcal{O}\left(\frac{n^2}{\epsilon^2}\right)$, $\Omega\left(\frac{n^2}{\epsilon}\right)$ [3]

As we show our quantum algorithms for classical distributional property testing problems with purified access can be naturally lifted to the case of quantum distributions, incurring an overhead of $\approx \sqrt{n}$, which is manifested in the complexities of Table 1.

Table 2 Complexities of Shannon / von Neumann entropy estimation with constant precision. It seems that the n-dependence is roughly quadratically higher for quantum distributions, while coherent quantum access gives a quadratic advantage for both classical and quantum distributions. This suggests that our entropy estimation algorithm has essentially optimal n-dependence for density operators with purified access, however we do not have a matching lower bound yet.

	Sample complexity	(Purified) Query complexity
Classical	$\Theta\left(\frac{n}{\log n}\right)$ [40]	$\widetilde{\Theta}\left(\sqrt{n}\right)$ [27, 15]
Quantum	$\Theta\left(n^2\right)$ [3]	$\widetilde{\boldsymbol{\mathcal{O}}}(\boldsymbol{n})$

1.3 New techniques

The motivating idea behind our approach is that if we can prepare a purification of a quantum distribution / density operator ρ, then we can construct a unitary U, which has this density operator in the top-left corner, using only two queries to U_ρ. This observation is due to [28]. We call such a unitary a *block-encoding* of ρ:

$$U = \begin{bmatrix} \rho & \cdot \\ \cdot & \cdot \end{bmatrix} \quad \Longleftrightarrow \quad \rho = \left(\langle 0|^{\otimes a} \otimes I\right) U \left(|0\rangle^{\otimes a} \otimes I\right).$$

One can think of a block-encoding as a post-selective implementation of the linear map ρ: given an input state $|\psi\rangle$, applying the unitary U to the state $|0\rangle^{\otimes a}|\psi\rangle$, measuring the first a-qubit register and post-selecting on the $|0\rangle^{\otimes a}$ outcome, we get a state $\propto \rho|\psi\rangle$ in the second register. Block-encodings are easy to work with, for example given a block-encoding of ρ and σ we can easily construct a block-encoding of $(\rho - \sigma)/2$, see for example in the work of [17].

Example application to ℓ^3-testing. The problem is to decide whether $\rho = \sigma$ or $\|\rho - \sigma\|_3 \geq \epsilon$, with query complexity $\mathcal{O}\left(\epsilon^{-\frac{3}{2}}\right)$. The first idea is that if we can prepare a purification of ρ and σ, then we can also prepare a purification of $(\rho + \sigma)/2$ by setting a qubit to the state

$(|0\rangle + |1\rangle)/\sqrt{2}$, and then controlled on the $|0\rangle$ or $|1\rangle$ value of the qubit run the process that samples from ρ or σ, respectively. The second idea is to combine the block-encodings of ρ and σ to apply the map $\frac{\rho-\sigma}{2}$ to the purification of $(\rho+\sigma)/2$, to get

$$\left|\frac{\rho+\sigma}{2}\right\rangle \mapsto \left(\frac{\rho-\sigma}{2}\otimes I\right)\left|\frac{\rho+\sigma}{2}\right\rangle|0\rangle + \ldots|1\rangle.$$

Finally, apply amplitude estimation with setting $M = \Theta(\epsilon^{-\frac{3}{2}})$. This works since if $\|\rho-\sigma\|_3 \geq \epsilon$, then the $|0\rangle$ ancilla state has probability $\mathrm{Tr}\big[(\rho-\sigma)^2(\rho+\sigma)\big]/8 \geq \mathrm{Tr}\big[|\rho-\sigma|^3\big]/8 \geq \epsilon^3/8$.

Working with singular values. The above is a promising approach because it directly makes the density operator in question operationally accessible. However, it turns out that using this simple block-encodings is often suboptimal for distribution testing, because a query in some sense gives access to the square-root of ρ, whereas this unitary has ρ itself in the top-left corner. Since the problems often heavily depend on smaller eigenvalues of ρ, the square root of ρ is more desirable since it has quadratically larger singular/eigenvalues.

One of our main technical contributions is to use a new type of block-encoding, which is a unitary matrix having a certain block proportional to a matrix A such that $A^\dagger A = \rho$, i.e., we use a "square-root" of the (quantum) distribution ρ (in the case of classical distributions rho is a diagonal matrix with the probabilities as diagonal entries). This new technique allows us to develop a unified approach for distributional property testing, which we consider one of our major contributions. It is this new perspective that enables us to derive several results in a relatively short paper. Once we establish this methodology the results are relatively easy to derive in a systematic way, both for classical and quantum distributions.

Therefore, we show how to efficiently construct a unitary matrix whose top-left corner contains a matrix with singular values $\sqrt{p_1}, \ldots, \sqrt{p_n}$, given purified access to a classical distribution p. To be more precise, we define a slight generalization of block-encodings called *projected unitary encodings*, which represent a matrix A in the form of $\Pi U \widetilde{\Pi}$, where $\Pi, \widetilde{\Pi}$ are orthogonal projectors and U is a unitary matrix. One can think about U in a projected unitary encoding as a post-selective implementation of the map $A\colon \mathrm{img}\left(\widetilde{\Pi}\right) \to \mathrm{img}(\Pi)$. Take for example $U := (U_p \otimes I)$, $\Pi := (\sum_{i=1}^n I \otimes |i\rangle\langle i| \otimes |i\rangle\langle i|)$, and $\widetilde{\Pi} := (|0\rangle\langle 0| \otimes |0\rangle\langle 0| \otimes I)$. As we show in Appendix A these operators form a projected unitary encoding of

$$A = \Pi U \widetilde{\Pi} = \sum_{i=1}^{n} \sqrt{p_i}|\phi_i\rangle\langle 0| \otimes |i\rangle\langle 0| \otimes |i\rangle\langle i|. \tag{1}$$

We can use a similar trick for a general density operator ρ too. However, there is a major difficulty which arises from the fact that we do not a prioiri know the diagonalizing basis of ρ. Therefore we use slightly different operators. Let W be a unitary,[6] mapping $|0\rangle|0\rangle \mapsto \sum_{j=1}^n \frac{|j\rangle|j\rangle}{\sqrt{n}}$. Let $U' := \big(I \otimes U_\rho^\dagger\big)\big(W^\dagger \otimes I\big)$, $\Pi' := (I \otimes |0\rangle\langle 0| \otimes |0\rangle\langle 0|)$ and $\widetilde{\Pi}$ as above. As we show in Appendix A these operators form a projected unitary encoding of

$$A' = \Pi' U' \widetilde{\Pi} = \sum_{i=1}^{n} \sqrt{\frac{p_i}{n}}|\phi_i'\rangle\langle 0| \otimes |0\rangle\langle 0| \otimes |0\rangle\langle \psi_i|, \tag{2}$$

where $\sum_{j=1}^n \frac{|\phi_j'\rangle|\phi_j\rangle}{\sqrt{n}} = \sum_{j=1}^n \frac{|j\rangle|j\rangle}{\sqrt{n}}$ is the Schmidt decomposition of the maximally entangled state under the basis $(|\phi_1\rangle, \ldots, |\phi_n\rangle)$.

[6] This unitary is easy to implement, e.g., by using a few Hadamard and CNOT gates.

As we can see, the case of general density operators is less efficient, it only gives operational access to the "square root" of ρ/n. We note that for (approximately) transforming a block-encoding of $A/\sqrt{n}$ to a block-encoding of $A/\mathcal{O}(1)$ it is necessary and sufficient to use the block-encoding of $A/\sqrt{n}$ about $\sqrt{n}$ times [21, Theorems 3&17]. This is essentially the reason for the $\approx\sqrt{n}$ overhead in our quantum algorithms for quantum distributions in Table 1. If the $1/\sqrt{n}$ factor could be directly improved, that would speed up our von Neumann entropy estimation algorithm Theorem 13, which seems unlikely, cf. Table 2. This suggests that it is impossible to obtain a more efficient block-encoding in the general case.

General recipe. We summarize our recipe to distributional property testing as follows.

1. Construct quantum circuit / unitary matrix operationally representing the distribution.
2. Transform the singular values of the corresponding matrix according to a desired function.
3. Apply the resulting map to the purification of the distribution, or another suitable state.
4. Estimate the amplitude of the flagged output state and conclude.

The above general scheme describes our approach to the problems we discuss in this paper. Sometimes it is useful to divide the probabilities / singular values into bins, and fine-tune the algorithm by using the approximate knowledge of the size of the singular values. This divide-and-conquer strategy is at the core of our improved tolerant ℓ^2-closeness tester of Theorem 14.

1.4 Related works on distributional property testing

Classical algorithms. Many distributional property testing problems fall into the category of *closeness testing*, where we are given the ability to take independent samples from two unknown distributions p and q with cardinality n, and the goal is to determine whether they are the same versus significantly different. For ℓ^1*-closeness testing*, which is about testing whether $p=q$ or $\|p-q\|_1\geq\epsilon$, [10] first gave a sublinear algorithm using $\tilde{O}(n^{2/3}/\epsilon^{8/3})$ samples to p and q. The follow-up work by [19] determined the optimal sample complexity as $\Theta(\max\{\frac{n^{2/3}}{\epsilon^{4/3}},\frac{n^{1/2}}{\epsilon^2}\})$; the same paper also gave a tight bound $\Theta(\frac{1}{\epsilon^2})$ for ℓ^2*-closeness testing*.

Besides closeness testing, a similar problem is *identity* testing where one of the distributions, say q, is known and we are given independent samples from the other distribution p. For ℓ^1 *identity testing*, it is known that the sample complexity can be smaller than that of ℓ^1-closeness testing, which was proved by [9] to be $\tilde{O}(\sqrt{n}/\epsilon^4)$ and then [37] gave the tight bound $\Theta(\sqrt{n}/\epsilon^2)$. More recently, Ref. [20] proposed a modular reduction-based approach for distributional property testing problems, which recovered all closeness and identity testing results above. Furthermore, they also studied *independence testing* (see also the previous studies by [9, 26, 2]), i.e., whether a distribution on $[n]\times[m]$ $(n\geq m)$ is a product distribution or at least ϵ-far in ℓ^1-distance from any product distribution, and determined the optimal bound $\Theta(\max\{\frac{n^{2/3}m^{1/3}}{\epsilon^{4/3}},\frac{(nm)^{1/2}}{\epsilon^2}\})$.

Apart from the relationship between distributions, properties of a single distribution also have been extensively studied. One of the most important properties is *Shannon entropy* [39] because it measures for example compressibility. The sample complexity of estimating $H(p)$ within additive error ϵ has been intensively studied [8, 35, 36]; in particular, [40, 41] gave an explicit algorithm for entropy estimation using $\Theta(\frac{n}{\epsilon\log n})$ samples when $\epsilon=\Omega(n^{-0.03})$ and $\epsilon=O(1)$; for the general case [25] and [42] gave the optimal estimator with $\Theta\left(\frac{n}{\epsilon\log n}+\frac{(\log n)^2}{\epsilon^2}\right)$ samples.

Quantum algorithms. The first paper on distributional property testing by quantum algorithms was by [13], which considered classical distributions with discrete quantum query-access (see Definition 3); it gives a quantum query complexity upper bound $O(\sqrt{n}/\epsilon^6)$ for ℓ^1-closeness testing and $O(n^{1/3}/\epsilon^{4/3})$ for identity testing to the uniform distribution on $[n]$. Subsequently, [18] gave an algorithm for identity testing (to an arbitrary known distribution) with $\widetilde{\mathcal{O}}\big(n^{1/3}/\epsilon^5\big)$ queries, and [29] improved the ϵ-dependence of ℓ^1-closeness testing to $\widetilde{\mathcal{O}}\big(\sqrt{n}/\epsilon^{2.5}\big)$. More recently, [27] studied entropy estimation under this model, and gave a quantum algorithm for Shannon entropy estimation with $\widetilde{\mathcal{O}}\big(\sqrt{n}/\epsilon^2\big)$ queries and also sublinear quantum algorithms for estimating Rényi entropies ([38]).

Another type of quantum property testing results ([32, 33, 34, 3, 7, 23]) concern *density matrices*, where the ℓ^1-distance becomes the trace distance and the Shannon entropy becomes the von Neumann entropy. To be more specific, for n-dimensional density matrices, the number of samples needed for ℓ^1 and ℓ^2-closeness testing are $\Theta(n/\epsilon^2)$ and $\Theta(1/\epsilon^2)$ ([7]), respectively. In addition [3] gave upper and lower bounds $\mathcal{O}\big(n^2/\epsilon^2\big), \Omega\big(n^2/\epsilon\big)$ for estimating the von Neumann entropy of an n-dimensional density matrix with accuracy ϵ.

1.5 Organization of the paper

The rest of the paper is organized as follows. In Section 2 we introduce two important quantum algorithmic techniques, amplitude estimation and singular value transformation. We give entropy estimators of classical and quantum distributions in Section 3. In Section 4 we give an (essentially) optimal quantum algorithm for tolerantly testing ℓ^2-closeness of classical distributions, and another efficient tolerant ℓ^2-closeness tester for quantum distributions. Proof details of projected encodings, polynomial approximations for singular value transformation, and corollaries about ℓ^1-closeness and independence testing are deferred to Appendix A, B, and C respectively.

2 Preliminaries

2.1 Amplitude estimation

Classically, given i.i.d. samples of a Bernoulli random variable X with $\mathbb{E}[X] = p$, it takes $\Theta(1/\epsilon^2)$ samples to estimate p within ϵ with high success probability. Quantumly, if we are given a unitary U such that

$$U|0\rangle|0\rangle = \sqrt{p}|0\rangle|\phi\rangle + |0^\perp\rangle, \quad \text{where } \||\phi\rangle\| = 1 \text{ and } (\langle 0| \otimes I)|0^\perp\rangle = 0, \tag{3}$$

then if measure the output state, we get 0 in the first register with probability p. Given access to U we can estimate the value of p quadratically more efficiently than what is possible by sampling:

▶ **Theorem 8** ([12, Theorem 12]). *Given U satisfying (3), the amplitude estimation algorithm outputs $\tilde{p}$ such that $\tilde{p} \in [0, 1]$ and*

$$|\tilde{p} - p| \leq \frac{2\pi\sqrt{p(1-p)}}{M} + \frac{\pi^2}{M^2} \tag{4}$$

with success probability at least $8/\pi^2$, using M calls to U and $U^\dagger$.

In particular, if we take $M = \left\lceil 2\pi\left(\frac{2\sqrt{p}}{\epsilon} + \frac{1}{\sqrt{\epsilon}}\right)\right\rceil = \Theta\left(\frac{\sqrt{p}}{\epsilon} + \frac{1}{\sqrt{\epsilon}}\right)$ in (4), we have

$$|\tilde{p} - p| \leq \frac{2\pi\sqrt{p(1-p)}}{2\pi}\epsilon + \frac{\pi^2}{4\pi^2}\epsilon^2 \leq \frac{\epsilon}{2} + \frac{\epsilon}{4} \leq \epsilon.$$

Therefore, using only $\Theta(1/\epsilon)$ implementations of U and $U^\dagger$, we could get an ϵ-additive approximation of p with success probability at least $8/\pi^2$, which is a quadratic speed-up compared to the classical sample complexity $\Theta(1/\epsilon^2)$. The success probability can be boosted to $1-\nu$ by executing the algorithm for $\Theta(\log 1/\nu)$ times and taking the median of the estimates.

2.2 Quantum singular value transformation

Singular value decomposition (SVD) is one of the most important tools in linear algebra, generalizing eigen-decomposition of Hermitian matrices. Recently, [21] proposed *quantum singular value transformation* which turns out to be very useful for property testing. Mathematically, it is defined as follows:

▶ **Definition 9** (Singular value transformation). *Let $f:\mathbb{R}\to\mathbb{C}$ be an even or odd function. Let $A\in\mathbb{C}^{\tilde{d}\times d}$ have the following singular value decomposition*

$$A=\sum_{i=1}^{d_{\min}}\varsigma_i|\tilde{\psi}_i\rangle\langle\psi_i|,$$

where $d_{\min}:=\min(d,\tilde{d})$. For the function f we define the singular value transform *of A as*

$$f^{(SV)}(A):=\begin{cases}\sum_{i=1}^{d_{\min}}f(\varsigma_i)|\tilde{\psi}_i\rangle\langle\psi_i| & \text{if } f \text{ is odd, and}\\ \sum_{i=1}^{d}f(\varsigma_i)|\psi_i\rangle\langle\psi_i| & \text{if } f \text{ is even, where for } i\in[d]\setminus[d_{\min}] \text{ we define } \varsigma_i:=0.\end{cases}$$

Quantum singular value transformation by real polynomials can be efficiently implemented on a quantum computer as follows:

▶ **Theorem 10** ([21, Corollary 18]). *Let $\mathcal{H}_U$ be a finite-dimensional Hilbert space and let $U,\Pi,\widetilde{\Pi}\in\mathrm{End}(\mathcal{H}_U)$ be linear operators on $\mathcal{H}_U$ such that U is a unitary, and $\Pi,\widetilde{\Pi}$ are orthogonal projectors. Suppose that $P=\sum_{k=0}^{n}a_kx^k\in\mathbb{R}[x]$ is a degree-n polynomial such that*

- *$a_k\neq 0$ only if $k\equiv n \mod 2$, and*
- *for all $x\in[-1,1]$: $|P(x)|\leq 1$.*

Then there exist $\Phi\in\mathbb{R}^n$, such that

$$P^{(SV)}\Big(\widetilde{\Pi}U\Pi\Big)=\begin{cases}\Big(\langle+|\otimes\widetilde{\Pi}\Big)\Big(|0\rangle\langle 0|\otimes U_\Phi+|1\rangle\langle 1|\otimes U_{-\Phi}\Big)\Big(|+\rangle\otimes\Pi\Big) & \text{if } n \text{ is odd, and}\\ \Big(\langle+|\otimes\Pi\Big)\Big(|0\rangle\langle 0|\otimes U_\Phi+|1\rangle\langle 1|\otimes U_{-\Phi}\Big)\Big(|+\rangle\otimes\Pi\Big) & \text{if } n \text{ is even,}\end{cases}$$

where $U_\Phi=e^{i\phi_1(2\widetilde{\Pi}-I)}U\prod_{j=1}^{(n-1)/2}\Big(e^{i\phi_{2j}(2\Pi-I)}U^\dagger e^{i\phi_{2j+1}(2\widetilde{\Pi}-I)}U\Big)$.[7]

Thus for an even or odd polynomial P of degree n, we can apply singular value transformation of the matrix $\widetilde{\Pi}U\Pi$ with n uses of U, $U^\dagger$ and the same number of controlled reflections $I-2\Pi, I-2\widetilde{\Pi}$.

To apply singular value transformation corresponding to our problems, we need low-degree polynomial approximations to the following functions, which we construct in Appendix B.

[7] This is the mathematical form for odd n; even n is defined similarly.

▶ **Lemma 11.** (Polynomial approximations) *Let $\beta \in (0,1]$, $\eta \in (0, \frac{1}{2}]$ and $t \geq 1$. There exists polynomials $\tilde{P}, \tilde{Q}, \tilde{S}$ such that*

- $\forall x \in [\frac{1}{t}, 1]\colon |\tilde{P}(x) - \frac{1}{2tx}| \leq \eta$, *and* $\forall x \in [-1,1]\colon -1 \leq \tilde{P}(x) = \tilde{P}(-x) \leq 1$,
- $\forall x \in [-\frac{1-\beta}{t}, \frac{1-\beta}{t}]\colon |\tilde{Q}(x) - tx| \leq \eta \cdot (tx)$, *and* $\forall x \in [-1,1]\colon \tilde{Q}(x) = -\tilde{Q}(-x) \leq 1$,
- $\forall x \in [\beta, 1]\colon |\tilde{S}(x) - \frac{\ln(1/x)}{2\ln(2/\beta)}| \leq \eta$, *and* $\forall x \in [-1,1]\colon -1 \leq \tilde{S}(x) = \tilde{S}(-x) \leq 1$,

moreover $\deg(\tilde{P}) = \mathcal{O}\Big(t \log\Big(\frac{1}{\eta}\Big)\Big)$, $\deg(\tilde{Q}) = \mathcal{O}\Big(\frac{t}{\beta} \log\Big(\frac{1}{\eta}\Big)\Big)$, *and* $\deg(\tilde{S}) = \mathcal{O}\Big(\frac{1}{\beta} \log\Big(\frac{1}{\eta}\Big)\Big)$.

3 Entropy estimation

3.1 Classical distributions with purified quantum query-access

Recall that we introduced purified quantum query-access in Definition 2. In particular, for a classical distribution p on $[n]$, we are given a unitary U_p acting on $\mathbb{C}^{n \times n}$ such that

$$U_p|0\rangle_A|0\rangle_B = |\psi_p\rangle = \sum_{i=1}^{n} \sqrt{p_i}|\phi_i\rangle_A|i\rangle_B. \tag{5}$$

We use U_p and $U_p^\dagger$ to estimate the Shannon entropy $H(p)$:

▶ **Theorem 12.** *For any $0 < \epsilon < 1$, we can estimate $H(p)$ with accuracy ϵ with success probability at least $2/3$ using $\mathcal{O}\Big(\frac{\sqrt{n}}{\epsilon^{1.5}} \log^{1.5}\big(\frac{n}{\epsilon}\big) \log\Big(\frac{\log n}{\epsilon}\Big)\Big)$ calls to U_p and $U_p^\dagger$.*

Proof. The general idea is to first construct a unitary matrix that has a specific matrix block with singular values $\sqrt{p_1}, \ldots, \sqrt{p_n}$. We use the construction of Eq. (1) and apply singular value transformation (Theorem 10) by a polynomial $\tilde{S}$ constructed in Lemma 11, setting $\eta = \frac{\epsilon}{24\ln(2/\beta)}$ and $\beta = \sqrt{\Delta}$ for $\Delta = \frac{\epsilon}{12n\ln(n/\epsilon)}$. Notice that this Δ satisfies

$$\Delta\Big(\ln\Big(\frac{1}{\Delta}\Big) + 4\ln\Big(\frac{2}{\beta}\Big)\Big) = \frac{\epsilon}{12n\ln(n/\epsilon)} \cdot \ln\frac{16(4n\ln\frac{n}{\epsilon})^3}{\epsilon^3} \leq \frac{\epsilon}{12n\ln(n/\epsilon)} \cdot \ln\frac{n^6}{\epsilon^6} = \frac{\epsilon}{2n}, \tag{6}$$

provided that $\frac{n}{\epsilon} \geq 152$. Note that the polynomial $\tilde{S}$ satisfies both conditions in Theorem 10. Applying the singular value transformed version of the operator (1) to the state $|\psi_p\rangle$ gives

$$|\widetilde{\Psi_p}\rangle = \sum_{i=1}^{n} \sqrt{p_i}\tilde{S}(\sqrt{p_i})|\phi_i\rangle_A|i\rangle_B|0\rangle + \ldots|1\rangle. \tag{7}$$

Preparing $|\widetilde{\Psi_p}\rangle$ costs $\deg\tilde{S} = \mathcal{O}\Big(\frac{1}{\beta}\log\Big(\frac{1}{\eta}\Big)\Big) = \mathcal{O}\Big(\sqrt{\frac{n}{\epsilon}\log\big(\frac{n}{\epsilon}\big)}\log\Big(\frac{\log n}{\epsilon}\Big)\Big)$ uses of U_p and $U_p^\dagger$ and the same number of controlled reflections through $\Pi, \widetilde{\Pi}$. Furthermore, Lemma 11 implies that for all i such that $p_i \geq \Delta$,

$$\Big|\frac{p_i\ln(1/p_i)}{4\ln(2/\beta)} - p_i\tilde{S}(\sqrt{p_i})\Big| = p_i \cdot \Big|\frac{\ln(1/\sqrt{p_i})}{2\ln(2/\beta)} - \tilde{S}(\sqrt{p_i})\Big| \leq \eta p_i. \tag{8}$$

For all i such that $p_i < \Delta$, we have

$$\Big|\frac{p_i\ln(1/p_i)}{4\ln(2/\beta)} - p_i\tilde{S}(\sqrt{p_i})\Big| \leq \frac{p_i\ln(1/p_i)}{4\ln(2/\beta)} + p_i \leq \frac{\Delta(\ln(\frac{1}{\Delta}) + 4\ln(2/\beta))}{4\ln(2/\beta)} \leq \frac{\epsilon}{8n\ln(2/\beta)}, \tag{9}$$

where the first inequality comes from the fact that $|\tilde{S}(x)| \leq 1$ for all $x \in [-1,1]$, the second inequality comes from the monotonicity of $x(\ln(1/x) + 4\ln(2/\beta))$ on $(0, \frac{1}{\Delta}]$, and the third inequality comes from (6). As a result of (5), (8), and (9), we have

$$\left|\left(\langle\psi_p|\otimes\langle 0|\right)|\widetilde{\Psi_p}\rangle - \frac{H(p)}{4\ln(2/\beta)}\right| = \left|p_i\tilde{S}(\sqrt{p_i}) - \sum_{i=1}^{n}\frac{p_i\log(1/p_i)}{4\ln(2/\beta)}\right|$$
$$\leq \sum_{i\colon p_i<\Delta}\frac{\epsilon}{8n\ln(2/\beta)} + \sum_{i\colon p_i\geq\Delta}\eta p_i$$
$$\leq \frac{\epsilon}{8\ln(2/\beta)} + \frac{\epsilon}{24\ln(2/\beta)} = \frac{\epsilon}{6\ln(2/\beta)}.$$

Therefore, $|4\ln(2/\beta)(\langle\psi_p|\otimes\langle 0|)|\widetilde{\Psi_p}\rangle - H(p)| \leq 2\epsilon/3$. By Theorem 8, we can use $\Theta(\ln(1/\beta)/\epsilon)$ applications of the unitaries (and their inverses) that implement $|\psi_p\rangle$ and $|\widetilde{\Psi_p}\rangle$ to estimate $(\langle\psi_p|\otimes\langle 0|)|\widetilde{\Psi_p}\rangle$ within additive error $\frac{\epsilon}{12\ln(2/\beta)}$. In total, this estimates $H(p)$ within additive error $\frac{\epsilon}{12\ln(2/\beta)}\cdot 4\ln(2/\beta)+\frac{2\epsilon}{3} = \epsilon$ with success probability at least $8/\pi^2$. The total complexity of the algorithm is

$$\mathcal{O}\left(\frac{\ln(1/\beta)}{\epsilon}\right)\cdot\mathcal{O}\left(\sqrt{\frac{n}{\epsilon}\log\left(\frac{n}{\epsilon}\right)}\log\left(\frac{\log n}{\epsilon}\right)\right) = \mathcal{O}\left(\frac{\sqrt{n}}{\epsilon^{1.5}}\log^{1.5}\left(\frac{n}{\epsilon}\right)\log\left(\frac{\log n}{\epsilon}\right)\right). \quad \blacktriangleleft$$

3.2 Density matrices with purified quantum query-access

For a density matrix ρ, we also assume the purified quantum query-access in Definition 2, i.e., a unitary oracle U_ρ acting as $U_\rho|0\rangle_A|0\rangle_B = |\rho\rangle = \sum_{i=1}^{n}\sqrt{p_i}|\phi_i\rangle_A|\psi_i\rangle_B$. We use U_ρ and $U_\rho^\dagger$ to estimate the von-Neumann entropy $H(\rho) = -\mathrm{Tr}[\rho\log\rho]$:

▶ **Theorem 13.** *For any $0<\epsilon<1$, we can estimate $H(p)$ with accuracy ϵ with success probability at least $2/3$ using $\widetilde{\mathcal{O}}\left(\frac{n}{\epsilon^{1.5}}\right)$ calls to U_ρ and $U_\rho^\dagger$.*

Proof. We use the construction of Eq. (2). The proof is essentially the same as that of Theorem 12 proceeding by constructing singular value transformation via Theorem 10, with the only difference that all probabilities are rescaled by a factor of $1/\sqrt{n}$ in (2); as a result, the number of calls to U_ρ and $U_\rho^\dagger$ is blown up to $\widetilde{\mathcal{O}}\left(\sqrt{n}\cdot\frac{\sqrt{n}}{\epsilon^{1.5}}\right) = \widetilde{\mathcal{O}}\left(\frac{n}{\epsilon^{1.5}}\right)$. ◀

4 Tolerant testers for ℓ^2-closeness with purified query-access

First we give an ℓ^2-closeness tester for unknown classical distributions p, q.

▶ **Theorem 14.** *Given purified quantum query-access for classical distributions p, q as in Definition 2, for any $\nu, \epsilon \in (0,1)$ the quantum query complexity of distinguishing the cases $\|p-q\|_2 \geq \epsilon$ and $\|p-q\|_2 \leq (1-\nu)\epsilon$ with success probability at least $2/3$ is*

$$\mathcal{O}\left(\frac{1}{\nu\epsilon}\log^3\left(\frac{1}{\nu\epsilon}\right)\log\log\left(\frac{1}{\nu\epsilon}\right)\right).$$

Proof. The main idea is to first bin the x elements based on the approximate value of $p(x)+q(x)$, then apply fine-tuned algorithms exploiting the knowledge of the approximate value of $p(x)+q(x)$.

Using amplitude estimation for any $k\in\mathbb{N}$ we can construct an algorithm $\mathcal{A}_k$ that for any input x with $p(x)+q(x)\geq 2^{-k}$ outputs "greater" with probability at least $2/3$, and for any x with $p(x)+q(x)\leq 2^{-k-1}$ outputs "smaller" and uses $\mathcal{O}\left(2^{\frac{k}{2}}\right)$ queries to U_p and U_q. Using $\mathcal{O}\left(\log(\frac{1}{\nu\epsilon})\right)$ repetitions we can boost the success probability to $1-\mathcal{O}(\mathrm{poly}(\nu\epsilon))$. Since our

algorithm only needs to succeed with constant probability, and will use these subroutines at most $\frac{1}{\mathrm{poly}(\nu\epsilon)}$ times, we can ignore the small failure probability. Therefore in the rest of the proof we assume without loss of generality, that $\mathcal{A}_k$ that solves perfectly the above question with (query) complexity $\mathcal{O}\Big(2^{\frac{k}{2}}\log(\frac{1}{\nu\epsilon}))\Big)$.

Algorithm 1 Estimating $\log_2(p(x)+q(x))$.

input $x \in [n]$, $\theta \in (0,1)$
for $k \in K := \left\{-1, 0, 1, 2, \ldots, \left\lceil \log_2\left(\frac{1}{\theta}\right)\right\rceil\right\}$ **do**
    Run algorithm $\mathcal{A}_k$ on $|x\rangle$ **if** output is "greater" **then return** k
return "less than θ"

For any x with $p(x)+q(x)\geq\theta$, Algorithm 1 outputs a k such that $p(x)+q(x)\in(2^{-k-1},2^{-k+1})$. However, note that this labeling is probabilistic; let us denote by $s_k(x)$ the probability that x is labeled by k. Observe that $s_k(x)=0$ unless $k\in\left\{\left\lfloor\log_2\left(\frac{1}{p(x)+q(x)}\right)\right\rfloor,\left\lceil\log_2\left(\frac{1}{p(x)+q(x)}\right)\right\rceil\right\}$ (otherwise the return is either "greater" or "less than"). Now let us express $\|p-q\|_2^2$ in terms of this "soft-selection" function $s(x)$.

$$\begin{aligned}\|p-q\|_2^2 &= \sum_x |p(x)-q(x)|^2 \\ &= \sum_x\sum_{k\in K} s_k(x)|p(x)-q(x)|^2+\eta && (\eta\in[0,2\theta)) \\ &= \sum_{k\in K} 2^{9-k}\sum_x s_k(x)\frac{p(x)+q(x)}{2}\frac{2^{-k-2}}{p(x)+q(x)}\left(\frac{p(x)-q(x)}{2^{-k+3}}\right)^2+\eta, && (10)\end{aligned}$$

where the bound on η follows from the observation that

$$\eta\leq\sum_{x\colon p(x)+q(x)<\theta}|p(x)-q(x)|^2\leq\sum_{x\colon p(x)+q(x)<\theta}(p(x)+q(x))^2<\theta\sum_{x\colon p(x)+q(x)<\theta}p(x)+q(x)<2\theta.$$

If for all $k\in K$ we have a $2^{k-9}\frac{\theta}{|K|}$-precise estimate of

$$\sum_x s_k(x)\frac{p(x)+q(x)}{2}\frac{2^{-k-2}}{p(x)+q(x)}\left(\frac{p(x)-q(x)}{2^{-k+3}}\right)^2, \qquad (11)$$

then we get a 3θ-precise estimate of $\|p-q\|_2^2$. In particular setting $\theta:=\nu\epsilon^2/6$, this solves the tolerant testing problem, since if $\|p-q\|\geq\epsilon$ then $\|p-q\|^2\geq\epsilon^2$, on the other hand if $\|p-q\|\leq(1-\nu)\epsilon$ then $\|p-q\|^2\leq(1-\nu)^2\epsilon^2\leq(1-\nu)\epsilon^2=\epsilon^2-\nu\epsilon^2$.

Now we describe how to construct a quantum algorithm that sets the first output qubit to $|0\rangle$ with probability (11). Start with preparing a purification of the distribution of $\frac{p(x)+q(x)}{2}$, then set the label of x to k with probability $s_k(x)$ using Algorithm 1 terminating it after using $\mathcal{A}_k$. Then separately apply the maps $\sqrt{\frac{2^{-k-2}}{p(x)+q(x)}}$ and $\frac{p(x)-q(x)}{2^{-k-3}}$ to the state.

Note that we do not need to apply the above transformations exactly, it is enough if apply them with precision say $2^{k-11}\frac{\theta}{|K|}$. We analyze the complexity of (approximately) implementing the above sketched algorithm. To implement the map $\sqrt{\frac{2^{-k-2}}{p(x)+q(x)}}$, we use the unitary of Eq. (1), and transform the singular values by the polynomial $\tilde{P}$ from Lemma 11 using Theorem 10. In order to implement the map $\frac{p(x)-q(x)}{2^{-k-2}}$, we again use the unitary of Eq. (1), but now separately for p and q. We amplify both the singular values $\sqrt{p(x)}$ and $\sqrt{q(x)}$

by a factor $\sqrt{2^{k-2}}$ using the polynomial $\tilde{Q}$ from Lemma 11 in Theorem 10. Then we create a block-encoding[8] of both and $2^{k-2}p(x)$ and $2^{k-2}q(x)$ and then combine them to get a block-encoding of $\frac{p(x)-q(x)}{2^{-k-3}}$. In both cases the query complexity of $\mathcal{O}(\theta/|K|)$-precisely implementing the transformations is $\mathcal{O}\big(2^{k/2}\log(|K|/\theta)\big) = \mathcal{O}\big(2^{k/2}\log(1/\theta)\big)$. Since computing the label k also costs $\mathcal{O}\big(2^{k/2}\log(1/(\nu\epsilon))\big)$, this is the overall complexity so far. Finally we estimate the probability of the first qubit being set to $|0\rangle$ with setting $M = \mathcal{O}\big(|K|2^{-k/2}/(\nu\epsilon)\big)$ in Theorem 8, and boost the success probability to $1-\mathcal{O}(1/|K|)$ with $\mathcal{O}(\log(|K|))$ repetitions. Thus for any $k \in K$ the overall complexity of estimating Eq. (11) with sufficient precision has (query) complexity $\mathcal{O}\Big(\frac{|K|}{\nu\epsilon}\log\big(\frac{1}{\nu\epsilon}\big)\log(|K|)\Big) = \mathcal{O}\big(\frac{1}{\nu\epsilon}\log^2\big(\frac{1}{\nu\epsilon}\big)\log\log\big(\frac{1}{\nu\epsilon}\big)\big)$. Therefore estimating $\|p-q\|_2^2$ to precision $\nu\epsilon^2/6$ with high probability has (query) complexity

$$\mathcal{O}\left(\frac{1}{\nu\epsilon}\log^3\left(\frac{1}{\nu\epsilon}\right)\log\log\left(\frac{1}{\nu\epsilon}\right)\right). \qquad \blacktriangleleft$$

It is easy to see an $\Omega\big(\frac{1}{\epsilon}\big)$ lower bound on the above problem even in the strongest quantum pure state input model Definition 4. Indeed, consider the case $n=2, q=(\frac{1}{2},\frac{1}{2})$ (the uniform distribution on $\{1,2\}$) and we want to test whether $p=q$ or $\|p-q\|_2 \geq \epsilon$. This is equivalent to test whether $p_1 = \frac{1}{2}$ or $|p_1 - \frac{1}{2}| \geq \frac{\epsilon}{\sqrt{2}}$; due to the optimality of amplitude estimation in Theorem 8, this task requires $\Omega(\frac{1}{\epsilon})$ quantum queries to the unitary U preparing the state $\sqrt{p_1}|1\rangle + \sqrt{p_2}|2\rangle$.

Now we prove the result below on (tolerant) ℓ^2-closeness testing for quantum distributions:

▶ **Theorem 15.** *Given $\epsilon, \nu \in (0,1)$ and two density operators $\rho, \sigma \in \mathbb{C}^{n\times n}$ with purified quantum query-access to U_ρ and U_σ as in Definition 2, it takes $\mathcal{O}\Big(\min\Big(\frac{\sqrt{n}}{\epsilon}, \frac{1}{\epsilon^2}\Big)\frac{1}{\nu}\Big)$ queries to $U_\rho, U_\rho^\dagger, U_\sigma, U_\sigma^\dagger$ to decide whether $\|\rho-\sigma\|_2 \geq \epsilon$ or $\|\rho-\sigma\|_2 \leq (1-\nu)\epsilon$, with success probability at least $2/3$.*

Proof. We can combine the block-encodings of ρ and σ to apply the map $\frac{\rho-\sigma}{2}$ to the maximally entangled state $\sum_{j=1}^n \frac{|j\rangle|j\rangle}{\sqrt{n}}$, which gives

$$\sum_{j=1}^n \frac{|j\rangle|j\rangle}{\sqrt{n}} \to \left(\frac{\rho-\sigma}{2}\otimes I\right)\sum_{j=1}^n \frac{|j\rangle|j\rangle}{\sqrt{n}}|0\rangle + \ldots|1\rangle.$$

The probability of measuring the $|0\rangle$ ancilla state is

$$\sum_{i,j=1}^n \frac{\langle i|\langle i|}{\sqrt{n}}\left(\frac{(\rho-\sigma)^2}{4}\otimes I\right)\frac{|j\rangle|j\rangle}{\sqrt{n}} = \frac{1}{4n}\sum_{i=1}^n \langle i|(\rho-\sigma)^2|i\rangle = \frac{1}{4n}\mathrm{Tr}[(\rho-\sigma)^2].$$

Thus it suffices to apply amplitude estimation with $M = \Theta\Big(\frac{\sqrt{n}}{\nu\epsilon}\Big)$ calls to $U_\rho, U_\rho^\dagger, U_\sigma, U_\sigma^\dagger$.

On the other hand, we can estimate $\|\rho-\sigma\|_2^2$ by observing that $\|\rho-\sigma\|_2^2 = \mathrm{Tr}\big[(\rho-\sigma)^2\big]$ $= \mathrm{Tr}\big[\rho^2\big] - 2\mathrm{Tr}[\rho\sigma] + \mathrm{Tr}\big[\sigma^2\big]$. Since the success probability of the SWAP test ([14]) on input states ρ, σ is $\frac{1}{2}(1+\mathrm{Tr}[\rho\sigma])$, we can individually estimate the latter quantities with precision $\mathcal{O}\big(\nu\epsilon^2\big)$ using amplitude estimation (Theorem 8) with $\mathcal{O}\big(\frac{1}{\nu\epsilon^2}\big)$ queries to $U_\rho, U_\rho^\dagger, U_\sigma, U_\sigma^\dagger$. As a result, we could decide whether $\|\rho-\sigma\|_2 \geq \epsilon$ or $\|\rho-\sigma\|_2 \leq (1-\nu)\epsilon$ using $\mathcal{O}\big(\frac{1}{\nu\epsilon^2}\big)$ queries.

The result of Theorem 15 hence follows by taking the minimum of the two complexities. ◀

[8] If we have a projected unitary encoding of $\Pi U\widetilde{\Pi} = A = \sum_i \varsigma_i|\psi_i\rangle\langle 0,i|$ with $\widetilde{\Pi} = |0\rangle\langle 0|\otimes I$, we can immediately turn it into a block-encoding of $A^\dagger A = \sum_i \varsigma_i^2|i\rangle\langle i|$ by e.g. applying Theorem 10 with the polynomial x^2.

5 Future work and open questions

Our paper raises a couple of natural open questions for future work, including:

- For which other distributional property testing problems can we get faster and simpler quantum algorithms using the presented methodology?
- Can we prove quantum lower bounds that match our upper bounds? For instance, can we prove an $\Omega\left(\frac{n}{\epsilon}\right)$ lower bound on estimating the von Neumann entropy in the purified quantum query-access model for density operators?
- Is there a lower bound technique which naturally fits our purified quantum query input model?
- Can we prove the conjecture that the purified and discrete query input models are equivalent for classical distributions, with respect to the query complexity of (distributional) property testing problems? For some recent progress in this direction see [11].

References

1 Jayadev Acharya, Hirakendu Das, Alon Orlitsky, and Ananda Theertha Suresh. A unified maximum likelihood approach for estimating symmetric properties of discrete distributions. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pages 11–21, 2017. URL: http://proceedings.mlr.press/v70/acharya17a.html, arXiv:1611.02960.

2 Jayadev Acharya, Constantinos Daskalakis, and Gautam Kamath. Optimal testing for properties of distributions. In *Advances in Neural Information Processing Systems 28 (NIPS)*, pages 3591–3599, 2015. URL: http://papers.nips.cc/paper/5839-optimal-testing-for-properties-of-distributions, arXiv:1507.05952.

3 Jayadev Acharya, Ibrahim Issa, Nirmal V. Shende, and Aaron B. Wagner. Measuring Quantum Entropy. In *2019 IEEE International Symposium on Information Theory (ISIT)*, pages 3012–3016, 2019. arXiv:1711.00814.

4 Dorit Aharonov and Amnon Ta-Shma. Adiabatic Quantum State Generation. *SIAM Journal on Computing*, 37(1):47–82, 2007. Earlier version in STOC'03, arXiv:quant-ph/0301023. doi:10.1137/060648829.

5 Srinivasan Arunachalam, Sourav Chakraborty, Troy Lee, and Ronald de Wolf. Two new results about quantum exact learnings, 2018. arXiv:1810.00481.

6 Srinivasan Arunachalam and Ronald de Wolf. Optimal Quantum Sample Complexity of Learning Algorithms. In *Proceedings of the 32nd IEEE Conference on Computational Complexity (CCC)*, pages 25:1–25:31, 2017. doi:10.4230/LIPIcs.CCC.2017.25.

7 Costin Bădescu, Ryan O'Donnell, and John Wright. Quantum state certification. In *Proceedings of the 51st ACM Symposium on the Theory of Computing (STOC)*, pages 503–514. ACM, 2019. doi:10.1145/3313276.3316344.

8 Tuğkan Batu, Sanjoy Dasgupta, Ravi Kumar, and Ronitt Rubinfeld. The complexity of approximating the entropy. *SIAM Journal on Computing*, 35(1):132–150, 2005. doi:10.1145/509907.510005.

9 Tuğkan Batu, Eldar Fischer, Lance Fortnow, Ravi Kumar, Ronitt Rubinfeld, and Patrick White. Testing random variables for independence and identity. In *Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 442–451, 2001. doi:10.1109/SFCS.2001.959920.

10 Tuğkan Batu, Lance Fortnow, Ronitt Rubinfeld, Warren D Smith, and Patrick White. Testing closeness of discrete distributions. *Journal of the ACM*, 60(1):4, 2013. doi:10.1145/2432622.2432626.

11 Aleksandrs Belovs. Quantum Algorithms for Classical Probability Distributions, 2019. arXiv:1904.02192.

12 Gilles Brassard, Peter Høyer, Michele Mosca, and Alain Tapp. Quantum Amplitude Amplification and Estimation. In *Quantum Computation and Quantum Information: A Millennium Volume*, volume 305 of *Contemporary Mathematics Series*, pages 53–74. AMS, 2002. doi:10.1090/conm/305.

13 Sergey Bravyi, Aram W. Harrow, and Avinatan Hassidim. Quantum algorithms for testing properties of distributions. *IEEE Transactions on Information Theory*, 57(6):3971–3981, 2011. doi:10.1109/TIT.2011.2134250.

14 Harry Buhrman, Richard Cleve, John Watrous, and Ronald de Wolf. Quantum Fingerprinting. *Physical Review Letters*, 87(16):167902, 2001. doi:10.1103/PhysRevLett.87.167902.

15 Mark Bun, Robin Kothari, and Justin Thaler. The polynomial method strikes back: Tight quantum query bounds via dual polynomials. In *Proceedings of the 50th ACM Symposium on the Theory of Computing (STOC)*, 2018. doi:10.1145/3188745.3188784.

16 André Chailloux. A Note on the Quantum Query Complexity of Permutation Symmetric Functions. In *Proceedings of the 10th Innovations in Theoretical Computer Science Conference (ITCS)*, pages 19:1–19:7, 2018. doi:10.4230/LIPIcs.ITCS.2019.19.

17 Shantanav Chakraborty, András Gilyén, and Stacey Jeffery. The power of block-encoded matrix powers: improved regression techniques via faster Hamiltonian simulation. In *Proceedings of the 46th International Colloquium on Automata, Languages, and Programming (ICALP)*, 2019. doi:10.4230/LIPIcs.ICALP.2019.28.

18 Sourav Chakraborty, Eldar Fischer, Arie Matsliah, and Ronald de Wolf. New Results on Quantum Property Testing. In *Proceedings of the 30th International Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, page 145, 2010. doi:10.4230/LIPIcs.FSTTCS.2010.145.

19 Siu-On Chan, Ilias Diakonikolas, Gregory Valiant, and Paul Valiant. Optimal algorithms for testing closeness of discrete distributions. In *Proceedings of the 25th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1193–1203, 2014. doi:10.1137/1.9781611973402.88.

20 Ilias Diakonikolas and Daniel M. Kane. A new approach for testing properties of discrete distributions. In *Proceedings of the 57th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 685–694, 2016. doi:10.1109/FOCS.2016.78.

21 András Gilyén, Yuan Su, Guang Hao Low, and Nathan Wiebe. Quantum singular value transformation and beyond: exponential improvements for quantum matrix arithmetics. In *Proceedings of the 51st ACM Symposium on the Theory of Computing (STOC)*, 2019. doi:10.1145/3313276.3316366.

22 Oded Goldreich. *Introduction to property testing*. Cambridge University Press, 2017.

23 Jeongwan Haah, Aram W. Harrow, Zhengfeng Ji, Xiaodi Wu, and Nengkun Yu. Sample-optimal tomography of quantum states. *IEEE Transactions on Information Theory*, 63(9):5628–5641, 2017. arXiv:1508.01797.

24 Yassine Hamoudi and Frédéric Magniez. Quantum Chebyshev's Inequality and Applications. In *Proceedings of the 46th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 69:1–69:16, 2019. doi:10.4230/LIPIcs.ICALP.2019.69.

25 Jiantao Jiao, Kartik Venkat, Yanjun Han, and Tsachy Weissman. Minimax estimation of functionals of discrete distributions. *IEEE Transactions on Information Theory*, 61(5):2835–2885, 2015. doi:10.1109/TIT.2015.2412945.

26 Reut Levi, Dana Ron, and Ronitt Rubinfeld. Testing properties of collections of distributions. *Theory of Computing*, 9(1):295–347, 2013. doi:10.4086/toc.2013.v009a008.

27 Tongyang Li and Xiaodi Wu. Quantum Query Complexity of Entropy Estimation. *IEEE Transactions on Information Theory*, 65(5):2899–2921, 2019. doi:10.1109/TIT.2018.2883306.

28 Guang Hao Low and Isaac L. Chuang. Hamiltonian Simulation by Qubitization. *Quantum*, 3:163, 2019. doi:10.22331/q-2019-07-12-163.

29 Ashley Montanaro. Quantum speedup of Monte Carlo methods. *Proceedings of the Royal Society A*, 471(2181), 2015. doi:10.1098/rspa.2015.0301.

30 Ashley Montanaro and Ronald de Wolf. *A Survey of Quantum Property Testing*. Number 7 in Graduate Surveys. Theory of Computing Library, 2016. arXiv:1310.2035, doi:10.4086/toc.gs.2016.007.

31 Michael A. Nielsen and Isaac L. Chuang. *Quantum computation and quantum information*. Cambridge University Press, 2000. doi:10.1017/CBO9780511976667.

32 Ryan O'Donnell and John Wright. Quantum Spectrum Testing. In *Proceedings of the 47th ACM Symposium on the Theory of Computing (STOC)*, pages 529–538, 2015. doi:10.1145/2746539.2746582.

33 Ryan O'Donnell and John Wright. Efficient Quantum Tomography. In *Proceedings of the 48th ACM Symposium on the Theory of Computing (STOC)*, pages 899–912, 2016. doi:10.1145/2897518.2897544.

34 Ryan O'Donnell and John Wright. Efficient quantum tomography II. In *Proceedings of the 49th ACM Symposium on the Theory of Computing (STOC)*, pages 962–974, 2017. doi:10.1145/3055399.3055454.

35 Liam Paninski. Estimation of entropy and mutual information. *Neural Computation*, 15(6):1191–1253, 2003. doi:10.1162/089976603321780272.

36 Liam Paninski. Estimating entropy on m bins given fewer than m samples. *IEEE Transactions on Information Theory*, 50(9):2200–2203, 2004. doi:10.1109/TIT.2004.833360.

37 Liam Paninski. A coincidence-based test for uniformity given very sparsely sampled discrete data. *IEEE Transactions on Information Theory*, 54(10):4750–4755, 2008. doi:10.1109/TIT.2008.928987.

38 Alfréd Rényi. On measures of entropy and information. In *Proceedings of the 4th Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 547–561. University of California Press, 1961. URL: https://projecteuclid.org/euclid.bsmsp/1200512181.

39 Claude E. Shannon. A Mathematical Theory of Communication. *Bell System Technical Journal*, 27(3):379–423, 1948. doi:10.1002/j.1538-7305.1948.tb01338.x.

40 Gregory Valiant and Paul Valiant. Estimating the unseen: an $n/log(n)$-sample estimator for entropy and support size, shown optimal via new CLTs. In *Proceedings of the 43rd ACM Symposium on the Theory of Computing (STOC)*, pages 685–694, 2011. doi:10.1145/1993636.1993727.

41 Gregory Valiant and Paul Valiant. The power of linear estimators. In *Proceedings of the 52nd IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 403–412, 2011. doi:10.1109/FOCS.2011.81.

42 Yihong Wu and Pengkun Yang. Minimax rates of entropy estimation on large alphabets via best polynomial approximation. *IEEE Transactions on Information Theory*, 62(6):3702–3720, 2016. doi:10.1109/TIT.2016.2548468.

A Projected unitary encodings used for singular value transformation

First we handle the case of classical distributions. Let U_p be a purified quantum oracle of a classical distribution p as in Definition 2, and let $U := (U_p \otimes I)$, also let $\Pi := (\sum_{i=1}^{n} I \otimes |i\rangle\langle i| \otimes |i\rangle\langle i|)$, $\widetilde{\Pi} := (|0\rangle\langle 0| \otimes |0\rangle\langle 0| \otimes I)$, then

$$\begin{aligned}\Pi U \widetilde{\Pi} = \Pi (U_p \otimes I)\widetilde{\Pi} &= \Big(\sum_{i=1}^{n} I \otimes |i\rangle\langle i| \otimes |i\rangle\langle i|\Big)(U_p \otimes I)\big(|0\rangle\langle 0| \otimes |0\rangle\langle 0| \otimes I\big) \\ &= \sum_{i=1}^{n} \Big((I \otimes |i\rangle\langle i|) U_p (|0\rangle\langle 0| \otimes |0\rangle\langle 0|)\Big) \otimes |i\rangle\langle i|\end{aligned}$$

$$\ldots = \sum_{i=1}^{n} \Big((I \otimes |i\rangle\langle i|) \sum_{j=1}^{n} \sqrt{p_j} |\phi_j\rangle|j\rangle\langle 0|\langle 0| \Big) \otimes |i\rangle\langle i|$$
$$= \sum_{i=1}^{n} \sqrt{p_i} |\phi_i\rangle\langle 0| \otimes |i\rangle\langle 0| \otimes |i\rangle\langle i|.$$

Now we turn to quantum distributions where we do not know the diagonalizing basis of the density operator ρ. Let U_ρ be a purified quantum oracle of a quantum distribution ρ as in Definition 2, and W a unitary, mapping $|0\rangle|0\rangle \mapsto \sum_{j=1}^{n} \frac{|j\rangle|j\rangle}{\sqrt{n}}$. Let $U' := \big(I \otimes U_\rho^\dagger\big)\big(W^\dagger \otimes I\big)$, $\Pi' := (I \otimes |0\rangle\langle 0| \otimes |0\rangle\langle 0|)$ and $\widetilde{\Pi}$ as above, then

$$\begin{aligned}
\Pi' U' \widetilde{\Pi} &= \Pi' \big(I \otimes U_\rho^\dagger\big)\big(W^\dagger \otimes I\big)\widetilde{\Pi} \\
&= \big(I \otimes (|0\rangle\langle 0| \otimes |0\rangle\langle 0| U_\rho^\dagger)\big) \left(\left(\sum_{j=1}^{n} \frac{|j\rangle|j\rangle}{\sqrt{n}} \right) \langle 0|\langle 0| \otimes I \right) \\
&= \left(I \otimes \sum_{i=1}^{n} \sqrt{p_i} |0\rangle|0\rangle\langle \phi_i|\langle \psi_i| \right) \left(\left(\sum_{j=1}^{n} \frac{|\phi'_j\rangle|\phi_j\rangle}{\sqrt{n}} \right) \langle 0|\langle 0| \otimes I \right) \\
&= \sum_{i=1}^{n} \sqrt{\frac{p_i}{n}} |\phi'_i\rangle|0\rangle|0\rangle\langle 0|\langle 0|\langle \psi_i|,
\end{aligned}$$

where $\sum_{j=1}^{n} \frac{|\phi'_j\rangle|\phi_j\rangle}{\sqrt{n}} = \sum_{j=1}^{n} \frac{|j\rangle|j\rangle}{\sqrt{n}}$ is the Schmidt decomposition of the maximally entangled state under the basis $(|\phi_1\rangle, \ldots, |\phi_n\rangle)$.

B Polynomial approximations for singular value transformation

We use the following result based on local Taylor series:

▶ **Lemma 16** ([21, Corollary 66]). *Let $x_0 \in [-1, 1]$, $r \in (0, 2]$, $\nu \in (0, r]$ and let $f\colon [-x_0 - r - \nu, x_0 + r + \nu] \to \mathbb{C}$ and be such that $f(x_0 + x) = \sum_{\ell=0}^{\infty} a_\ell x^\ell$ for all $x \in [-r - \nu, r + \nu]$. Suppose $B > 0$ is such that $\sum_{\ell=0}^{\infty} (r + \nu)^\ell |a_\ell| \leq B$. Let $\epsilon \in \big(0, \frac{1}{2B}\big]$, then there is an efficiently computable polynomial $P \in \mathbb{C}[x]$ of degree $\mathcal{O}\big(\frac{1}{\nu} \log\big(\frac{B}{\epsilon}\big)\big)$ such that*[9]

$$\begin{aligned}
\|f(x) - P(x)\|_{[x_0 - r, x_0 + r]} &\leq \epsilon \\
\|P(x)\|_{[-1,1]} &\leq \epsilon + \|f(x)\|_{[x_0 - r - \nu/2, x_0 + r + \nu/2]} \leq \epsilon + B \\
\|P(x)\|_{[-1,1] \setminus [x_0 - r - \nu/2, x_0 + r + \nu/2]} &\leq \epsilon.
\end{aligned}$$

We can use the above result to construct the following useful polynomial approximations.

▶ **Lemma 11.** (Polynomial approximations) *Let $\beta \in (0, 1]$, $\eta \in (0, \frac{1}{2}]$ and $t \geq 1$. There exists polynomials $\tilde{P}, \tilde{Q}, \tilde{S}$ such that*

- $\forall x \in [\frac{1}{t}, 1]\colon |\tilde{P}(x) - \frac{1}{2tx}| \leq \eta$, *and* $\forall x \in [-1, 1]\colon -1 \leq \tilde{P}(x) = \tilde{P}(-x) \leq 1$,
- $\forall x \in [-\frac{1-\beta}{t}, \frac{1-\beta}{t}]\colon |\tilde{Q}(x) - tx| \leq \eta \cdot (tx)$, *and* $\forall x \in [-1, 1]\colon \tilde{Q}(x) = -\tilde{Q}(-x) \leq 1$,
- $\forall x \in [\beta, 1]\colon |\tilde{S}(x) - \frac{\ln(1/x)}{2\ln(2/\beta)}| \leq \eta$, *and* $\forall x \in [-1, 1]\colon -1 \leq \tilde{S}(x) = \tilde{S}(-x) \leq 1$,

moreover $\deg(\tilde{P}) = \mathcal{O}\Big(t \log\Big(\frac{1}{\eta}\Big)\Big)$, $\deg(\tilde{Q}) = \mathcal{O}\Big(\frac{t}{\beta} \log\Big(\frac{1}{\eta}\Big)\Big)$, *and* $\deg(\tilde{S}) = \mathcal{O}\Big(\frac{1}{\beta} \log\Big(\frac{1}{\eta}\Big)\Big)$.

[9] For a function $g\colon \mathbb{R} \to \mathbb{C}$, and an interval $[a, b] \subseteq \mathbb{R}$, we define $\|g\|_{[a,b]} := \max_{x \in [a,b]} |g(x)|$.

Proof. For the construction of the $\tilde{P}$ and $\tilde{Q}$ polynomials see Corollary 67 and Theorem 30 of [21], respectively. It remains to construct the polynomial $\tilde{S}$ above.

Denote $f(x) = \frac{\ln(1/x)}{2\ln(2/\beta)}$; by taking $\epsilon = \eta/2$, $x_0 = 1$, $r = 1-\beta$, $\nu = \frac{\beta}{2}$, and $B = \frac{1}{2}$ in Corollary 16, we have a polynomial $S \in \mathbb{C}[x]$ of degree $\mathcal{O}\big(\frac{1}{\nu}\log(\frac{B}{\epsilon})\big) = \mathcal{O}\Big(\frac{1}{\beta}\log(\frac{1}{\eta})\Big)$ such that

$$\|f(x) - S(x)\|_{[\beta,2-\beta]} \le \eta/2 \tag{12}$$

$$\|S(x)\|_{[-1,1]} \le B + \eta/2 \le (1+\eta)/2 \tag{13}$$

$$\|S(x)\|_{[-1,\frac{\beta}{2}]} \le \eta/2. \tag{14}$$

Note that $B = \frac{1}{2}$ is valid because the Taylor series of $f(x)$ at $x = 1$ is $\frac{1}{2\ln(2/\beta)}\sum_{l=1}^{\infty}\frac{(-1)^l x^l}{l}$, and as a result we could take

$$\begin{aligned} B = \frac{1}{2\ln(2/\beta)}\sum_{l=1}^{\infty}\frac{(1-\beta/2)^l}{l} &= -\frac{1}{2\ln(2/\beta)}\sum_{l=1}^{\infty}\frac{(-1)^{l-1}}{l}(-1+\beta/2)^l \\ &= -\frac{1}{2\ln(2/\beta)}\ln\frac{\beta}{2} = \frac{1}{2}. \end{aligned}$$

However, S is not an even polynomial in general; we instead take $\tilde{S}(x) = S(x) + S(-x)$ for all $x \in [-1,1]$. Then by (12) and (14) we have

$$\big\|f(x) - \tilde{S}(x)\big\|_{[\beta,1]} \le \big\|f(x) - \tilde{S}(x)\big\|_{[\beta,1]} + \big\|\tilde{S}(-x)\big\|_{[\beta,1]} \le \frac{\eta}{2} + \frac{\eta}{2} = \eta. \tag{15}$$

Furthermore, $\tilde{S}$ is an even polynomial such that $\deg(\tilde{S}) = \mathcal{O}\Big(\frac{1}{\beta}\log(\frac{1}{\eta})\Big)$; hence (13) and (14) imply

$$\big\|\tilde{S}(x)\big\|_{[-1,1]} = \big\|\tilde{S}(x)\big\|_{[0,1]} \le \|S(x)\|_{[0,1]} + \|S(x)\|_{[-1,0]} \le \frac{1+\eta}{2} + \frac{\eta}{2} \le 1$$

given $\eta \le 1/2$. (Finally we can take the real part of $\tilde{S}(x)$ if it has some complex coefficients.) ◀

C Corollaries of our ℓ^2-closeness testing results

C.1 ℓ^1-closeness testing with purified query-access

▶ **Corollary 17.** *Given $\epsilon > 0$ and two distributions p, q on the domain $[n]$ with purified quantum query-access via U_p and U_q as in Definition 2, it takes $\widetilde{\mathcal{O}}\Big(\frac{\sqrt{n}}{\epsilon}\Big)$ queries to $U_p, U_p^\dagger, U_q, U_q^\dagger$ to decide whether $p = q$ or $\|p-q\|_1 \ge \epsilon$ with success probability at least $2/3$. Similarly for density operators $\rho, \sigma \in \mathbb{C}^{n\times n}$ with purified quantum query-access via U_ρ and U_σ, it takes $\mathcal{O}\big(\frac{n}{\epsilon}\big)$ queries to $U_\rho, U_\rho^\dagger, U_\sigma, U_\sigma^\dagger$ to decide whether $\rho = \sigma$ or $\|\rho-\sigma\|_1 \ge \epsilon$ with success probability at least $2/3$.*

Proof. By the Cauchy-Schwartz inequality we have $\|p-q\|_2 \ge \frac{1}{\sqrt{n}}\|p-q\|_1$, therefore Theorem 14 implies our claim by taking $\epsilon \leftarrow \epsilon/\sqrt{n}$ therein. Similarly, Theorem 15 implies our claim for quantum distributions ρ and σ. ◀

C.2 Independence testing with purified query-access

▶ **Corollary 18.** *Given $\epsilon > 0$ and a classical distribution p on $[n] \times [m]$ with the purified quantum query-access via U_p as in Definition 2, it takes $\widetilde{\mathcal{O}}\left(\frac{\sqrt{nm}}{\epsilon}\right)$ queries to $U_p, U_p^\dagger$ to decide whether p is a product distribution on $[n] \times [m]$ or p is ϵ-far in ℓ^1-norm from any product distribution on $[n] \times [m]$ with success probability at least $2/3$.*

Proof. We define p_A to be the margin of p on the first marginal space, i.e., $p_A(i) = \sum_{j=1}^m p(i,j)$ for all $i \in [n]$. We similarly define p_B to be the margin of p on the second marginal space, i.e., $p_B(j) = \sum_{i=1}^n p(i,j)$ for all $j \in [m]$. Assume the quantum oracle U_p from Definition 2 acts as

$$U_p|0\rangle_A|0\rangle_B|0\rangle_C = \sum_{i=1}^n \sum_{j=1}^m \sqrt{p(i,j)}|i\rangle_A|j\rangle_B|\psi_{i,j}\rangle_C;$$

if we denote $|\phi_i\rangle = \sum_{j=1}^m \frac{\sqrt{p(i,j)}}{\sqrt{p_A(i)}}|j\rangle|\psi_{i,j}\rangle$ for all $i \in [n]$ and $|\varphi_j\rangle = \sum_{i=1}^n \frac{\sqrt{p(i,j)}}{\sqrt{p_B(j)}}|i\rangle|\psi_{i,j}\rangle$ for all $j \in [m]$, then we have

$$U_p|0\rangle_A|0\rangle_B|0\rangle_C = \sum_{i=1}^n \sqrt{p_A(i)}|i\rangle_A|\phi_i\rangle_{B,C} = \sum_{j=1}^m \sqrt{p_B(j)}|j\rangle_B|\varphi_j\rangle_{A,C}.$$

As a result,

$$(U_p \otimes U_p)(|0\rangle^{\otimes 6}) = \sum_{i=1}^n \sum_{j=1}^m \sqrt{p_A(i)}\sqrt{p_B(j)}|i\rangle|j\rangle|\phi_i\rangle|\varphi_j\rangle;$$

in other words, one purified quantum query to the distribution $p_A \times p_B$ can be implemented by two queries to U_p.

If p is a product distribution on $[n] \times [m]$, then $p = p_A \times p_B$; if p is ϵ-far in ℓ^1-norm from any product distribution on $[n] \times [m]$, then $\|p - p_A \times p_B\|_1 \geq \epsilon$. Therefore, the problem of independence testing reduces to ℓ^1-closeness testing for distributions on $[n] \times [m]$, and hence Corollary 18 follows from Corollary 17. ◀

Similarly, Corollary 17 implies that the quantum query complexity of testing independence of quantum distributions is $\mathcal{O}\left(\frac{nm}{\epsilon}\right)$.

On Local Testability in the Non-Signaling Setting

Alessandro Chiesa
UC Berkeley, CA, USA
alexch@berkeley.edu

Peter Manohar
Carnegie Mellon University, Pittsburgh, PA, USA
pmanohar@cs.cmu.edu

Igor Shinkar
Simon Fraser University, Burnaby, Canada
ishinkar@sfu.ca

Abstract

Non-signaling strategies are a generalization of quantum strategies that have been studied in physics for decades, and have recently found applications in theoretical computer science. These applications motivate the study of local-to-global phenomena for *non-signaling functions*.

We prove that low-degree testing in the non-signaling setting is possible, assuming that the locality of the non-signaling function exceeds a threshold. We additionally show that if the locality is below the threshold then the test fails spectacularly, in that there exists a non-signaling function which passes the test with probability 1 and yet is maximally far from being low-degree.

Along the way, we present general results about the local testability of linear codes in the non-signaling setting. These include formulating natural definitions that capture the condition that a non-signaling function "belongs" to a given code, and characterizing the sets of local constraints that imply membership in the code. We prove these results by formulating a logical inference system for linear constraints on non-signaling functions that is complete and sound.

2012 ACM Subject Classification Theory of computation → Computational complexity and cryptography

Keywords and phrases non-signaling strategies, locally testable codes, low-degree testing, Fourier analysis

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.26

Related Version Full version available at `https://eccc.weizmann.ac.il/report/2019/070/`.

Funding *Alessandro Chiesa*: Ethereum Foundation and the Interchain Foundation
Peter Manohar: NSF Graduate Research Fellowship Program (under Grant No. DGE1745016); and the ARCS Foundation. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.
Igor Shinkar: NSERC Discovery Grant

Acknowledgements We are grateful to Thomas Vidick for suggesting using irreducible curves to extend our initial non-testability result for axis-parallel lines to the case of general lines.

1 Introduction

Locally testable codes (LTCs) are error correcting codes in which one can verify whether a given string belongs to the code by reading only a few (randomly chosen) bits from the string. Goldreich and Sudan [6] have described LTCs as the "combinatorial counterparts of the complexity theoretic notion of PCPs", motivating the standalone study of these objects.

In this work we study local testability for *non-signaling strategies*, which are a class of non-local strategies that generalize quantum strategies, capturing the maximum amount of "non-local correlation" that can occur under the assumption that spatially-isolated parties cannot communicate instantaneously. Non-signaling strategies have been studied in physics for decades [14, 10, 12], in order to better understand quantum entanglement. Recently they have gained attention in computer science due to their applications to hardness of approximation [8] and delegation of computation [7, 9]. PCPs sound against non-signaling strategies (nsPCPs) underlie these applications, which motivates the study of local testability in the non-signaling setting.

Given an integer n, a field $\mathbb{F}$, and a locality parameter $k \leq n$, the object that we study is a *k-non-signaling function* $\mathcal{F}\colon [n] \to \mathbb{F}$, which extends the notion of a function $f\colon [n] \to \mathbb{F}$ as follows.[1]

▶ **Definition 1.** *A **k-non-signaling function** $\mathcal{F}\colon [n] \to \mathbb{F}$ is a collection $\{\mathcal{F}_S\}_{S\subseteq[n]:|S|\leq k}$ where each $\mathcal{F}_S$ is a distribution over local functions $g\colon S \to \mathbb{F}$, and for any two subsets $R \subseteq S \subseteq [n]$ with $|S| \leq k$ it holds that the distribution $\mathcal{F}_R$ and the marginal distribution $\mathcal{F}_S|_R$ are equal.[2] (The set of all such $\mathcal{F}$ are the solutions to the k-relaxation in the Sherali–Adams hierarchy [16].)*

The evaluation of $\mathcal{F}$ on a set S is a single sample $g\colon S \to \mathbb{F}$ from the distribution $\mathcal{F}_S$. Intuitively, a k-non-signaling function is like a quantum function: evaluation is probabilistic and only happens once, just like quantum measurement; and $\mathcal{F}$ can only be evaluated on at most k points simultaneously, which is similar to the uncertainty principle. As k approaches n, $\mathcal{F}$ behaves more like a classical function and, when $k = n$, $\mathcal{F}$ is a distribution over functions $f\colon [n] \to \mathbb{F}$.

Local testability of non-signaling functions may sound like an oxymoron, because non-signaling functions, at least superficially, are collections of local distributions with no global structure that we can talk about. Yet prior work has shown that local-to-global phenomena *are possible*.

For example, [4] shows that any non-signaling function passing the linearity test [3] with high probability is well-approximated by a *quasi-distribution* supported on linear functions. This result was later used in [5] to show that the exponential-length constant-query PCP of [2] is sound against non-signaling strategies.

The results obtained in [4, 5] naturally raise the question of whether local testability in the non-signaling setting is possible for other codes, like those based on low-degree polynomials. After all, both linearity testing and low-degree testing do work in the quantum setting [11].

Recall that, in the classical setting, local testability plays a central role in PCP constructions, many of which can be described as having two main components.

- *Property testing:* check with few queries whether or not the given proof π belongs to a code $\mathbf{C}$.
- *Checking computation:* given that π is a codeword in $\mathbf{C}$ (or at least is close to a codeword), check with few queries whether or not π proves the desired statement.

This modular approach has enabled the study of local testability as a natural standalone goal, which in turn has led to improved PCP constructions.

---

[1] There are two distinct definitions of a non-signaling strategy, depending on whether the strategy is meant to represent isolated parties or a function. The former is used for MIPs [7, 9], while the latter is used for PCPs and property testing [7, 9, 4, 5]. We use the latter definition, although equivalent statements of all our results will hold when adopting the former definition (see the appendix in [4]).

[2] A common relaxation of this condition requires that these two distributions are only statistically (or computationally) close. While we consider the standard definition, we note that this is without loss of generality as [4] shows that every statistically (or computationally) non-signaling strategy is close to an (exact) non-signaling strategy.

Inspired by this state of affairs, we initiate the study of locally testable codes in general in the non-signaling setting, focusing specifically on the case of low-degree testing. We believe that, similarly to the classical setting, understanding local testability against non-signaling strategies will enable researchers to construct more efficient non-signaling PCPs.

1.1 Low-degree testing against non-signaling functions

We show that a simple low-degree test, the *evenly-spaced points test*, tests proximity to degree-d non-signaling functions when $k \geq O(d^2)$, and *fails* to test proximity when $k \leq O(d^2)$.

The evenly-spaced points test. Let $m, d \in \mathbb{N}$ and p be a prime with $p \geq d+2$. Given a function $f\colon \mathbb{F}_p^m \to \mathbb{F}_p$, the degree-$d$ evenly-spaced points test: (1) samples a random point $x \in \mathbb{F}_p^m$ and slope $h \in \mathbb{F}_p^m \setminus \{0^m\}$, (2) checks that $\sum_{i=0}^{d+1} c_i f(x+ih) = 0$, where $c_i = (-1)^i \binom{d+1}{i}$. It is well-known that if f passes the degree-d evenly-spaced points test with high probability, then f is close to (the evaluation of) an m-variate polynomial of total degree at most d [15]. Below we ask whether the test is also sound in the non-signaling setting.

Suppose that a k-non-signaling function $\mathcal{F}\colon \mathbb{F}_p^m \to \mathbb{F}_p$ passes the evenly-spaced points test with high probability. Can we deduce any global low-degree structure about $\mathcal{F}$?

In more detail, the probabilistic experiment that we consider is this: first we sample x and h according to the distribution of the evenly-spaced points test, and let the query set S be $\{x + ih : i \in \{0, \ldots, d+1\}\}$; then we sample a local function $g\colon S \to \mathbb{F}_p$ according to the distribution $\mathcal{F}_S$; and finally we check that $\sum_{i=0}^{d+1} c_i g(x+ih) = 0$.

The answer to the above question will, in general, depend on the locality parameter k of $\mathcal{F}$. At minimum, we need $k \geq d+2$ for otherwise we cannot even run the evenly-spaced points test (k is the maximum number of simultaneous queries to $\mathcal{F}$). At the other extreme, when k has the maximum value ($k = p^m$) then we are back to the classical case because $\mathcal{F}$ is now a distribution over functions $f\colon \mathbb{F}_p^m \to \mathbb{F}_p$; hence if $\mathcal{F}$ passes the test with high probability then (one can verify that) with high probability a function f sampled according $\mathcal{F}$ is close to low-degree. In fact, even when $k \geq O(d^m)$, we are in a trivial case, as one can query $\mathcal{F}$ on an interpolating set, a "cube of $(d+1)^m$ points".

We are thus interested in whether or not the test works for *non-trivial* values of k, namely when $O(d) \leq k < O(d^m)$, and thus we will assume that $m \geq 2$. In this regime, k is large enough to run the test, and yet is small enough so that one cannot query an interpolating set. Our first result shows that the test succeeds in the non-signaling setting when $k \geq O(d^2)$. This is a non-signaling analogue of the evenly-spaced points test, similar to how [4] gives a non-signaling analogue of the linearity test of [3].

▶ **Theorem 2** (informal). *Let $\mathcal{F}\colon \mathbb{F}_p^m \to \mathbb{F}_p$ be a k-non-signaling function that passes the degree-d evenly-spaced points test with high probability, where $k \geq (d+2)^2$. Then $\mathcal{F}$ has a global individual degree-d structure.*

One drawback of the above theorem is that the conclusion only asserts that $\mathcal{F}$ has an individual degree-d structure, when one would like to conclude that $\mathcal{F}$ has a total degree-d structure. We discuss the difficulty of extending the result to total degree-d in Remark 13.

Our second result shows that the test fails when $k \leq O(d^2)$, and moreover it fails *even when the test passes with probability* 1, namely, it fails in the worst possible sense.

▶ **Theorem 3** (informal). *For every k with $2d+2 \leq k < \frac{3}{16}(d+2)^2$, there exists a k-non-signaling function that passes the evenly-spaced points test with probability 1, and yet is $(1-\frac{1}{p})$-far from all degree-d k-non-signaling functions.*

Theorem 3 is surprising, as it reveals that in the non-signaling setting, there is a regime of k in which the low-degree test fails. This stands in sharp contrast to the fact that for linearity testing, there is *no regime* of k in which non-signaling linearity test fails. Our results thus suggest that low-degree testing is a qualitatively different task, as it has a regime of k where a natural test fails.

Theorem 3 also shows that in the case of bivariate testing when $m = 2$, there is (up to constants) no non-trivial value of k where the test succeeds. This is counterintuitive, as bivariate testing is a natural test for which we would expect some guarantee to hold (regardless of how weak), at the very least when the test passes with probability 1.

Other low-degree tests. We note that Theorem 3 generalizes to *any* low-degree test over an arbitrary finite field $\mathbb{F}$ that (1) works by checking constraints that lie along a line, and (2) has perfect completeness.

Beyond low-degree testing. Our theorems on low-degree testing come from applying more general results that we prove about the structure of local characterizations for *any linear code*, in the non-signaling setting. We view our general results on local characterizations as a significant technical contribution within this paper, and we now discuss them.

1.2 Local characterizations and linear proofs

Local characterizations are fundamental to the study of locally testable codes [15]. They express membership in a given linear code via a set of low-weight constraints, and they naturally induce a canonical tester: sample a random low-weight constraint and check if the given word satisfies it. In order to prove the negative result in Theorem 3, we do not need to consider distributions on constraints, but instead we only need to study how constraints express code membership, via *exact* local characterizations [15]. Below we describe one of our main technical contributions, which informally consists of establishing necessary and sufficient conditions for when a constraint set is a local characterization for a code, in the non-signaling setting. We begin by recalling known facts about local characterizations in the classical setting, and then proceed to the non-signaling setting.

The classical setting. A *constraint set* $T \subseteq \mathbb{F}^n$ for a linear code $\mathbf{C} \subseteq \mathbb{F}^n$ is a subset of its dual code $\mathbf{C}^\perp$. A constraint set T is a *ℓ-local characterization* of $\mathbf{C}$ if every $\alpha \in T$ has at most ℓ non-zero entries, and the condition "$\langle \alpha, f \rangle = 0$ for every $\alpha \in T$" implies that $f \in \mathbf{C}$ (and conversely).

For example, the set $\{e_x + e_y - e_{x+y} : x, y \in \{0,1\}^n\}$ where e_x is the x-th standard basis vector in $\{0,1\}^{\{0,1\}^n}$ is a 3-local characterization of the Hadamard code, because $f(x) + f(y) - f(x+y) = 0$ for every $x, y \in \{0,1\}^n$ implies that f is a linear function, and conversely. As another example, the Reed–Muller code containing all polynomials $f \colon \mathbb{F}^m \to \mathbb{F}$ in m variables of total degree at most d has a $(d+2)$-local characterization T, where T contains a constraint α for each subset S of $\mathbb{F}^m$ of size $d+2$ that is contained in a line.

There is a simple condition that is both necessary and sufficient for a constraint set T to be a local characterization for $\mathbf{C}$: the span of T equals $\mathbf{C}^\perp$. In this work it is useful to view this condition instead through the lens of mathematical logic, as follows. Given a constraint set T and $\alpha \in \mathbb{F}^n$, we define the notion of a linear proof.

▶ **Definition 4** (Linear proof). *We write $T \vdash \alpha$ (T proves α) if there exists a sequence $(\alpha_0{:=}0^n, \alpha_1, \ldots, \alpha_{r-1}, \alpha_r{:=}\alpha)$ with each $\alpha_i \in \mathbb{F}^n$ such that, for every $i \in [r]$, one of the following holds:*

- *$\exists\, j < i$ and $b \in \mathbb{F}$ such that $\alpha_i = b\alpha_j$,*
- *$\exists\, j < i$ and $\gamma \in T$ such that $\alpha_i = \alpha_j + \gamma$,*
- *$\exists\, j_1, j_2 < i$ such that $\alpha_i = \alpha_{j_1} + \alpha_{j_2}$.*

As an example, suppose that $\alpha = \sum_{i=1}^{r} b_i\gamma_i$ with each $b_i \in \mathbb{F}$ and $\gamma_i \in T$. Then the sequence $(0^n, \gamma_1, b_1\gamma_1, \ldots, \gamma_r, b_r\gamma_r, \alpha_1, \ldots, \alpha_r)$, where each α_i is the partial sum $b_1\gamma_1 + \cdots + b_i\gamma_i$, gives a linear proof that $\alpha \in \mathrm{span}(T)$.

One can immediately see that $T \vdash \alpha$ if and only if $\alpha \in \mathrm{span}(T)$. In particular, we have the following lemma.

▶ **Lemma 5.** *Linear proofs are (i) complete: if $\langle \gamma, f\rangle = 0$ for every $\gamma \in T$ implies $\langle \alpha, f\rangle = 0$, then $T \vdash \alpha$; and (ii) sound: if $\langle \gamma, f\rangle = 0$ for every $\gamma \in T$ and $T \vdash \alpha$, then $\langle \alpha, f\rangle = 0$. In particular, a constraint set T is a local characterization of a linear code $\mathbf{C}$ if and only if $T \vdash \mathbf{C}^{\perp}$.*

Our goal is to establish a non-signaling analogue of Lemma 5.

A motivating example. We illustrate via an example why a statement like Lemma 5 is non-trivial in the non-signaling setting. Let $n \in \mathbb{N}$ be even, and let $T = \{1^n - e_i : i \in [n]\} \subseteq \{0,1\}^n$, i.e. T contains every vector that is 1 in all but one of the coordinates, where it is 0. Classically, one can check that T is a $(n-1)$-local characterization of the code $\mathbf{C} = \{0^n\}$, as if $f \in \{0,1\}^n$ satisfies $\langle \alpha, f\rangle = 0$ for every $\alpha \in T$ (equivalently, $\sum_{\ell \neq i} f(\ell) = 0$ for every $i \in [n]$), then we must have $f = 0^n$, since n is even. This is because $T \vdash e_i$, and so f must satisfy $f(i) = \langle e_i, f\rangle = 0$.

However, there exist $(n-1)$-non-signaling functions that satisfy every constraint in T and yet are not identically 0; the non-signaling function which outputs uniformly random bits with parity 0 on every set of size exactly $n-1$ is one such example. In particular, T is not a $(n-1)$-local characterization of $\mathbf{C}$. To see why, let us examine where the classical argument that $f(i) = 0$ fails for $(n-1)$-non-signaling functions. Recall that an $(n-1)$-non-signaling function can only be evaluated simultaneously at $n-1$ points. Thus, while one can classically argue, for example, that $f(i) + f(j) = 0$ via the argument that $\sum_{\ell \neq i} f(\ell) = 0$ and $\sum_{\ell \neq j} f(\ell) = 0$ implies that $f(i) + f(j) = \sum_{\ell \neq i} f(\ell) + \sum_{\ell \neq j} f(\ell) = 0$, this reasoning is no longer valid in the non-signaling setting because it requires f to be simultaneously defined at every $i \in [n]$, which is $n > k = n-1$ points. In particular, any proof that $\langle e_i, f\rangle = 0$ from T requires f to be simultaneously defined on all n points, so this logical reasoning is not valid in the non-signaling setting.

An equivalence for non-signaling functions. We prove an analogous equivalence in the non-signaling setting, which informally states that a suitable notion of local characterization for any linear code is equivalent to being able to prove all low weight elements of $\mathbf{C}^{\perp}$ using *local* proofs. This equivalence is a strict generalization of Lemma 5. The example above can thus be viewed as a case where a low weight element of $\mathbf{C}^{\perp}$ (namely, e_i) has no local proof from a particular T.

We begin by formulating a notion of local characterization that works for constraint sets applied to non-signaling functions rather than (classical) functions. There are two main qualitative differences with the classical case. First, the definition depends on the locality parameter k because we need to specify the locality of the non-signaling functions that we

consider. Second, the requirement that a non-signaling function "belongs" to a code $\mathbf{C}$ is expressed via a property that we call $\mathbf{C}$-*explainability*, on which we comment after the definition.

▶ **Definition 6** (informal). *A constraint set $T \subseteq \mathbf{C}^\perp$ is a* **ℓ-local characterization** *for $(\mathbf{C}, k)$ if every $\alpha \in T$ has at most ℓ non-zero entries, and the set of k-non-signaling functions that satisfy every $\alpha \in T$ with probability 1 equals the set of k-non-signaling functions that are "$\mathbf{C}$-explainable".*

The term "$\mathbf{C}$-explainable" refers to the condition that the given non-signaling function is, with probability 1, consistent with the restriction of some codeword in $\mathbf{C}$. This condition is motivated by non-trivial properties of the Fourier spectrum of non-signaling functions that we discuss later on (see Section 2.5). For now, it suffices to say that if a non-signaling function $\mathcal{F}$ is $\mathbf{C}$-explainable then $\mathcal{F}$ satisfies natural *global* properties that extend code membership to the non-signaling setting.

We remark that Definition 6 reduces to the classical notion of local characterization when setting $k := n$. We now introduce the notion of local linear proofs that we use in our equivalence.

▶ **Definition 7** (k-local linear proof). *Given a constraint set T and $\alpha \in \mathbb{F}^n$, we write $T \vdash_k \alpha$ if there exists a sequence $(\alpha_0{:=}0^n, \alpha_1, \ldots, \alpha_{r-1}, \alpha_r{:=}\alpha)$ with each $\alpha_i \in \mathbb{F}^n$ such that, for every $i \in [r]$, one of the following holds:*

- *$\exists\, j < i$ and $b \in \mathbb{F}$ such that $\alpha_i = b\alpha_j$*
- *$\exists\, j < i$ and $\gamma \in T$ such that $|\mathrm{supp}(\alpha_j) \cup \mathrm{supp}(\gamma)| \leq k$ and $\alpha_i = \alpha_j + \gamma$*
- *$\exists\, j_1, j_2 < i$ such that $|\mathrm{supp}(\alpha_{j_1}) \cup \mathrm{supp}(\alpha_{j_2})| \leq k$ and $\alpha_i = \alpha_{j_1} + \alpha_{j_2}$.*

Above, $\mathrm{supp}(\alpha)$ denotes the set of indices $i \in [n]$ where $\alpha_i \neq 0$, and $\mathrm{wt}(\alpha)$ is the size of $\mathrm{supp}(\alpha)$. Notice that Definition 7 is nearly identical to Definition 4: the only change is the addition of the restriction on the support size in the second and third bullets.

The motivation behind Definition 7 is the following fact: if $T \vdash_k \alpha$ then any k-non-signaling function that satisfies every constraint in T must satisfy α as well. Definition 7 thus captures a notion of constraint propagation for non-signaling functions. The restriction on the support size in the second and third bullets is there because querying a k-non-signaling function on more than k points simultaneously is undefined.

We now state our main technical contribution in this section, a non-signaling analogue of Lemma 5.

▶ **Theorem 8** (informal). *k-local linear proofs are complete and sound for k-non-signaling functions. In particular, a constraint set T is a ℓ-local characterization for $(\mathbf{C}, k)$ if and only if $T \vdash_k \alpha$ for every $\alpha \in \mathbf{C}^\perp$ with $\mathrm{wt}(\alpha) \leq k$.*

Proving that k-local linear proofs are sound is straightforward; the interesting component of Theorem 8 is showing that k-local linear proofs are complete. We do this by showing that for every T there exists a k-non-signaling function that satisfies every α where $T \vdash_k \alpha$, and violates every α where $T \nvdash_k \alpha$ with probability $1 - \frac{1}{|\mathbb{F}|}$.

When $k = n$ in Theorem 8 we recover the classical statement (Lemma 5). This is because when $k = n$, $T \vdash_k \alpha$ if and only if $T \vdash \alpha$. However, when $k < n$, the equivalence is qualitatively different from its classical analogue. While Lemma 5 essentially captures a simple linear algebraic statement (the constraints span the dual code), Theorem 8 is a non-trivial statement that *does not involve linear spaces*. This is because the requirement $T \vdash_k \alpha$ depends on k in a way that breaks linearity, as exhibited by our motivating example earlier.

1.3 Roadmap

In Section 2 we provide an overview of the proofs of our results. Then, in Section 3 and Section 4 we formally define non-signaling functions, quasi-distributions, and discuss the relationship between them using Fourier analysis. In Section 5 we discuss what it means for a non-signaling function to "belong" to a given linear code. In Section 6 we prove that the non-signaling low-degree test works (Theorem 2). In Section 7 we prove an equivalence between local characterizations for non-signaling linear codes and local linear proofs (Theorem 8). We conclude in Section 8, by using Theorem 8 to show that the non-signaling low-degree test fails for small locality (Theorem 3).

2 Techniques

We outline the techniques used to prove our results. We begin by explaining the Fourier structure of non-signaling functions in Section 2.1. This structure is fundamental to the proofs of our results. We then outline the proof of Theorem 2 in Section 2.2. In Section 2.3 we outline our proof of the relationship between local characterizations and local linear proofs. In Section 2.4 we use the techniques and main theorem from Section 2.3 to show Theorem 3, that any low-degree lines test fails for non-signaling functions when $k \leq O(d^2)$. Finally, in Section 2.5 we justify our definition of "**C**-explainability".

Notation. A k-non-signaling function $\mathcal{F}$ is defined by local distributions $\mathcal{F}_S$ for each $S \subseteq [n]$ with $|S| \leq k$. Because of this, when studying non-signaling functions we naturally encounter situations where we only consider subsets of a domain containing at most k elements, or vectors in $\mathbb{F}^n$ of weight at most k. We introduce notation to make referring to these notions more convenient. For a subset $S \subseteq [n]$ we write $S \subseteq [n]_{\leq k}$ if $|S| \leq k$. For a vector $\alpha \in \mathbb{F}^n$, we let $\mathrm{supp}(\alpha) = \{i \in [n] : \alpha_i \neq 0\}$ and $\mathrm{wt}(\alpha) = |\mathrm{supp}(\alpha)|$. For a set of vectors $R \subseteq \mathbb{F}^n$, we let $R_{\leq k} \subseteq R$ denote the subset $\{\alpha \in R : \mathrm{wt}(\alpha) \leq k\}$. In particular, $\mathbb{F}^n_{\leq k}$ denotes the set $\{\alpha \in \mathbb{F}^n : \mathrm{wt}(\alpha) \leq k\}$. For a subset $S \subseteq [n]$, we use similar notation and let $R_{\subseteq S} = \{\alpha \in R : \mathrm{supp}(\alpha) \subseteq S\}$.

2.1 The Fourier structure of non-signaling functions

We make frequent use of Fourier analysis to state and establish properties of non-signaling functions. Below we recall basic facts about Fourier analysis, explain their application to quasi-distributions, and state an equivalence between non-signaling functions and quasi-distributions. This equivalence motivates a definition for the Fourier spectrum of a non-signaling function.

Refresher on Fourier analysis. Let $\mathbb{F}$ be the finite field of size q with characteristic p, and $\mathbb{F}_p$ the prime subfield of $\mathbb{F}$. The inner product of $F_1, F_2 \colon \mathbb{F}^n \to \mathbb{C}$ is $\langle F_1, F_2 \rangle := \frac{1}{q^n} \sum_{f \in \mathbb{F}^n} \overline{F_1(f)} F_2(f)$. The *character* corresponding to $\alpha \in \mathbb{F}^n$ is the function $\chi_\alpha \colon \mathbb{F}^n \to \mathbb{C}$ defined as $\chi_\alpha(f) := \omega^{\mathrm{Tr}(\langle \alpha, f \rangle)}$ where: $\mathrm{Tr} \colon \mathbb{F} \to \mathbb{F}_p$ is the trace map; $\langle \alpha, f \rangle$ is the inner product $\sum_{i=1}^n \alpha_i f_i$; $\omega = e^{2\pi i/p}$ is a primitive complex p-th root of unity; and ω^j is defined by thinking of $j \in \mathbb{F}_p$ as an integer in $\{0, 1, \ldots, p-1\}$. The characters $\{\chi_\alpha\}_{\alpha \in \mathbb{F}^n}$ form an orthonormal basis of the space of all functions $F \colon \mathbb{F}^n \to \mathbb{C}$, so every function $F \colon \mathbb{F}^n \to \mathbb{C}$ can be written as

$$F(\cdot) = \sum_{\alpha \in \mathbb{F}^n} \widehat{F}(\alpha) \chi_\alpha(\cdot) \ , \text{ where } \widehat{F}(\alpha) := \langle \chi_\alpha, F \rangle \ .$$

The values $\{\widehat{F}(\alpha)\}_{\alpha \in \mathbb{F}^n}$ are called the *Fourier coefficients* of F.

Quasi-distributions. A *quasi-distribution* $\mathcal{Q}$ over functions $f\colon [n] \to \mathbb{F}$ is a distribution where the probability weights are complex numbers that "add up" to real probabilities. More formally, a quasi-distribution is a function $\mathcal{Q}\colon \mathbb{F}^n \to \mathbb{C}$ where $\sum_{f\in\mathbb{F}^n} \mathcal{Q}(f) = 1$. (We abuse notation and identify a function $f\colon [n] \to \mathbb{F}$ with the vector in $\mathbb{F}^n$ corresponding to its evaluation table.) We say that $\mathcal{Q}$ is *k-local* if the marginals $\mathcal{Q}|_S$ for each $S \subseteq [n]_{\leq k}$ are distributions, namely, if for each $S \subseteq [n]_{\leq k}$ and $g\colon S \to \mathbb{F}$ it holds that $\sum_{f\in\mathbb{F}^n : f|_S = g} \mathcal{Q}(f)$ is a non-negative real number. We can decompose a quasi-distribution $\mathcal{Q}$ according to the Fourier basis: we can write $\mathcal{Q}(f) = \sum_{\alpha\in\mathbb{F}^n} \widehat{\mathcal{Q}}(\alpha)\chi_\alpha(f)$, where $\{\chi_\alpha\}_{\alpha\in\mathbb{F}^n}$ are the characters and $\{\widehat{\mathcal{Q}}(\alpha)\}_{\alpha\in\mathbb{F}^n}$ are the Fourier coefficients of $\mathcal{Q}$.

Equivalence lemma. The following lemma shows that k-local quasi-distributions and k-non-signaling functions are equivalent, and exposes the Fourier structure of non-signaling functions.

▶ **Lemma 9.** *A quasi-distribution $\mathcal{Q}$ is equivalent to a k-non-signaling function $\mathcal{F}$ if and only if for every $\alpha \in \mathbb{F}^n_{\leq k}$ it holds that $\widehat{\mathcal{Q}}(\alpha) = \frac{1}{q^n}\sum_{j\in\mathbb{F}_p} \omega^{-j} \Pr[\mathrm{Tr}(\langle\alpha,\mathcal{F}\rangle) = j]$, where the random variable $\langle\alpha,\mathcal{F}\rangle$ has the probability distribution given by*

$$\left\{\Pr[\langle\alpha,\mathcal{F}\rangle = b] := \Pr_{f\leftarrow\mathcal{F}_{\mathrm{supp}(\alpha)}}\left[\textstyle\sum_{i\in\mathrm{supp}(\alpha)} \alpha_i f(i) = b\right]\right\}_{b\in\mathbb{F}} .$$

The foregoing lemma motivates defining the Fourier coefficients of a k-non-signaling function $\mathcal{F}$ as follows: for every $\alpha \in \mathbb{F}^n$ with $\mathrm{wt}(\alpha) \leq k$ we define

$$\widehat{\mathcal{F}}(\alpha) := \frac{1}{q^n}\sum_{j\in\mathbb{F}_p} \omega^{-j} \Pr[\mathrm{Tr}(\langle\alpha,\mathcal{F}\rangle) = j] .$$

For more details on the above, including the proof of our Equivalence Lemma, see Section 4.

2.2 Low-degree testing

We outline the proof of Theorem 2. As a simple case, we first state and prove the theorem in the zero error case (when test passes with probability 1), and then we briefly explain how to extend the proof to the robust case (when the test passes with probability $1 - \varepsilon$).

The zero error case. Let $\mathbf{C}$ be the set of all m-variate polynomials of *individual* degree d. Formally, we first show the following.

▶ **Theorem 10** (formal version of Theorem 2, zero error case)**.** *Let $m, d \in \mathbb{N}$ and p be a prime with $p \geq d + 2$. Let $\mathcal{F}\colon \mathbb{F}_p^m \to \mathbb{F}_p$ be a k-non-signaling function, and suppose that $\mathcal{F}$ passes the degree-d evenly-spaced points test with probability 1. Then $\mathcal{F}$ (viewed as a $\lfloor k/(d+2)\rfloor$-non-signaling function) is $\mathbf{C}$-explainable.*

In the language of Section 1.2, Theorem 10 shows that T, the set of linear constraints checked by the degree-d evenly spaced points test, is a $(d+2)$-local characterization of $\mathbf{C}$.

Theorem 10 is a non-signaling analogue of the following classical fact: if $f\colon \mathbb{F}_p^m \to \mathbb{F}_p$ passes the evenly-spaced points test with probability 1, then f is a polynomial of total degree d. (Note that in Theorem 10 we only conclude that $\mathcal{F}$ has individual degree d. We remark on the difference after the proof.) Our proof of Theorem 10 can be interpreted as taking a local proof of the aforementioned classical fact, and lifting it to the non-signaling setting.

Concretely, let us consider the following simple classical statement.

▶ **Theorem 11** (folklore). *Let $f\colon \mathbb{F}_p^m \to \mathbb{F}_p$ be a function such that, for every line L, f agrees with a univariate degree-d polynomial on L. Then for every $S \subseteq \mathbb{F}_p^m$, there exists a degree-$d$ function g such that $g|_S = f|_S$.*

There are multiple known proofs of Theorem 11. To demonstrate the challenges in the non-signaling setting, we first outline a standard classical proof of Theorem 11 that will not generalize to the non-signaling setting. The proof uses the following lemma.

▶ **Lemma 12.** *Suppose that $f\colon \mathbb{F}_p^m \to \mathbb{F}_p$ is a polynomial where $\deg(f) = d < p$. Then there exists a line L in $\mathbb{F}_p^m$ such that $f|_L$ is a univariate polynomial of degree exactly d.*

The above lemma is shown by considering the function $f(a + tb)$ where $a, b \in \mathbb{F}_p^m$, and arguing that the coefficient of t^d in $f(a + tb)$ is a non-zero polynomial in the variables a, b, and hence does not vanish for all $a, b \in \mathbb{F}_p^m$. Thus, there exists a line $L(t) = a + tb$ for which the coefficient of t^d is non-zero, and therefore $f|_L$ has degree exactly d.

With the above lemma, one can prove Theorem 11 as follows. Suppose that f is a polynomial of degree $d' > d$. Then by the lemma there exists a line L such that $f|_L$ has degree $d' > d$, contradicting the fact that $f|_L$ has degree at most d.[3]

The above proof is a good example of a proof that will *not* extend to the non-signaling setting. This is because the proof of Lemma 12 is "global", in the sense that arguing about the polynomial coefficients of f requires "knowing" $f(a)$ for $\Omega(d^m)$ points $a \in \mathbb{F}_p^m$, as this is the minimum number of evaluations of f needed for all polynomial coefficients of f to be fixed. As indicated by Section 1.2, the types of classical proofs that will extend to the non-signaling setting are those with small locality, i.e. ones that require looking at $f(a)$ for only a small number of a at a time. This implies that the above proof will not work for k-non-signaling functions when $k \leq O(d^m)$, i.e., when k is a non-trivial value.

We instead present the following local proof of Theorem 11. Since this proof has small locality it will extend to the non-signaling setting, and thus imply Theorem 10.

Let $S \subseteq \mathbb{F}_p^m$. We wish to show that $f|_S = g|_S$ for some $g \in \mathbf{C}$. Let $S_0 = \emptyset$, and for each $i \in [m]$ define $S_i \subseteq \mathbb{F}_p^i$ to be the projection of S to the first i coordinates, so S_i is the set of all $(a_1, \ldots, a_i) \in \mathbb{F}_p^i$ such that $(a_1, \ldots, a_i, b_{i+1}, \ldots, b_m) \in S$ for some $(b_{i+1}, \ldots, b_m) \in \mathbb{F}_p^{m-i}$. Note that $S_m = S$.

We prove by induction that for every $i \in [m]$ and every $b_{i+1}, \ldots, b_m \in \mathbb{F}_p$ there exists an individual degree-d polynomial $g_i\colon \mathbb{F}_p^i \to \mathbb{F}_p$ such that $f|_{S_i \times \{(b_{i+1}, \ldots, b_m)\}} = g_i|_{S_i}$. This proves Theorem 11, as $S_m = S$. The base case ($i = 1$) holds since f looks degree-d on every line, so in particular f is degree-d on the line $\mathbb{F}_p \times \{(b_2, \ldots, b_m)\}$, which contains S_1.

We now argue the induction step. Suppose that the induction hypothesis holds for $i - 1$ and every $b_i, \ldots, b_m \in \mathbb{F}_p$. The induction hypothesis implies that for each $j \in \{0, \ldots, d\}$, there exists an individual degree-d polynomial $g_{i-1}^{(j)}\colon \mathbb{F}_p^{i-1} \to \mathbb{F}_p$ such that $g_{i-1}^{(j)}|_{S_{i-1}} = f|_{S_{i-1} \times \{j\} \times \{(b_{i+1}, \ldots, b_m)\}}$. Let $g_i\colon \mathbb{F}_p^i \to \mathbb{F}_p$ be defined by interpolating the $g_{i-1}^{(j)}$'s along the i-th axis, i.e. $g_i(x_1, \ldots, x_i) := \sum_{j=0}^{d} \delta_j(x_i) \cdot g_{i-1}^{(j)}(x_1, \ldots, x_{i-1})$ where $\delta_j(y)$ is the unique degree-d univariate polynomial that is 1 if $y = j$ and 0 otherwise. We then argue that f agrees with g_i on $S_{i-1} \times \mathbb{F}_p$. This is because f agrees with g_i on $S_{i-1} \times \{0, \ldots, d\}$ (since here $g_i = g_{i-1}^{(j)} = f$ by the induction hypothesis), and therefore agrees with g_i on $S_{i-1} \times \mathbb{F}_p$ by polynomial interpolation, since f looks degree-d on any axis-parallel line along the i-th axis.

[3] The argument as stated does not quite work, as the lemma only holds when $d' < p$. Here, we ignore this technicality to simplify the presentation of the argument.

The above proof can be adapted to an $|S|(d+2)$-local proof (as stated above, it is $|S|p$-local). We thus conclude that if a k-non-signaling function $\mathcal{F}$ looks degree-d on every line L, then it also looks individual degree-d on every S where $|S|(d+2) \leq k$, i.e., $\mathcal{F}$ (viewed as a $\lfloor k/(d+2) \rfloor$-non-signaling function) is $\mathbf{C}$-explainable.

In the aforementioned argument, we have crucially required that $\mathcal{F}$ looks low-degree along *every* line, rather than merely on sets of evenly-spaced points, which are the only constraints checked by the test. Thus, we must show that if $\mathcal{F}$ passes the degree-d evenly-spaced points test with probability 1, then $\mathcal{F}$ looks degree-d on arbitrary subsets of any line. This last step can be viewed as the following. Let T be the set of constraints checked by the evenly-spaced test, and let T' be the set of all low-weight line constraints (weight at most $k-d-2$) satisfied by degree-d polynomials. We show that $T \vdash_k T'$, so $\mathcal{F}$ (by Theorem 8) must also satisfy all constraints in T', and thus looks degree-d on arbitrary subsets of lines, which concludes the proof of Theorem 10.

▶ Remark 13 (total degree vs. individual degree). Theorem 2 only concludes that $\mathcal{F}$ has an individual degree-d structure, when one might expect to conclude that it has a *total* degree-d structure, as it passes the evenly-spaced points test along *random lines*. Indeed, this is the conclusion in the classical setting. The difficulty in establishing such a result comes from Lemma 12. The classical analysis of the low-degree test proceeds by induction, initially concluding that $f\colon \mathbb{F}^m \to \mathbb{F}$ is a polynomial that is degree-d in x_m and total degree-d in all the other variables, and hence is a total degree-$2d$ polynomial. Then, by using Lemma 12 one concludes that in fact f has total degree-d, not $2d$. A non-signaling analogue of Lemma 12 would allow us to conclude a total degree-d structure. However, as explained earlier the classical proof of Lemma 12 is not local, so it does not lift to a non-signaling one. Exploring whether or not the gap between total and individual degree is necessary in the non-signaling setting is thus an intriguing open question.

The robust case. We now explain how to adapt the above proof to the robust case. Our goal now is to show that $\mathcal{F}$ is close to a $\mathbf{C}$-explainable non-signaling function, where the distance between two k-non-signaling functions $\mathcal{F}$ and $\mathcal{G}$ is defined as

$$\Delta_k(\mathcal{F},\mathcal{G}) = \max_{S\subseteq[n],|S|\leq k} \Delta_{\mathrm{TV}}(\mathcal{F}_S,\mathcal{G}_S) \ ,$$

where Δ_{TV} is the total variation distance between distributions [4]. As in the case of linearity testing in [4], this is impossible, as the definition of distance requires that $\mathcal{F}$ be close to $\mathbf{C}$-explainable on all sets $S \subseteq \mathbb{F}_p^m$ with $|S| \leq k$. In particular, if $\mathcal{F}$ looks low-degree on all lines but one, then $\mathcal{F}$ will be very far from $\mathbf{C}$-explainable. Following [4], we instead show that an appropriately defined self-correction of $\mathcal{F}$, denoted by $\hat{\mathcal{F}}$, is close to $\mathbf{C}$-explainable. Informally, $\hat{\mathcal{F}}(x)$ is defined by querying $\mathcal{F}$ on a random evenly-spaced line L passing through x, and then setting $\hat{\mathcal{F}}(x)$ to be the value at x obtained by locally decoding $\mathcal{F}$ along L. $\hat{\mathcal{F}}$ is a $\hat{k}$-non-signaling function, where $\hat{k} = k/(d+1)$.

We prove Theorem 2 via the following four steps:

1. Average to worst case reduction: we show that if $\mathcal{F}$ passes the evenly-spaced points test with high probability, then $\hat{\mathcal{F}}$ looks low-degree on *every* set of evenly-spaced set of points contained in a line L with high probability.
2. From evenly-spaced points to arbitrary subsets of a line: we show that if $\hat{\mathcal{F}}$ looks low-degree on every set of evenly-spaced set of points contained in a line, then $\hat{\mathcal{F}}$ looks low-degree on every subset of every line L.

3. Robust local characterization: we show that T, the set of constraints where the support of the constraint is contained in some line L, is a robust local characterization of $\mathbf{C}$, i.e. that if $\hat{\mathcal{F}}$ satisfies every $\alpha \in T$ with high probability, then $\hat{\mathcal{F}}$ satisfies every $\alpha \in \mathbf{C}^{\perp}_{\leq k'}$ with high probability, where $k' = \hat{k}/(d+2)$.
4. Finishing the proof: we show that if $\hat{\mathcal{F}}$ satisfies every $\alpha \in \mathbf{C}^{\perp}_{\leq k'}$ with high probability, then $\hat{\mathcal{F}}$ is close to a $\mathbf{C}$-explainable non-signaling function.

We have already discussed the proofs of the second and third steps in the zero error case. In the robust case, the main difference is that we now pay some small error in union bounds every time we use the fact that $\hat{\mathcal{F}}$ looks low-degree along an evenly-spaced line.

The first step follows from our non-trivial definition of $\hat{\mathcal{F}}$. Naively, one might define $\hat{\mathcal{F}}$ for each x by locally decoding its value from $\mathcal{F}$ along a random evenly-spaced line containing x. This does not work. Instead, we decode its value along the line $L_x(t) = x + iw_x$, where the slopes (the w_x's) are correlated so that $w_{x+y} = w_x + w_y - w_{0^n}$. These correlations, combined with the fact that $L(t)$ looks random for each x, allows us to show that $\hat{\mathcal{F}}$ looks low-degree on every evenly-spaced set of points.

The final step follows abstractly from the more general statements we show for all linear codes, and relies on our characterization of the Fourier spectrum of $\mathbf{C}$-explainable non-signaling functions.

2.3 Local characterizations and linear proofs

We outline the proof of Theorem 8; we assume familiarity with the notions introduced in Section 1.2. We begin by formally defining local characterizations.

Local characterizations. We say that a k-non-signaling function $\mathcal{F}$ is $\mathbf{C}$-*explainable* if for every $S \subseteq [n]_{\leq k}$, with probability 1 the function $f\colon S \to \mathbb{F}$ sampled from $\mathcal{F}_S$ is in $\mathbf{C}|_S$. (See Section 2.5 for a discussion of this definition.) Recall from Definition 6 that a subset $T \subseteq \mathbf{C}^{\perp}$ is an ℓ-local characterization of $(\mathbf{C}, k)$ if every $\alpha \in T$ has $\mathrm{wt}(\alpha) \leq \ell$ and the set of k-non-signaling functions $\mathcal{F}$ where $\Pr[\langle \alpha, \mathcal{F} \rangle = 0] = 1$ for every $\alpha \in T$ equals the set of $\mathbf{C}$-explainable k-non-signaling functions.

Outline of the proof. The proof of Theorem 8 has two directions: completeness and soundness. For soundness, we show that if $T \vdash_k \alpha$, then for any k-non-signaling function $\mathcal{F}$ where $\langle \gamma, \mathcal{F} \rangle = 0$ holds with probability 1 for every $\gamma \in T$, it also holds that $\langle \alpha, \mathcal{F} \rangle = 0$ with probability 1. Intuitively, this means that any k-non-signaling function satisfying every constraint in T must satisfy α as well, and therefore shows that our definition of "proof" makes sense. The proof of this direction is straightforward, and can be found in Section 7.1.

To show completeness, we explicitly construct a k-non-signaling function $\mathcal{F}$ that satisfies every constraint α where $T \vdash_k \alpha$ with probability 1, and satisfies every other constraint α with probability $\frac{1}{|\mathbb{F}|}$. Our construction of $\mathcal{F}$ makes crucial use of the notion of a *local subspace* that we introduce.

▶ **Definition 14.** *A* **k-local subspace** *$\mathcal{V}$ is a subset of $\mathbb{F}^n_{\leq k}$ that looks like a subspace when restricted to local views of size at most k, i.e., $\mathcal{V}_{\subseteq S}$ is a linear subspace in $\mathbb{F}^n$ for every $S \subseteq [n]_{\leq k}$.*

We show that for any k-local subspace $\mathcal{V}$ there is a k-non-signaling function $\mathcal{F}$ where $\Pr[\langle \alpha, \mathcal{F} \rangle = 0] = 1$ for every $\alpha \in \mathcal{V}$ and $\Pr[\langle \alpha, \mathcal{F} \rangle = 0] = \frac{1}{|\mathbb{F}|}$ otherwise. We then show that the set of α's provable from T, which is $\{\alpha \in \mathbb{F}^n_{\leq k} : T \vdash_k \alpha\}$, is a k-local subspace. This latter step is straightforward, and the proof is in Section 7.3. We now discuss the first step, which is non-trivial.

Non-signaling functions from local subspaces. Given a k-local subspace $\mathcal{V}$, we argue that there is a k-non-signaling function $\mathcal{F}$ where $\Pr[\langle \alpha, \mathcal{F} \rangle = 0] = 1$ for every $\alpha \in \mathcal{V}$, and $\Pr[\langle \alpha, \mathcal{F} \rangle = 0] = \frac{1}{|\mathbb{F}|}$ for every $\alpha \notin \mathcal{V}$. We construct $\mathcal{F} = \{\mathcal{F}_S\}_{S \subseteq [n] : |S| \leq k}$ by specifying its local distributions $\mathcal{F}_S$.

A distribution over functions $f \colon S \to \mathbb{F}$ is a function that maps each f to a non-negative real number such that the total sum is 1. With this viewpoint, we first define $\mathcal{F}_S$ as a *function* that maps each $f \colon S \to \mathbb{F}$ to a complex number. Then, we show that the total sum is 1 and that each f is mapped to a non-negative real number, so that the function $\mathcal{F}_S$ is indeed a distribution.

We define the function $\mathcal{F}_S \colon \mathbb{F}^S \to \mathbb{C}$ by specifying its Fourier coefficients:

$$\widehat{\mathcal{F}_S}(\alpha) := \begin{cases} \frac{1}{q^{|S|}} & \text{if } \alpha \in \mathcal{V} \\ 0 & \text{if } \alpha \notin \mathcal{V} \end{cases} ,$$

These "local" Fourier coefficients should *not* be confused with the Fourier coefficients of $\mathcal{F}$ that are defined in Section 2.1. In fact, at this point the non-signaling function $\mathcal{F}$ is not yet defined.

This completely specifies $\mathcal{F}_S$ as a function $\mathbb{F}^S \to \mathbb{C}$. We show that since $\mathcal{V}$ is a k-local subspace, $\mathcal{F}_S$ is in fact a distribution. First, $\sum_{f \in \mathbb{F}^S} \mathcal{F}_S(f) = 1$ because $\widehat{\mathcal{F}_S}(0^S) = 1/q^{|S|}$ since $\mathcal{V}$ is a k-local subspace, and thus must contain 0^n. Hence, it suffices to show that $\mathcal{F}_S(f) \in \mathbb{R}_{\geq 0}$ for each $f \in \mathbb{F}^S$. For each $f \in \mathbb{F}^S$ we have

$$\mathcal{F}_S(f) = \sum_{\alpha \in \mathbb{F}^S} \widehat{\mathcal{F}_S}(\alpha) \chi_\alpha(f) = \sum_{\alpha \in \mathcal{V}_{\subseteq S}} \widehat{\mathcal{F}_S}(\alpha) \chi_\alpha(f) ,$$

since we have defined $\mathcal{F}_S$ in this way using its Fourier coefficients. There are two cases: either $\langle \alpha, f \rangle = 0$ for every $\alpha \in \mathcal{V}_{\subseteq S}$, in which case the sum is $|\mathcal{V}_{\subseteq S}|/q^{|S|}$, or $\langle \alpha, f \rangle \neq 0$ for some $\alpha \in \mathcal{V}_{\subseteq S}$. In the latter case, we use the fact that $\mathcal{V}_{\subseteq S}$ is a linear subspace to show that the sum is 0. In either case, we conclude that $\mathcal{F}_S(f)$ is a non-negative real number, and therefore that $\mathcal{F}_S$ is a distribution.

Next, we argue that the collection of local distributions $\{\mathcal{F}_S\}_{S \subseteq [n]_{\leq k}}$ is indeed non-signaling. This follows from a lemma that we prove that shows that a collection of local distributions is non-signaling if and only if the Fourier coefficients of the local distributions (after removing the normalization factors) are the same. Thus the k-non-signaling function $\mathcal{F}$ is well-defined.

Finally, we show that $\mathcal{F}$ satisfies the desired properties. This follows from our definition of each $\mathcal{F}_S$, as the construction implies that the Fourier coefficients of $\mathcal{F}$ satisfy:

$$\widehat{\mathcal{F}}(\alpha) := \begin{cases} \frac{1}{q^n} & \text{if } \alpha \in \mathcal{V} \\ 0 & \text{if } \alpha \notin \mathcal{V} \end{cases} .$$

This corresponds to having $\Pr[\langle \alpha, \mathcal{F} \rangle = 0] = 1$ for every $\alpha \in \mathcal{V}$, and the random variable $\langle \alpha, \mathcal{F} \rangle$ having the uniform distribution when $\alpha \notin \mathcal{V}$, which completes the proof.

2.4 Low-degree testing fails for small locality

We discuss how to prove Theorem 3. We let $\mathbf{C}$ denote the linear code of polynomials $f \colon \mathbb{F}^m \to \mathbb{F}$ of total degree-d, and let T be the set of all $\alpha \in \mathbf{C}^\perp$ whose support is contained in a line in $\mathbb{F}^m$. Note that for any low-degree test that only checks line constraints and has perfect completeness, if we let T_0 be the set of constraints checked by this test, we will have $T_0 \subseteq T$.

The main combinatorial quantity that we use in our proof is the *rank* of an element $\alpha \in \mathbf{C}^\perp$, defined as

$$\mathsf{rank}_T(\alpha) := \min_{T' \subseteq T : \alpha \in \mathrm{span}(T')} |T'| \ .$$

Note that $\mathsf{rank}_T(\alpha)$ is a non-negative integer, as $\mathrm{span}(T) = \mathbf{C}^\perp$.

We now sketch the proof of Theorem 3 in three steps.

(1) Interval Cut Lemma. We show a generic lemma about the relationship between rank and provability from T. Informally, we show that in order for T to prove α of rank at least r, T must also prove some β of "intermediate" rank. Formally, we show that if there is an interval $[r/2, r)$ such that every β with rank in this interval is *not* provable from T, then every α of rank at least r is also not provable from T. We prove this *Interval Cut Lemma* via the fact that rank_T is subadditive, that is, $\mathsf{rank}_T(\alpha + \beta) \leq \mathsf{rank}_T(\alpha) + \mathsf{rank}_T(\beta)$. Subadditivity implies that for every interval $[r/2, r)$, in order to prove a constraint of rank $\geq r$ from constraints of rank $< r/2$ there must be an intermediate constraint β with rank in $[r/2, r)$ bridging the gap.

(2) Two combinatorial facts. We prove two combinatorial facts about the dual code of $\mathbf{C}$.

- There exists $\alpha^* \in \mathbf{C}^\perp$ where $\mathrm{wt}(\alpha) = 2d+2$ and $\mathrm{supp}(\alpha) \subseteq \{(a, a^2, 0^{m-2}) : a \in \mathbb{F}\} \subseteq \mathbb{F}^m$, i.e., $\mathrm{supp}(\alpha)$ is contained along the curve $x_1^2 - x_2 = 0$ embedded on the plane $x_3 = x_4 = \cdots = x_m = 0$ of $\mathbb{F}^m$.
  Proof sketch. If f is an m-variate polynomial of total degree d then $f(t, t^2, 0, \ldots, 0)$ is a polynomial of degree $\leq 2d$ in t. Thus, there is an element $\alpha^* \in \mathbf{C}^\perp$ supported on this curve of weight $2d+2$ that checks some linear constraint. This shows the existence of the desired α^*.
- For every $\beta \in \mathbf{C}^\perp$ with $\mathsf{rank}_T(\beta) \in \{(d+2)/4, \ldots, (d+2)/2\}$ it holds that $\mathrm{wt}(\beta) \geq \frac{3}{16}(d+2)^2$.
  Proof sketch. Any β of rank r is the sum of *exactly* r line constraints, where each constraint is on a *distinct* line. Each new constraint adds at least $d+2$ weight to β, ignoring the weight that is removed by cancellation. The amount of cancellation is at most the number of intersection points, which is not too large when r is in $\{(d+2)/4, \ldots, (d+2)/2\}$, thus implying that $\mathrm{wt}(\beta) \geq \frac{3}{16}(d+2)^2$.

(3) Completing the proof. Theorem 3 follows from the Interval Cut Lemma, the two combinatorial facts, and Theorem 8. Any $\beta \in \mathbf{C}^\perp$ with rank in $[(d+2)/4, (d+2)/2)$ has weight $\geq \frac{3}{16}(d+2)^2$, and thus is *not* provable when $k < \frac{3}{16}(d+2)^2$. Since α^* has weight $2d+2$ and is supported only on the diagonal, it has rank $\geq d+1$, as each line constraint increases the number of points on the curve by at most 2, by Bézout's theorem. The Interval Cut Lemma implies that α^* is also not provable. The non-signaling function constructed in the proof of Theorem 8 thus passes the random lines test with probability 1 yet satisfies α^* with probability only $1/|\mathbb{F}|$. But, α^* must be satisfied with probability 1 by any non-signaling function that is "locally low-degree", which completes the proof.

2.5 Fourier spectrum of non-signaling linear codes

We have so far adopted the definition that a k-non-signaling function $\mathcal{F}$ is "in" a linear code $\mathbf{C} \subseteq \mathbb{F}^n$ if a function $f \colon S \to \mathbb{F}$ sampled from $\mathcal{F}_S$ is in $\mathbf{C}|_S$ with probability 1 for every $S \subseteq [n]_{\leq k}$. Indeed, we use this "$\mathbf{C}$-explainability" to define the notion of a *local characterization* (see Definition 6).

We now provide thorough justification for this choice. We view the definitions and results below as a conceptual contribution that sheds light on basic properties of non-signaling functions.

In the classical setting, a function $f\colon [n] \to \mathbb{F}$ "looks like" a codeword of $\mathbf{C}$ if, well, it equals some codeword in $\mathbf{C}$. The issue at hand is that, in the non-signaling setting, it is not immediately clear what it means for a non-signaling function $\mathcal{F}$ to be "in" $\mathbf{C}$ because $\mathcal{F}$ is a collection of local distributions. Below are two natural ways to capture this notion.

▶ **Definition 15** (informal). *Let $\mathcal{F}\colon [n] \to \mathbb{F}$ be a k-non-signaling function.*

- *We say that $\mathcal{F}$ is* **C-supported** *if it is equivalent to a k-local quasi-distribution $\mathcal{Q}\colon \mathbb{F}^n \to \mathbb{C}$ that is supported on $\mathbf{C}$, namely, $\mathcal{Q}(f) = 0$ for all $f \notin \mathbf{C}$.*[4]
- *We say that $\mathcal{F}$ is* **C-explainable** *if, for all $S \subseteq [n]_{\leq k}$, the distribution $\mathcal{F}_S$ is supported on $\mathbf{C}|_S$. In other words, the output of $\mathcal{F}$ is always consistent with the restriction of some codeword in $\mathbf{C}$.*

The first definition is motivated by our Equivalence Lemma (Lemma 9), and imposes a "global" property on the non-signaling function. The second definition, implied by the first one, instead takes a "local" approach, imposing consistency with relevant restrictions of the code.

In the following lemma, we quantify the difference between the notions of "**C**-supported" and "**C**-explainable" by characterizing the Fourier spectrum in each case. For convenience, we denote by $\mathbf{C}^{\perp}_{\leq k}$ the set $\{\alpha \in \mathbf{C}^{\perp} : \mathrm{wt}(\alpha) \leq k\}$, which are the constraints with at most k non-zero entries.

▶ **Lemma 16** (informal). *Let $\mathcal{F}\colon [n] \to \mathbb{F}$ be a k-non-signaling function.*

- *$\mathcal{F}$ is $\mathbf{C}$-supported $\leftrightarrow$ the Fourier coefficients $\{\widehat{\mathcal{F}}(\alpha)\}_{\alpha \in \mathbb{F}^n_{\leq k}}$ are constant on each coset of $\mathbf{C}^{\perp}$.*
- *$\mathcal{F}$ is $\mathbf{C}$-explainable $\leftrightarrow$ the Fourier coefficient $\widehat{\mathcal{F}}(\alpha)$ equals $\frac{1}{q^n}$ for every $\alpha \in \mathbf{C}^{\perp}_{\leq k}$.*

We additionally prove that the foregoing structure is robust to errors: $\mathcal{F}$ is close to being $\mathbf{C}$-supported if and only if its Fourier coefficients are almost constant on every coset of $\mathbf{C}^{\perp}$; moreover $\mathcal{F}$ is close to being $\mathbf{C}$-explainable if and only if $\widehat{\mathcal{F}}(\alpha)$ is close to $\frac{1}{q^n}$ for every $\alpha \in \mathbf{C}^{\perp}_{\leq k}$.

One may interpret Lemma 16 as "bad news" because it shows that the notions of "**C**-supported" and "**C**-explainable" are in fact *distinct*. Which one is the correct one to use? From the perspective of local testability, we may regard "**C**-supported" as more desirable, because it requires a global structure to hold. We prove that, fortunately, the two notions are equivalent up to a small change in parameters, reinforcing our belief that we have identified the right notions.

▶ **Lemma 17** (informal). *Let $\mathcal{F}\colon [n] \to \mathbb{F}$ be a k-non-signaling function.*

- *If $\mathcal{F}$ is $\mathbf{C}$-supported, then $\mathcal{F}$ is $\mathbf{C}$-explainable.*
- *If $\mathcal{F}$ is $\mathbf{C}$-explainable, then $\mathcal{F}$ (viewed as a $k/2$-non-signaling function) is $\mathbf{C}$-supported.*

In light of the above, it suffices to study non-signaling functions that are **C**-explainable. We have used this notion in our results on local characterizations (see Definition 6), as it is more natural in this setting: the set of **C**-explainable k-non-signaling functions are precisely those that are consistent with the set of constraints $\mathbf{C}^{\perp}_{\leq k}$.

[4] When $\mathbf{C}$ is the Hadamard code, this definition equals the notion of a *linear* non-signaling function from [4].

Detailed definitions and proofs can be found in Section 5. Below we provide proof sketches for Lemmas 16 and 17. The Fourier structure of non-signaling functions, discussed in Section 2.1, underlies all of these proofs.

2.5.1 Fourier spectrum of a C-supported function

We outline the proof of the first item of Lemma 16. A k-non-signaling function $\mathcal{F}$ that is $\mathbf{C}$-supported is by definition equivalent to a quasi-distribution $\mathcal{Q}$ supported on $\mathbf{C}$. We explain why all such non-signaling functions have Fourier coefficients that are constant on cosets of $\mathbf{C}^\perp$, that is, $\widehat{\mathcal{F}}(\alpha) = \widehat{\mathcal{F}}(\alpha')$ for every $\alpha, \alpha' \in \mathbb{F}^n_{\leq k}$ with $\alpha - \alpha' \in \mathbf{C}^\perp$. We compare the following two affine spaces:

$$V_1 = \left\{ \mathcal{Q}\colon \mathbb{F}^n \to \mathbb{C} \text{ s.t. } \sum_{f \in \mathbf{C}} \mathcal{Q}(f) = 1 \text{ and } \mathcal{Q}(f) = 0 \ \forall f \notin \mathbf{C} \right\} ,$$

$$V_2 = \left\{ \mathcal{Q}\colon \mathbb{F}^n \to \mathbb{C} \text{ s.t. } \widehat{\mathcal{Q}}(0^n) = \frac{1}{q^n} \text{ and } \widehat{\mathcal{Q}}(\alpha) = \widehat{\mathcal{Q}}(\alpha + \gamma) \ \forall \alpha \in \mathbb{F}^n, \gamma \in \mathbf{C}^\perp \right\} .$$

The affine space V_1 corresponds to quasi-distributions that are supported on $\mathbf{C}$, while V_2 corresponds to quasi-distributions whose Fourier coefficients satisfy the desired characterization. It suffices to prove that $V_1 = V_2$. First we show that $\dim(V_1) = \dim(V_2)$, and then that $V_1 \subseteq V_2$.

The dimension of V_1 is $|\mathbf{C}| - 1$ because the $|\mathbf{C}|$ free terms are subject to a single linear constraint. The dimension of V_2 is $q^n/\left|\mathbf{C}^\perp\right| - 1$ because the Fourier coefficients are constant on each coset of $\mathbf{C}^\perp$, and on each coset they may have an arbitrary value; the one exception is the coset $\mathbf{C}^\perp$, where the Fourier coefficients must be $\frac{1}{q^n}$. Recalling that $q^n = |\mathbf{C}| \cdot \left|\mathbf{C}^\perp\right|$, we deduce that $\dim(V_1) = \dim(V_2)$.

Next we show that $V_1 \subseteq V_2$. For any $\mathcal{Q} \in V_1$ and $\alpha \in \mathbb{F}^n$ we have by definition

$$\widehat{\mathcal{Q}}(\alpha) := \frac{1}{q^n} \cdot \sum_{f \in \mathbb{F}^n} \mathcal{Q}(f) \cdot \omega^{-\mathrm{Tr}(\langle \alpha, f \rangle)} .$$

Since $\mathcal{Q} \in V_1$, any function f in the support of $\mathcal{Q}$ must be in $\mathbf{C}$. Therefore, for any $\gamma \in \mathbf{C}^\perp$ have $\langle \gamma, f \rangle = 0$, so that $\omega^{\mathrm{Tr}(\langle \gamma, f \rangle)} = \omega^{\mathrm{Tr}(0)} = 1$. This implies that $\widehat{\mathcal{Q}}(\alpha) = \widehat{\mathcal{Q}}(\alpha + \gamma)$. Intuitively, when we shift α by γ the sum remains unchanged because each term in the sum is multiplied by $\omega^{-\mathrm{Tr}(\langle \gamma, f \rangle)} = 1$. Thus $V_1 \subseteq V_2$. Since $\dim(V_1) = \dim(V_2)$ and $V_1 \subseteq V_2$, we conclude that $V_1 = V_2$.

2.5.2 Fourier spectrum of a C-explainable function

We outline the proof of the second item of Lemma 16. The characterization of $\mathbf{C}$-explainable functions relies on the fact that the Fourier coefficient $\widehat{\mathcal{F}}(\alpha)$ is related to the distribution of the random variable $\langle \alpha, \mathcal{F} \rangle$, i.e., the distribution $(\Pr[\langle \alpha, \mathcal{F} \rangle = b])_{b \in \mathbb{F}}$. This intuition can be quantified via (a generalization of) the DFT matrix $M \in \mathbb{C}^{q \times q}$, which is the matrix defined as $M_{a,b} := \omega^{-\mathrm{Tr}(ab)}$ (entries are indexed by $\mathbb{F}$); M is invertible and $\frac{1}{\sqrt{q}} M$ is unitary.

Recall that the Fourier coefficients of $\mathcal{F}$ are defined as follows:

$$\forall\, \alpha \in \mathbb{F}^n_{\leq k} \quad \widehat{\mathcal{F}}(\alpha) := \frac{1}{q^n} \sum_{j \in \mathbb{F}_p} \omega^{-j} \Pr[\mathrm{Tr}(\langle \alpha, \mathcal{F} \rangle) = j] .$$

Letting $v := (\Pr[\langle \alpha, \mathcal{F} \rangle = b])_{b \in \mathbb{F}}$, expanding the definitions shows that $Mv = (q^n \widehat{\mathcal{F}}(a\alpha))_{a \in \mathbb{F}}$. The linear transformation M thus quantifies the relation between the distribution $(\Pr[\langle \alpha, \mathcal{F} \rangle = b])_{b \in \mathbb{F}}$ and the Fourier coefficients $(q^n \widehat{\mathcal{F}}(a\alpha))_{a \in \mathbb{F}}$.

Now, given a k-non-signaling function $\mathcal{F}$, we first show that $\mathcal{F}$ is $\mathbf{C}$-explainable if and only if $\Pr[\langle \alpha, \mathcal{F} \rangle = 0] = 1$ for every $\alpha \in \mathbf{C}^{\perp}_{\leq k}$. This follows from the fact that any local function $g \colon S \to \mathbb{F}$ that satisfies every $\alpha \in \mathbf{C}^{\perp}_{\subseteq S}$ can be extended into a codeword $f \in \mathbf{C}$. Using the matrix M, we can relate the condition that $\mathcal{F}$ satisfies every $\alpha \in \mathbf{C}^{\perp}_{\leq k}$ with probability 1 to its Fourier spectrum. Specifically, we have that $\Pr[\langle \alpha, \mathcal{F} \rangle = 0] = 1$ if and only if $(q^n \widehat{\mathcal{F}}(a\alpha))_{a \in \mathbb{F}} = M(1, 0, \ldots, 0)^{\top}$. Since $M(1, 0, \ldots, 0)^{\top} = (1, \ldots, 1)^{\top}$, we get that $\Pr[\langle \alpha, \mathcal{F} \rangle = 0] = 1$ if and only if $\widehat{\mathcal{F}}(a\alpha) = \frac{1}{q^n}$ for every $a \in \mathbb{F}$, completing the proof.

2.5.3 The relationship between C-supported and C-explainable

We outline the proof of Lemma 17. First note that Lemma 16 immediately implies that a $\mathbf{C}$-supported k-non-signaling function $\mathcal{F}$ is $\mathbf{C}$-explainable, because if $\mathcal{F}$ is $\mathbf{C}$-supported then $\widehat{\mathcal{F}}(\alpha) = \widehat{\mathcal{F}}(0^n) = \frac{1}{q^n}$ for every $\alpha \in \mathbf{C}^{\perp}_{\leq k}$, implying that $\mathcal{F}$ is $\mathbf{C}$-explainable.

Conversely, if $\mathcal{F}$ is $\mathbf{C}$-explainable, then for any $\alpha, \alpha' \in \mathbb{F}^n_{\leq k/2}$ with $\alpha - \alpha' \in \mathbf{C}^{\perp}$ we get that for any $b \in \mathbb{F}$,

$$\Pr[\langle \alpha, \mathcal{F} \rangle = b] = \Pr[\langle \alpha', \mathcal{F} \rangle + \langle \alpha - \alpha', \mathcal{F} \rangle = b] = \Pr[\langle \alpha', \mathcal{F} \rangle = b] \ ,$$

since $\Pr[\langle \alpha - \alpha', \mathcal{F} \rangle = 0] = 1$ as $\alpha - \alpha' \in \mathbf{C}^{\perp}$ and $\mathcal{F}$ is $\mathbf{C}$-explainable. This shows that the vectors $(\Pr[\langle \alpha, \mathcal{F} \rangle = b])_{b \in \mathbb{F}}$ and $(\Pr[\langle \alpha', \mathcal{F} \rangle = b])_{b \in \mathbb{F}}$ are equal, which implies that the Fourier coefficients $\widehat{\mathcal{F}}(\alpha)$ and $\widehat{\mathcal{F}}(\alpha')$ are equal. By Lemma 16, this completes the proof. Note that we crucially need $\mathrm{wt}(\alpha), \mathrm{wt}(\alpha') \leq k/2$ so that $\mathrm{wt}(\alpha - \alpha') \leq k$, as otherwise $\Pr[\langle \alpha - \alpha', \mathcal{F} \rangle = 0]$ is undefined.

3 Preliminaries

Throughout this paper we let $n \in \mathbb{N}$ be an arbitrary positive integer, and $k \in \mathbb{N}$ a positive integer that is at most n. We use $\mathbb{F}$ to denote the finite field of size q with characteristic p, and $\mathbb{F}_p$ to denote the prime subfield of $\mathbb{F}$. We often abuse notation and identify a function $f \colon [n] \to \mathbb{F}$ with its evaluation table in $\mathbb{F}^n$. For a vector $\alpha \in \mathbb{F}^n$ we let $\mathrm{supp}(\alpha) := \{i \in [n] : \alpha_i \neq 0\}$, and we let $\mathrm{wt}(\alpha) := |\mathrm{supp}(\alpha)|$. For a set of vectors $R \subseteq \mathbb{F}^n$, we let $R_{\leq \ell} \subseteq R$ denote the subset $\{\alpha \in R : \mathrm{wt}(\alpha) \leq \ell\}$. In particular, $\mathbb{F}^n_{\leq k}$ contains all vectors $\alpha \in \mathbb{F}^n$ of weight at most k. For a subset $S \subseteq [n]$, we let $R_{\subseteq S} = \{\alpha \in R : \mathrm{supp}(\alpha) \subseteq S\}$; we also write $S \subseteq [n]_{\leq \ell}$ if $|S| \leq \ell$.

3.1 Non-signaling functions

We define *non-signaling functions* and *quasi-distributions*, and introduce useful notation for them. The definitions are almost identical to those in [4], but extended to any finite field.

▶ **Definition 18.** *A* **k-non-signaling function** $\mathcal{F} \colon [n] \to \mathbb{F}$ *is a collection* $\mathcal{F} = \{\mathcal{F}_S\}_{S \subseteq [n]_{\leq k}}$ *where (i) each* $\mathcal{F}_S$ *is a distribution over functions* $f \colon S \to \mathbb{F}$*, and (ii) for every two subsets* S *and* R *each of size at most* k*, the restrictions of* $\mathcal{F}_S$ *and* $\mathcal{F}_R$ *to* $S \cap R$ *are equal as distributions.*

Note that any function $f \colon [n] \to \mathbb{F}$ induces a n-non-signaling function by setting $\mathcal{F}_S$ to be the distribution that outputs $f|_S$ with probability 1. More generally, any distribution $\mathcal{D}$ over functions $f \colon [n] \to \mathbb{F}$ induces a corresponding n-non-signaling function by defining $\mathcal{F}_S$ to be the distribution that samples $f \leftarrow \mathcal{D}$ and outputs $f|_S$.

Given a set $S \subseteq [n]_{\leq k}$ and function $g \in \mathbb{F}^S$, we define

$$\Pr\left[\, \mathcal{F}(S) = g \,\right] := \Pr[\, g \leftarrow \mathcal{F}_S \,] \ .$$

The non-signaling property in this notation is the following: for every two subsets $S, R \subseteq [n]_{\leq k}$ and every string $g \in \mathbb{F}^{S \cap T}$, $\Pr[\mathcal{F}(S)|_{S\cap T} = g] = \Pr[\mathcal{F}(T)|_{S\cap T} = g]$, where the probability is over the randomness of $\mathcal{F}$.

We extend the above notation to every $E \subseteq \mathbb{F}^S$ in the natural way by defining $\Pr[\mathcal{F}(S) \in E]$ to be $\Pr_{f \leftarrow \mathcal{F}_S}[f \in E]$. We highlight the case when E is an "inner product event", as we will encounter this case frequently.

▶ **Definition 19.** *Let $\mathcal{F}\colon [n] \to \mathbb{F}$ be a k-non-signaling function. For $\alpha \in \mathbb{F}^n_{\leq k}$ and $b \in \mathbb{F}$, we define*

$$\Pr[\langle \alpha, \mathcal{F} \rangle = b] := \Pr_{f \leftarrow \mathcal{F}_{\mathrm{supp}(\alpha)}} \left[\sum_{i \in \mathrm{supp}(\alpha)} \alpha_i f(i) = b \right] .$$

Similarly, we define $\Pr[\mathrm{Tr}(\langle \alpha, \mathcal{F} \rangle) = j] := \sum_{b \in \mathbb{F}: \mathrm{Tr}(b) = j} \Pr[\langle \alpha, \mathcal{F} \rangle = b]$ for every $j \in \mathbb{F}_p$.

The probability above is well-defined since $\mathrm{wt}(\alpha) \leq k$, and so we query $\mathcal{F}$ on at most k points.

Since $\mathcal{F}$ is non-signaling, $\Pr[\langle \alpha, \mathcal{F} \rangle = b] = \Pr_{f \leftarrow \mathcal{F}_S}[\sum_{i \in S} \alpha_i f(i)]$ for any set $S \supseteq \mathrm{supp}(\alpha)$. The intuition behind the above definition is that the inner product $\langle \alpha, g \rangle$ for any $g\colon [n] \to \mathbb{F}$ can be computed only given $g|_{\mathrm{supp}(\alpha)}$, namely, given g restricted to a set of size at most k.

3.2 Quasi-distributions

A quasi-distribution extends the notion of a probability distribution by allowing probabilities to be complex, and is the main tool that we use to analyze non-signaling functions.

▶ **Definition 20.**

- *A* **quasi-distribution** *is a function $\mathcal{Q}\colon \mathbb{F}^n \to \mathbb{C}$ where $\sum_{f \in \mathbb{F}^n} \mathcal{Q}(f) = 1$.*
- *For a set of functions $R \subseteq \mathbb{F}^n$, we say that $\mathcal{Q}$ is* **supported** *on R if $\{f \in \mathbb{F}^n : \mathcal{Q}(f) \neq 0\} \subseteq R$.*
- *For a positive integer ℓ, we say that $\mathcal{Q}$ is* **ℓ-local** *if the marginals $\mathcal{Q}|_S$ for each $S \subseteq [n]_{\leq \ell}$ are distributions ($\sum_{f \in \mathbb{F}^n : f|_S = g} \mathcal{Q}(f)$ is a non-negative real number for each $S \subseteq [n]_{\leq \ell}$ and $g\colon S \to \mathbb{F}$).*

If $\mathcal{Q}$ is ℓ-local, then for every subset $S \subseteq [n]_{\leq \ell}$, we may view $\mathcal{Q}|_S$ as a probability distribution over $\mathbb{F}^S$. If $\mathcal{Q}$ is ℓ-local then it is s-local for every $s \in \{0, 1, \ldots, \ell\}$.

▶ **Definition 21.** *Given a quasi-distribution $\mathcal{Q}$, a subset $S \subseteq [n]$, and $g \in \mathbb{F}^S$, we define the* **quasi-probability** *of the event "$\mathcal{Q}(S) = g$" to be the following complex number*

$$\widetilde{\Pr}[\mathcal{Q}(S) = g] := \sum_{f \in \mathbb{F}^n : f|_S = g} \mathcal{Q}(f) .$$

(The tilde above Pr *denotes that quasi-probabilities are not necessarily non-negative real numbers.)*

Given a subset $E \subseteq \mathbb{F}^S$, we similarly define $\widetilde{\Pr}[\mathcal{Q}(S) \in E] := \sum_{f \in \mathbb{F}^n : f|_S \in E} \mathcal{Q}(f)$.

As for non-signaling functions, we highlight the case when E is an inner product event.

▶ **Definition 22.** *Let $\mathcal{Q}\colon \mathbb{F}^n \to \mathbb{C}$ be a quasi-distribution. For $\alpha \in \mathbb{F}^n$ and $b \in \mathbb{F}$, we define*

$$\widetilde{\Pr}[\langle \alpha, \mathcal{Q} \rangle = b] := \sum_{f \in \mathbb{F}^n : \langle \alpha, f \rangle = b} \mathcal{Q}(f) .$$

Similarly, we define $\widetilde{\Pr}[\mathrm{Tr}(\langle \alpha, \mathcal{Q} \rangle) = j] := \sum_{b \in \mathbb{F}: \mathrm{Tr}(b) = j} \widetilde{\Pr}[\langle \alpha, \mathcal{Q} \rangle = b]$ for every $j \in \mathbb{F}_p$.

▶ **Definition 23** (statistical distance). *Given a finite domain $[n]$ and an integer $\ell \in \{1,\ldots,|D|\}$, the* **$\Delta_\ell$-distance** *between two quasi-distributions $\mathcal{Q}$ and $\mathcal{Q}'$ is*

$$\Delta_\ell(\mathcal{Q},\mathcal{Q}') := \max_{S\subseteq[n]_{\leq\ell}} \Delta(\mathcal{Q}|_S,\mathcal{Q}'|_S) \ ,$$

where $\Delta(\mathcal{Q}|_S,\mathcal{Q}'|_S) := \max_{E\subseteq\mathbb{F}^S} \left|\widetilde{\Pr}[\mathcal{Q}(S)\in E] - \widetilde{\Pr}[\mathcal{Q}'(S)\in E]\right|$.

We say that $\mathcal{Q}$ and $\mathcal{Q}'$ are **ε-close in the Δ_ℓ-distance** *if $\Delta_\ell(\mathcal{Q},\mathcal{Q}') \leq \varepsilon$; else, they are ε-far.*

▶ Remark 24 (distance for non-signaling functions). The definition of Δ_ℓ-distance naturally extends to defining distances between k-non-signaling functions, as well as between quasi-distributions and k-non-signaling functions, provided that $\ell \leq k$.

The notion above generalizes the standard notion of statistical (total variation) distance: if $\mathcal{Q}$ and $\mathcal{Q}'$ are *distributions* then their Δ_n-distance equals their statistical distance. Also note that if $\mathcal{Q}$ and $\mathcal{Q}'$ are ℓ-local quasi-distributions then their Δ_ℓ-distance equals the maximum statistical distance, across all subsets $S \subseteq [n]$ with $|S| \leq \ell$, between the two *distributions* $\mathcal{Q}|_S$ and $\mathcal{Q}'|_S$ – in particular this means that any experiment that queries exactly one set of size at most ℓ cannot distinguish between the two quasi-distributions with probability greater than $\Delta_\ell(\mathcal{Q},\mathcal{Q}')$.

We stress that $\Delta_\ell(\mathcal{Q},\mathcal{Q}') = 0$ does *not* necessarily mean that $\mathcal{Q} = \mathcal{Q}'$! In fact, it is possible to have $\Delta_\ell(\mathcal{Q},\mathcal{Q}') = 0$ while $\sum_{f\in U} |\mathcal{Q}(f) - \mathcal{Q}'(f)|$ is arbitrarily large. We also remark that the Δ_ℓ-distance is not necessarily upper bounded by 1, and is in general unbounded.

▶ **Definition 25** (approximate locality). *Given a finite domain $[n]$, an integer $\ell \in \{1,\ldots,n\}$, and a real number $\varepsilon \geq 0$, a quasi-distribution $\mathcal{Q}$ over U_n is* **(ℓ,ε)-local** *if, for every subset $S \subseteq [n]_{\leq\ell}$ and every event $E \subseteq \{0,1\}^S$,*

$$\min_{x\in[0,1]} \left\{\left|\widetilde{\Pr}[\mathcal{Q}(S)\in E] - x\right|\right\} \in [0,\varepsilon] \ .$$

Approximate locality generalizes the notion of (exact) locality as in Definition 20. Below, we state a lemma that if$\mathcal{Q}$ is (ℓ,ε)-local and is supported over a linear code $\mathbf{C}$, then there is an ℓ-local $\mathcal{Q}'$ over $\mathbf{C}$ that is close to $\mathcal{Q}$. The proof idea is similar to that of "smoothening" almost-feasible solutions to Sherali–Adams relaxations into feasible ones [13].

▶ **Lemma 26.** *If $\mathcal{Q}$ is a (ℓ,ε)-local quasi-distribution over $\mathbf{C}$, then there is an ℓ-local quasi-distribution $\mathcal{Q}'$ over $\mathbf{C}$ such that $\Delta_\ell(\mathcal{Q},\mathcal{Q}') < q^\ell\varepsilon$.*

We omit the proof of Lemma 26 as it is identical to the proof of lemma 7.8 in [4], just replacing the field $\mathbb{F}_2 = \{0,1\}$ with a general field $\mathbb{F}$.

4 Fourier analysis of non-signaling functions

We prove statements about the Fourier structure of non-signaling functions, and prove the Equivalence Lemma. In Section 4.1 we recall basic facts about Fourier analysis of functions over finite fields. In Section 4.2 we relate Fourier coefficients to probabilities and quasi-probabilities. In Section 4.3 we prove that non-signaling functions and quasi-distributions are equivalent notions.

4.1 Fourier analysis of functions over finite fields

We consider functions of the type $F\colon \mathbb{F}^n \to \mathbb{C}$. For two such functions F_1 and F_2, we define their inner product as $\langle F_1, F_2\rangle := \frac{1}{q^n}\sum_{x\in\mathbb{F}^n} \overline{F_1(x)}F_2(x)$. For every $\alpha \in \mathbb{F}^n$, we define the *character* $\chi_\alpha\colon \mathbb{F}^n \to \mathbb{C}$ as $\chi_\alpha(x) := \omega^{\mathrm{Tr}(\langle\alpha,x\rangle)}$ where: (1) $\mathrm{Tr}\colon \mathbb{F} \to \mathbb{F}_p$ is the trace map; (2) $\langle\alpha, x\rangle$ is the inner product $\sum_{i=1}^n \alpha_i x_i$; (3) $\omega = e^{2\pi i/p}$ is a primitive complex p-th root of unity; and (4) ω^j is defined by thinking of $j \in \mathbb{F}_p$ as an integer in $\mathbb{Z}$. The functions $\{\chi_\alpha\}_{\alpha\in\mathbb{F}^n}$ form an orthonormal basis of the space of all functions $f\colon \mathbb{F}^n \to \mathbb{C}$, so every function $F\colon \mathbb{F}^n \to \mathbb{C}$ can be written as

$$F(\cdot) = \sum_{\alpha\in\mathbb{F}^n} \widehat{F}(\alpha)\chi_\alpha(\cdot) \ , \text{ where } \widehat{F}(\alpha) := \langle\chi_\alpha, F\rangle \ .$$

The values $\{\widehat{F}(\alpha)\}_{\alpha\in\mathbb{F}^n}$ are the *Fourier coefficients* of F. We recall and prove a few useful identifies.

Parseval's identity. For every two functions $F, G\colon \mathbb{F}^n \to \mathbb{C}$,

$$\langle F, G\rangle = \frac{1}{q^n}\sum_{x\in\mathbb{F}^n} \overline{F(x)}G(x) = \sum_{\alpha\in\mathbb{F}^n} \overline{\widehat{F}(\alpha)}\widehat{G}(\alpha) \ .$$

Proof.

$$\frac{1}{q^n}\sum_{x\in\mathbb{F}^n} \overline{F(x)}G(x) = \frac{1}{q^n}\sum_{x\in\mathbb{F}^n} \overline{\left(\sum_{\alpha\in\mathbb{F}^n} \widehat{F}(\alpha)\chi_\alpha(x)\right)}\left(\sum_{\beta\in\mathbb{F}^n} \widehat{G}(\beta)\chi_\beta(x)\right)$$
$$= \sum_{\alpha\in\mathbb{F}^n}\sum_{\beta\in\mathbb{F}^n} \overline{\widehat{F}(\alpha)}\widehat{G}(\beta)\langle\chi_\alpha,\chi_\beta\rangle = \sum_{\alpha\in\mathbb{F}^n} \overline{\widehat{F}(\alpha)}\widehat{G}(\alpha) \ ,$$

since $\{\chi_\alpha\}_{\alpha\in\mathbb{F}^n}$ are orthonormal. ◀

Plancherel's identity. As a corollary of the above,

$$\frac{1}{q^n}\sum_{x\in\mathbb{F}^n} |F(x)|^2 = \sum_{\alpha\in\mathbb{F}^n} |\widehat{F}(\alpha)|^2 \ .$$

The case of indicator functions. When analyzing non-signaling functions and quasi-distributions we will apply the above identities in the case where F is an indicator function $\mathbf{1}_E$ for a set $E \subseteq \mathbb{F}^n$. In this case, by Plancherel's identity we have that $|E|/q^n = \sum_{\alpha\in\mathbb{F}^n} |\widehat{\mathbf{1}_E}(\alpha)|^2$. In particular, by the Cauchy–Schwarz inequality, this implies that

$$\|\widehat{\mathbf{1}_E}\|_1 = \sum_{\alpha\in\mathbb{F}^n} |\widehat{\mathbf{1}_E}(\alpha)| \le \sqrt{\sum_{\alpha\in\mathbb{F}^n} |\widehat{\mathbf{1}_E}(\alpha)|^2}\cdot\sqrt{\sum_{\alpha\in\mathbb{F}^n} 1} \le \sqrt{|E|/q^n}\cdot q^{n/2} = \sqrt{|E|} \ .$$

If we let $F(x) = \mathbf{1}_E(x)$, then Parseval's identity becomes the following lemma.

▶ **Lemma 27.** *Let $G\colon \mathbb{F}^n \to \mathbb{C}$ be a function and $E \subseteq \mathbb{F}^n$. Then*

$$\frac{1}{q^n}\sum_{x\in E} G(x) = \frac{1}{q^n}\sum_{\alpha\in\mathbb{F}^n} \overline{\widehat{\mathbf{1}_E}(\alpha)}\sum_{x\in\mathbb{F}^n} G(x)\omega^{-\mathrm{Tr}(\langle\alpha,x\rangle)} = \sum_{\alpha\in\mathbb{F}^n} \overline{\widehat{\mathbf{1}_E}(\alpha)}\widehat{G}(\alpha) \ .$$

4.2 Relating the Fourier spectrum to the probabilities of events

A quasi-distribution $\mathcal{Q}$ is a function $\mathcal{Q}\colon \mathbb{F}^n \to \mathbb{C}$ that maps a function $f\colon [n] \to \mathbb{F}$ (identified with the corresponding vector $\mathbb{F}^n$) to $\mathcal{Q}(f)$. We can write $\mathcal{Q}(\cdot) = \sum_{\alpha\in\mathbb{F}^n} \widehat{\mathcal{Q}}(\alpha)\chi_\alpha(\cdot)$, where $\{\chi_\alpha\}_{\alpha\in\mathbb{F}^n}$ are the characters and $\{\widehat{\mathcal{Q}}(\alpha)\}_{\alpha\in\mathbb{F}^n}$ are $\mathcal{Q}$'s Fourier coefficients. For $S \subseteq [n]$ and $\alpha \in \mathbb{F}^S$, we abuse notation and use $\widehat{\mathcal{Q}}(\alpha)$ to refer to $\widehat{\mathcal{Q}}(\beta)$ where $\beta \in \mathbb{F}^n$ has $\beta_i = \alpha_i$ for all $i \in S$ and 0 otherwise.

The lemma below relates the inner product quasi-probabilities defined in Definition 22 to the Fourier coefficients of $\mathcal{Q}$.

▶ **Lemma 28.** *Let $\mathcal{Q}\colon \mathbb{F}^n \to \mathbb{C}$ be a quasi-distribution. For every $\alpha \in \mathbb{F}^n$,*

$$\widehat{\mathcal{Q}}(\alpha) = \frac{1}{q^n}\sum_{j\in\mathbb{F}_p} \omega^{-j}\widetilde{\Pr}[\mathrm{Tr}(\langle\alpha,\mathcal{Q}\rangle) = j] \ .$$

Proof of Lemma 28. By definition,

$$\begin{aligned}\widehat{\mathcal{Q}}(\alpha) = \langle\chi_\alpha,\mathcal{Q}(\cdot)\rangle &= \frac{1}{q^n}\sum_f \overline{\chi_\alpha(f)}\mathcal{Q}(f) = \frac{1}{q^n}\sum_{j\in\mathbb{F}_p}\omega^{-j}\sum_{f:\chi_\alpha(f)=\omega^j}\mathcal{Q}(f)\\ &= \frac{1}{q^n}\sum_{j\in\mathbb{F}_p}\omega^{-j}\sum_{f:\mathrm{Tr}(\langle\alpha,f\rangle)=j}\mathcal{Q}(f) = \frac{1}{q^n}\sum_{j\in\mathbb{F}_p}\omega^{-j}\widetilde{\Pr}[\mathrm{Tr}(\langle\alpha,\mathcal{Q}\rangle) = j] \ . \end{aligned}$$ ◀

The above lemma implies that the Fourier coefficients $(\widehat{\mathcal{Q}}(a\alpha))_{a\in\mathbb{F}}$ are determined by the quasi-probabilities $(\Pr[\langle\alpha,\mathcal{Q}\rangle = b])_{b\in\mathbb{F}}$, as the quasi-probabilities $(\Pr[\langle\alpha,\mathcal{Q}\rangle = b])_{b\in\mathbb{F}}$ determine the quasi-probabilities $(\Pr[\langle a\alpha,\mathcal{Q}\rangle = b])_{b\in\mathbb{F}}$ for every $a \in \mathbb{F}$. In fact, there is a linear transformation M that maps $(\Pr[\langle\alpha,\mathcal{Q}\rangle = b])_{b\in\mathbb{F}}$ to $(q^n\widehat{\mathcal{Q}}(a\alpha))_{a\in\mathbb{F}}$. Below, we state a well-known lemma about M.

▶ **Lemma 29.** *Let $M \in \mathbb{C}^{q\times q}$ be the matrix defined as $M_{a,b} := \omega^{-\mathrm{Tr}(ab)}$ (entries are indexed by $\mathbb{F}$). Then M is invertible and $\frac{1}{\sqrt{q}}M$ is unitary (namely, $M^\dagger \cdot M = qI$). In particular, for every vector $(v_b)_{b\in\mathbb{F}}$ with values in $\mathbb{C}$, the map $(v_b)_{b\in\mathbb{F}} \mapsto (\sum_{b\in\mathbb{F}}\omega^{-\mathrm{Tr}(ab)}v_b)_{a\in\mathbb{F}}$ is a bijection.*

We additionally prove the following lemma, which relates the Fourier spectrum of the quasi-distribution $\mathcal{Q}|_S$ to the Fourier spectrum of $\mathcal{Q}$.

▶ **Lemma 30.** *Let $\mathcal{Q}\colon \mathbb{F}^n \to \mathbb{C}$ be a quasi-distribution. Let $S \subseteq [n]$, and let $\mathcal{Q}|_S$ denote the restriction of $\mathcal{Q}$ to S, namely, $\mathcal{Q}|_S$ is the quasi-distribution from $\mathbb{F}^S$ to $\mathbb{C}$ where $\mathcal{Q}|_S(g) := \sum_{f:f|_S=g}\mathcal{Q}(f)$. Then for every $\alpha \in \mathbb{F}^S$ it holds that $q^{|S|}\widehat{\mathcal{Q}|_S}(\alpha) = q^n\widehat{\mathcal{Q}}(\alpha)$.*[5]

Proof of Lemma 30.

$$q^{|S|}\widehat{\mathcal{Q}|_S}(\alpha) = \sum_{g\in\mathbb{F}^S}\mathcal{Q}|_S(g)\omega^{-\mathrm{Tr}(\langle\alpha,g\rangle)} = \sum_{f\in\mathbb{F}^n}\mathcal{Q}(f)\omega^{-\mathrm{Tr}(\langle\alpha,f\rangle)} = q^n\widehat{\mathcal{Q}}(\alpha) \ .$$ ◀

If $\mathcal{F}\colon [n] \to \mathbb{F}$ is a k-non-signaling function, then for any $\alpha \in \mathbb{F}^n_{\le k}$ and $b \in \mathbb{F}$ we have defined $\Pr[\langle\alpha,\mathcal{F}\rangle = b]$ in Definition 19 to be $\Pr_{f\leftarrow\mathcal{F}_{\mathrm{supp}(\alpha)}}[\langle\alpha,f\rangle = b]$. Note that the probability is well-defined since $\mathrm{wt}(\alpha) \le k$ (so we query $\mathcal{F}$ on at most k points). Also note that Lemma 28 implies that, for every $\alpha \in \mathbb{F}^n_{\le k}$, we can define the Fourier coefficient $\widehat{\mathcal{F}}(\alpha)$ of $\mathcal{F}$ as

$$\widehat{\mathcal{F}}(\alpha) := \frac{1}{q^n}\sum_{j\in\mathbb{F}_p}\omega^{-j}\Pr[\mathrm{Tr}(\langle\alpha,\mathcal{F}\rangle) = j] \ .$$

[5] The vector α in $\widehat{\mathcal{Q}}(\alpha)$ is treated as a element in $\mathbb{F}^n$ with $\alpha_j = 0$ for all $j \notin S$

With the above definitions, we can prove the following two corollaries of Lemma 27. The first is for non-signaling functions, and the second is for quasi-distributions.

▶ **Corollary 31.** *For any k-non-signaling function $\mathcal{F}\colon [n] \to \mathbb{F}$, set $S \subseteq [n]$, and event $E \subseteq \mathbb{F}^S$,*

$$\Pr[\mathcal{F}(S) \in E] = \sum_{\alpha \in \mathbb{F}^S} \overline{\widehat{\mathbf{1}_E}(\alpha)} \sum_{j \in \mathbb{F}_p} \omega^{-j} \Pr[\mathrm{Tr}(\langle \alpha, \mathcal{F} \rangle) = j] = q^n \sum_{\alpha \in \mathbb{F}^S} \overline{\widehat{\mathbf{1}_E}(\alpha)} \widehat{\mathcal{F}}(\alpha) \ .$$

Proof. Apply Lemma 27 with $G\colon \mathbb{F}^S \to \mathbb{C}$ defined as $G(x) := \Pr[\mathcal{F}_S(i) = x_i \ \forall i \in S]$. ◀

▶ **Corollary 32.** *For any quasi-distribution $\mathcal{Q}\colon \mathbb{F}^n \to \mathbb{C}$, set $S \subseteq [n]$, and event $E \subseteq \mathbb{F}^S$,*

$$\widetilde{\Pr}[\mathcal{Q}(S) \in E] = \sum_{f\colon f(S) \in E} \mathcal{Q}(f) = \sum_{\alpha \in \mathbb{F}^S} \overline{\widehat{\mathbf{1}_E}(\alpha)} \sum_{j \in \mathbb{F}_p} \omega^{-j} \widetilde{\Pr}[\mathrm{Tr}(\langle \alpha, \mathcal{Q} \rangle) = j] = q^n \sum_{\alpha \in \mathbb{F}^S} \overline{\widehat{\mathbf{1}_E}(\alpha)} \widehat{\mathcal{Q}}(\alpha) \ .$$

Proof. Apply Lemma 27 to the function $G\colon \mathbb{F}^S \to \mathbb{C}$ that is the quasi-distribution $\mathcal{Q}|_S$. Then observe that for every $\alpha \in \mathbb{F}^S$, $q^{|S|} \widehat{\mathcal{Q}|_S}(\alpha) = q^n \widehat{\mathcal{Q}}(\alpha)$ by Lemma 30. ◀

The above two lemmas allow us to bound the distance between a k-non-signaling function $\mathcal{F}$ and a quasi-distribution $\mathcal{Q}$ in terms of their Fourier spectra.

▶ **Lemma 33.** *Let $\mathcal{F}\colon [n] \to \mathbb{F}$ be a k-non-signaling function and $\mathcal{Q}\colon \mathbb{F}^n \to \mathbb{C}$ a quasi-distribution. For any set $S \subseteq [n]_{\leq k}$ and event $E \subseteq \mathbb{F}^S$,*

$$\left| \Pr[\mathcal{F}(S) \in E] - \widetilde{\Pr}[\mathcal{Q}(S) \in E] \right| \leq q^n \sum_{\alpha \in \mathbb{F}^S} \left| \widehat{\mathbf{1}_E}(\alpha) \right| \left| \widehat{\mathcal{F}}(\alpha) - \widehat{\mathcal{Q}}(\alpha) \right| \ .$$

In particular, $\Delta_k(\mathcal{Q}, \mathcal{F}) \leq q^{n+k/2} \max_{\alpha \in \mathbb{F}^n_{\leq k}} |\widehat{\mathcal{F}}(\alpha) - \widehat{\mathcal{Q}}(\alpha)|$.

▶ **Corollary 34.** *Let $\mathcal{F}\colon [n] \to \mathbb{F}$ be a k-non-signaling function and $\mathcal{Q}\colon \mathbb{F}^n \to \mathbb{C}$ a quasi-distribution. Then $\Delta_k(\mathcal{Q}, \mathcal{F}) = 0$ if and only if $\widehat{\mathcal{F}}(\alpha) = \widehat{\mathcal{Q}}(\alpha)$ for every $\alpha \in \mathbb{F}^n_{\leq k}$.*

Proof of Lemma 33. The first equation follows immediately from Corollary 31 and Corollary 32. For the second part of the lemma,

$$\begin{aligned}
\Delta_k(\mathcal{Q}, \mathcal{F}) &\leq \max_{S \subseteq [n]_{\leq k}} \max_{E \subseteq \mathbb{F}^S} q^n \sum_{\alpha \in \mathbb{F}^S} \left| \overline{\widehat{\mathbf{1}_E}(\alpha)} \right| \left| \widehat{\mathcal{F}}(\alpha) - \widehat{\mathcal{Q}}(\alpha) \right| \\
&\leq q^n \max_{S \subseteq [n]_{\leq k}} \left(\left(\max_{E \subseteq \mathbb{F}^S} \sum_{\alpha \in \mathbb{F}^S} \left| \widehat{\mathbf{1}_E}(\alpha) \right| \right) \max_{\alpha \in \mathbb{F}^S} \left| \widehat{\mathcal{F}}(\alpha) - \widehat{\mathcal{Q}}(\alpha) \right| \right) \\
&\leq q^n \left(\max_{S \subseteq [n]_{\leq k}} q^{|S|/2} \right) \max_{\alpha \in \mathbb{F}^n_{\leq k}} \left| \widehat{\mathcal{F}}(\alpha) - \widehat{\mathcal{Q}}(\alpha) \right| \\
&\leq q^{n+k/2} \max_{\alpha \in \mathbb{F}^n_{\leq k}} \left| \widehat{\mathcal{F}}(\alpha) - \widehat{\mathcal{Q}}(\alpha) \right| \ .
\end{aligned}$$ ◀

Proof of Corollary 34. If $\widehat{\mathcal{F}}(\alpha) = \widehat{\mathcal{Q}}(\alpha)$ for every $\alpha \in \mathbb{F}^n_{\leq k}$, then by Lemma 33 it follows that $\Delta_k(\mathcal{Q}, \mathcal{F}) = 0$. Conversely, if $\Delta_k(\mathcal{Q}, \mathcal{F}) = 0$, then for every $\alpha \in \mathbb{F}^n_{\leq k}$ and $j \in \mathbb{F}_p$ it holds that $\widetilde{\Pr}[\mathrm{Tr}(\langle \alpha, \mathcal{Q} \rangle) = j] = \Pr[\mathrm{Tr}(\langle \alpha, \mathcal{F} \rangle) = j]$, as these are both events. This implies that $q^n \widehat{\mathcal{Q}}(\alpha) = \sum_{j \in \mathbb{F}_p} \omega^{-j} \widetilde{\Pr}[\mathrm{Tr}(\langle \alpha, \mathcal{Q} \rangle) = j] = \sum_{j \in \mathbb{F}_p} \omega^{-j} \Pr[\mathrm{Tr}(\langle \alpha, \mathcal{F} \rangle) = j] = q^n \widehat{\mathcal{F}}(\alpha)$. ◀

Suppose that we are given a collection of local distributions $(\mathcal{F}_S)_{S \subseteq [n]_{\leq k}}$, namely, $\mathcal{F}_S$ is a distribution over functions $f\colon S \to \mathbb{F}$. We can think of each local distribution $\mathcal{F}_S$ as a function $\mathcal{F}_S\colon \mathbb{F}^S \to \mathbb{C}$, and in this way define for each local distribution $\mathcal{F}_S$ the Fourier coefficients $\widehat{\mathcal{F}_S}(\alpha)$ for each $\alpha \in \mathbb{F}^n_{\subseteq S}$. In the following lemma, we characterize when $(\mathcal{F}_S)_{S \subseteq [n]_{\leq k}}$ is k-non-signaling in terms of the Fourier spectra of the local distributions.

▶ **Lemma 35.** *Let $(\mathcal{F}_S)_{S\subseteq[n]_{\leq k}}$ be a collection of local distributions. Then $(\mathcal{F}_S)_{S\subseteq[n]_{\leq k}}$ is a k-non-signaling function if and only if $q^{|S|}\widehat{\mathcal{F}_S}(\alpha) = q^{|R|}\widehat{\mathcal{F}_R}(\alpha)$ for every $S \subseteq [n]_{\leq k}$, $R \subseteq S$, and $\alpha \in \mathbb{F}^n_{\subseteq R}$.*

Proof. Suppose $(\mathcal{F}_S)_{S\subseteq[n]_{\leq k}}$ is a k-non-signaling function. Fix $S \subseteq [n]_{\leq k}$, $R \subseteq S$, and $\alpha \in \mathbb{F}^n_{\subseteq R}$. Since the collection of local distributions is k-non-signaling we have that $\mathcal{F}_S|_R = \mathcal{F}_R$. Therefore by Lemma 30 we have that $q^{|S|}\widehat{\mathcal{F}_S}(\alpha) = q^{|R|}\widehat{\mathcal{F}_R}(\alpha)$.

Now, fix $S \subseteq [n]_{\leq k}$ and $R \subseteq S$. Applying Corollary 32 to the distributions $\mathcal{F}_S$ and $\mathcal{F}_R$, we see that if $q^{|S|}\widehat{\mathcal{F}_S}(\alpha) = q^{|R|}\widehat{\mathcal{F}_R}(\alpha)$ for every $\alpha \in \mathbb{F}^n_{\subseteq R}$, then $\mathcal{F}_S|_R \equiv \mathcal{F}_R$. Hence, $(\mathcal{F}_S)_{S\subseteq[n]_{\leq k}}$ is k-non-signaling. ◀

4.3 Equivalence between non-signaling functions and quasi-distributions

We show that k-non-signaling functions and k-local quasi-distributions are equivalent. Every k-local quasi-distribution $\mathcal{Q}$ induces a k-non-signaling function $\mathcal{F}$ (Proposition 36). Conversely, every k-non-signaling function $\mathcal{F}$ can be described by a k-local quasi-distribution $\mathcal{Q}$ (Proposition 37). In fact, the set of such quasi-distributions is an affine subspace of co-dimension $\sum_{i=0}^{k}\binom{n}{i}\cdot(q-1)^i$ in $\mathbb{C}^{q^n}$. The first direction of the equivalence is elementary; the other direction is the interesting one.

The aforementioned result is a special case of a result of Abramsky and Brandenburger [1] that establishes an equivalence between *non-signaling empirical models* (a general notion of non-signaling experiments in the language of sheaf theory) and quasi-distributions over *global sections*. Our result strengthens this equivalence by giving an explicit characterization of the affine subspace of quasi-distributions describing a non-signaling function, by leveraging Fourier-analytic tools. This also extends to any finite field $\mathbb{F}$ the equivalence lemma for $\mathbb{F}_2$ presented in [4].[6]

▶ **Proposition 36.** *For every k-local quasi-distribution $\mathcal{Q}$ over functions $f\colon [n] \to \mathbb{F}$ there exists a k-non-signaling function $\mathcal{F}\colon [n] \to \mathbb{F}$ such that $\Delta_k(\mathcal{Q}, \mathcal{F}) = 0$.*

Proof. For every subset $S \subseteq [n]_{\leq k}$, define $\mathcal{F}_S$ to be the distribution over functions $f\colon S \to \mathbb{F}$ where $\Pr[\mathcal{F}_S \text{ outputs } f] := \widetilde{\Pr}[\mathcal{Q}(S) = f(S)]$, namely, such that $\mathcal{F}_S \equiv \mathcal{Q}|_S$. Note that $\mathcal{F}_S$ is a distribution because $\mathcal{Q}$ is k-local, so the relevant probabilities are in $[0,1]$ and sum to 1. The definition immediately implies that $\Pr[\mathcal{F}(S) = g] = \widetilde{\Pr}[\mathcal{Q}(S) = g]$ for every string $g \in \mathbb{F}^S$, and so $\Delta_k(\mathcal{Q}, \mathcal{F}) = 0$. We are left to argue that $\mathcal{F} = \{\mathcal{F}_S\}_{S\subseteq[n]_{\leq k}}$ is k-non-signaling. Let $S \subseteq [n]_{\leq k}$, and let $R \subseteq S$. By definition of $\mathcal{F}$ and Lemma 30 we have that for every $\alpha \in \mathbb{F}^R$, $q^{|S|}\widehat{\mathcal{F}_S}(\alpha) = q^{|S|}\widehat{\mathcal{Q}|_S}(\alpha) = q^{|R|}\widehat{\mathcal{Q}|_R}(\alpha) = q^{|R|}\widehat{\mathcal{F}_R}(\alpha)$. By Lemma 35, it follows that $\mathcal{F}$ is k-non-signaling. ◀

▶ **Proposition 37.** *For every k-non-signaling function $\mathcal{F}\colon [n] \to \mathbb{F}$, there exists a k-local quasi-distribution $\mathcal{Q}$ over functions $f\colon [n] \to \mathbb{F}$ such that $\Delta_k(\mathcal{F}, \mathcal{Q}) = 0$. Moreover, the set of such $\mathcal{Q}$'s (viewed as vectors in $\mathbb{C}^{q^n}$) is the affine subspace of co-dimension $\sum_{i=0}^{k}\binom{n}{i}\cdot(q-1)^i$ in $\mathbb{C}^{q^n}$ given by $\mathcal{Q}_0 + \operatorname{span}\{\chi_\alpha : \alpha \in \mathbb{F}^n, \operatorname{wt}(\alpha) > k\}$, where $\mathcal{Q}_0$ is any solution.*

[6] The characterization further extends to functions taking values in any finite alphabet Σ (not necessarily a field) by adding an abelian group structure to Σ (for example, by identifying Σ with $\mathbb{Z}/|\Sigma|\mathbb{Z}$), and then using analogous tools from Fourier analysis over finite abelian groups.

Proof. Let $\mathcal{Q}$ be a quasi-distribution over functions $f\colon [n] \to \mathbb{F}$. By Corollary 34, it holds that $\Delta_k(\mathcal{Q}, \mathcal{F}) = 0$ if and only if $\widehat{\mathcal{F}}(\alpha) = \widehat{\mathcal{Q}}(\alpha)$ for all $\alpha \in \mathbb{F}^n_{\leq k}$.

Let $\mathcal{Q}_0$ be the quasi-distribution with Fourier coefficients $\widehat{\mathcal{Q}}(\alpha) := \widehat{\mathcal{F}}(\alpha)$ for all α of weight at most k and $\widehat{\mathcal{Q}}(\alpha) := 0$ otherwise. Consider the affine subspace $\mathcal{Q}_0 + \mathrm{span}\{\chi_\alpha : \alpha \in \mathbb{F}^n, \mathrm{wt}(\alpha) > k\}$. By Corollary 34, every quasi-distribution $\mathcal{Q}$ in the affine subspace satisfies $\Delta_k(\mathcal{Q}, \mathcal{F}) = 0$. We note that this affine subspace has dimension $\sum_{i=0}^{k} \binom{n}{i} \cdot (q-1)^i$.

Conversely, suppose that $\mathcal{Q}$ satisfies $\Delta_k(\mathcal{Q}, \mathcal{F}) = 0$. Then by Corollary 34 it holds that $\widehat{\mathcal{Q}}(\alpha) = \widehat{\mathcal{F}}(\alpha)$ for all $\alpha \in \mathbb{F}^n_{\leq k}$, which implies that $\mathcal{Q}$ is in the aforementioned affine subspace. Hence, the affine subspace contains all $\mathcal{Q}$ such that $\Delta_k(\mathcal{Q}, \mathcal{F}) = 0$. ◀

5 Non-signaling linear codes

We wish to define what it means for a non-signaling function $\mathcal{F}\colon [n] \to \mathbb{F}$ to be "in" a linear code $\mathbf{C} \subseteq \mathbb{F}^n$. We introduce two natural definitions for the above goal. The first definition is motivated by the equivalence between non-signaling functions and quasi-distributions established in Section 4.3. The second definition is motivated by a notion of local consistency.

For each of the two definitions, we *characterize* the Fourier spectrum of non-signaling strategies that satisfy the definition, in the exact and in the robust case. Also, we prove a strong relationship between the two definitions, showing that they are *equivalent* (up to a small loss in parameters). The compelling structure that we uncover supports our choice of definitions.

For this section, we remind the reader that a linear code $\mathbf{C}$ over $\mathbb{F}$ with block length n is a linear subspace of $\mathbb{F}^n$. We equivalently also view $\mathbf{C}$ as a linear subspace of the set of all functions $f\colon [n] \to \mathbb{F}$. The dual code of $\mathbf{C}$ is the linear subspace $\mathbf{C}^\perp := \{\alpha : \langle \alpha, f \rangle = 0 \ \forall f \in \mathbf{C}\} \subseteq \mathbb{F}^n$.

5.1 Quasi-distributions supported on linear codes

The equivalence between non-signaling functions and quasi-distributions in Section 4.3 suggests a natural way to capture when a non-signaling function is "in" a given linear code.

▶ **Definition 38.** *Given a k-non-signaling strategy $\mathcal{F}\colon [n] \to \mathbb{F}$, code $\mathbf{C} \subseteq \mathbb{F}^n$ and parameter $k' \leq k$, we say that $\mathcal{F}$ is* **$(\mathbf{C}, k')$-supported** *if there exists a k'-local quasi-distribution $\mathcal{Q}\colon \mathbb{F}^n \to \mathbb{C}$ supported on $\mathbf{C}$ such that $\Delta_{k'}(\mathcal{Q}, \mathcal{F}) = 0$.*

In light of the characterization of the Fourier spectra of quasi-distributions equivalent to a given non-signaling function in Section 4.3, it is natural to ask if the Fourier spectrum of a quasi-distribution supported on $\mathbf{C}$ has a special structure. In the following lemma, we characterize the Fourier spectrum of quasi-distributions supported on a given linear code $\mathbf{C}$. Informally, we show that the condition "Fourier coefficients are constants on cosets of $\mathbf{C}^\perp$" is necessary and sufficient.

▶ **Lemma 39.** *Let $\mathbf{C} \subseteq \mathbb{F}^n$ be a linear code. A quasi-distribution $\mathcal{Q}\colon \mathbb{F}^n \to \mathbb{C}$ is supported on $\mathbf{C}$ if and only if $\widehat{\mathcal{Q}}(\alpha) = \widehat{\mathcal{Q}}(\alpha')$ for all $\alpha, \alpha' \in \mathbb{F}^n$ such that $\alpha - \alpha' \in \mathbf{C}^\perp$.*

The foregoing statement immediately gives us a corollary about non-signaling functions.

▶ **Corollary 40.** *A k-non-signaling strategy $\mathcal{F}\colon [n] \to \mathbb{F}$ is $(\mathbf{C}, k')$-supported if and only if for all $\alpha, \alpha' \in \mathbb{F}^n_{\leq k}$ such that $\alpha - \alpha' \in \mathbf{C}^\perp$ it holds that $\widehat{\mathcal{F}}(\alpha) = \widehat{\mathcal{F}}(\alpha')$.*

Next, we wish to study the Fourier spectrum of a quasi-distribution $\mathcal{Q}$ that is merely *close* to being supported on $\mathbf{C}$. For this case, we give the following "robust" version of Lemma 39.

▶ **Lemma 41.** *Let $\mathbf{C} \subseteq \mathbb{F}^n$ be a linear code, and let $\mathcal{Q}$ be a quasi-distribution.*

- *Suppose that there exists a quasi-distribution $\mathcal{Q}'$ supported on $\mathbf{C}$ such that $\Delta_k(\mathcal{Q}, \mathcal{Q}') \leq \delta$. Then for all $\alpha, \alpha' \in \mathbb{F}^n_{\leq k}$ and $\alpha - \alpha' \in \mathbf{C}^\perp$ it holds that $\left|\widehat{\mathcal{Q}}(\alpha) - \widehat{\mathcal{Q}}(\alpha')\right| \leq \frac{2\delta}{q^n}$.*
- *Conversely, suppose that for all $\alpha, \alpha' \in \mathbb{F}^n_{\leq k}$ and $\alpha - \alpha' \in \mathbf{C}^\perp$ it holds that $\left|\widehat{\mathcal{Q}}(\alpha) - \widehat{\mathcal{Q}}(\alpha')\right| \leq \frac{2\delta}{q^n}$. Then there exists a quasi-distribution $\mathcal{Q}'$ supported on $\mathbf{C}$ such that $\Delta_k(\mathcal{Q}, \mathcal{Q}') \leq q^{k/2} \cdot 2\delta$.*

We note that in Lemma 41, neither quasi-distribution is required to be local.

5.1.1 Proof of Lemma 39

Define the affine spaces

$$V_1 = \left\{ \mathcal{Q} \colon \mathbb{F}^n \to \mathbb{C} \text{ s.t. } \sum_{f \in \mathbf{C}} \mathcal{Q}(f) = 1 \text{ and } \mathcal{Q}(f) = 0 \ \forall f \notin \mathbf{C} \right\} ,$$

$$V_2 = \left\{ \mathcal{Q} \colon \mathbb{F}^n \to \mathbb{C} \text{ s.t. } \widehat{\mathcal{Q}}(0^n) = \frac{1}{q^n} \text{ and } \widehat{\mathcal{Q}}(\alpha) = \widehat{\mathcal{Q}}(\alpha + \gamma) \ \forall \alpha \in \mathbb{F}^n, \gamma \in \mathbf{C}^\perp \right\} .$$

It suffices to prove that $V_1 = V_2$. First we show that $\dim(V_1) = \dim(V_2)$. The dimension of V_1 is $|\mathbf{C}| - 1$ because the $|\mathbf{C}|$ free terms are subject to a single linear constraint. The dimension of V_2 is $q^n / \left|\mathbf{C}^\perp\right| - 1$ because the Fourier coefficients are constant on each coset of $\mathbf{C}^\perp$, and on each coset they can take on an arbitrary value; the one exception is the coset $\mathbf{C}^\perp$, on which the Fourier coefficients must be $\frac{1}{q^n}$. Recalling that $q^n = |\mathbf{C}| \cdot |\mathbf{C}^\perp|$, we deduce that $\dim(V_1) = \dim(V_2)$.

Next we show that $V_1 \subseteq V_2$. Fix $\mathcal{Q} \in V_1$. Since $\sum_f \mathcal{Q}(f) = 1$, we have $\widehat{\mathcal{Q}}(0^n) = \frac{1}{q^n} \sum_f \mathcal{Q}(f) \cdot \omega^0 = \frac{1}{q^n}$. Moreover, for any $\alpha \in \mathbb{F}^n$ and $\gamma \in \mathbf{C}^\perp$,

$$\widehat{\mathcal{Q}}(\alpha + \gamma) = \frac{1}{q^n} \cdot \sum_f \mathcal{Q}(f) \cdot \omega^{-\mathrm{Tr}(\langle \alpha + \gamma f \rangle)} = \frac{1}{q^n} \cdot \sum_f \mathcal{Q}(f) \cdot \omega^{-\mathrm{Tr}(\langle \alpha, f \rangle)} \cdot \omega^{-\mathrm{Tr}(\langle \gamma, f \rangle)} .$$

Since $\mathcal{Q} \in V_1$, if $\mathcal{Q}(f) \neq 0$ then $f \in \mathbf{C}$ and hence $\omega^{\mathrm{Tr}(\langle \gamma, f \rangle)} = \omega^{\mathrm{Tr}(0)} = 1$. Therefore,

$$\widehat{\mathcal{Q}}(\alpha + \gamma) = \frac{1}{q^n} \cdot \sum_f \mathcal{Q}(f) \cdot \omega^{-\mathrm{Tr}(\langle \alpha, f \rangle)} = \widehat{\mathcal{Q}}(\alpha) .$$

Thus $V_1 \subseteq V_2$. Since $\dim(V_1) = \dim(V_2)$ and $V_1 \subseteq V_2$, we conclude that $V_1 = V_2$.

5.1.2 Proof of Lemma 41

Suppose $\mathcal{Q} \colon \mathbb{F}^n \to \mathbb{C}$ is a quasi-distribution such that there exists a quasi-distribution $\mathcal{Q}'$ supported on $\mathbf{C}$ with $\Delta_k(\mathcal{Q}, \mathcal{Q}') \leq \delta$. Fix $\alpha \in \mathbb{F}^n_{\leq k}$, so that $S = \mathrm{supp}(\alpha)$ has $|S| \leq k$. Since $\Delta_k(\mathcal{Q}, \mathcal{Q}') \leq \delta$, we have that $\sum_{g \in \mathbb{F}^S} \left|\widetilde{\Pr}[\mathcal{Q}(S) = g] - \widetilde{\Pr}[\mathcal{Q}'(S) = g]\right| \leq \delta$. Therefore,

$$\begin{aligned}
\left|\widehat{\mathcal{Q}}(\alpha) - \widehat{\mathcal{Q}}'(\alpha)\right| &\leq \frac{1}{q^n} \sum_{j \in \mathbb{F}_p} \left|\omega^{-j}\right| \left|\widetilde{\Pr}[\mathrm{Tr}(\langle \alpha, \mathcal{Q} \rangle) = j] - \widetilde{\Pr}[\mathrm{Tr}(\langle \alpha, \mathcal{Q}' \rangle) = j]\right| \\
&= \frac{1}{q^n} \sum_{j \in \mathbb{F}_p} \sum_{g \in \mathbb{F}^S : \mathrm{Tr}(\langle \alpha, g \rangle) = j} \left|\widetilde{\Pr}[\mathcal{Q}(S) = g] - \widetilde{\Pr}[\mathcal{Q}'(S) = g]\right| \\
&\leq \frac{1}{q^n} \sum_{g \in \mathbb{F}^S} \left|\widetilde{\Pr}[\mathcal{Q}(S) = g] - \widetilde{\Pr}[\mathcal{Q}'(S) = g]\right| \leq \frac{\delta}{q^n} .
\end{aligned}$$

By Lemma 39, we know that for every $\alpha, \alpha' \in \mathbb{F}^n$ such that $\alpha - \alpha' \in \mathbf{C}^\perp$ it holds that $\left|\widehat{\mathcal{Q}}'(\alpha) - \widehat{\mathcal{Q}}'(\alpha')\right| = 0$. Hence, for every $\alpha, \alpha' \in \mathbb{F}^n_{\leq k}$ such that $\alpha - \alpha' \in \mathbf{C}^\perp$ it holds that

$$\begin{aligned}\left|\widehat{\mathcal{Q}}(\alpha) - \widehat{\mathcal{Q}}(\alpha')\right| &\leq \left|\widehat{\mathcal{Q}}(\alpha) - \widehat{\mathcal{Q}}'(\alpha)\right| + \left|\widehat{\mathcal{Q}}'(\alpha) - \widehat{\mathcal{Q}}'(\alpha')\right| + \left|\widehat{\mathcal{Q}}'(\alpha') - \widehat{\mathcal{Q}}(\alpha')\right| \\ &\leq \frac{\delta}{q^n} + 0 + \frac{\delta}{q^n} = \frac{2\delta}{q^n} \ .\end{aligned}$$

Now, suppose that $\mathcal{Q}$ is a quasi-distribution such that $\left|\widehat{\mathcal{Q}}(\alpha) - \widehat{\mathcal{Q}}(\alpha')\right| \leq \frac{2\delta}{q^n}$ for all $\alpha, \alpha' \in \mathbb{F}^n_{\leq k}$ such that $\alpha - \alpha' \in \mathbf{C}^\perp$. For each $\alpha \in \mathbb{F}^n$, let γ_α be an element of the coset $\alpha + \mathbf{C}^\perp$ of minimal weight (ties are broken arbitrarily). Define $\mathcal{Q}'$ to be the quasi-distribution where $\widehat{\mathcal{Q}}'(\alpha) := \widehat{\mathcal{Q}}(\gamma_\alpha)$ if $\mathrm{wt}(\gamma_\alpha) \leq k$ and 0 otherwise. By construction, for any $\alpha, \alpha' \in \mathbb{F}^n$ such that $\alpha - \alpha' \in \mathbf{C}^\perp$ it holds that $\widehat{\mathcal{Q}}'(\alpha) = \widehat{\mathcal{Q}}'(\alpha')$, so $\mathcal{Q}'$ is supported on $\mathbf{C}$ by Lemma 39. Let $\alpha \in \mathbb{F}^n_{\leq k}$. By construction, we know that $\left|\widehat{\mathcal{Q}}(\alpha) - \widehat{\mathcal{Q}}'(\alpha)\right| \leq \left|\widehat{\mathcal{Q}}(\alpha) - \widehat{\mathcal{Q}}(\gamma_\alpha)\right| + \left|\widehat{\mathcal{Q}}(\gamma_\alpha) - \widehat{\mathcal{Q}}'(\alpha)\right| \leq \frac{2\delta}{q^n} + 0 = \frac{2\delta}{q^n}$, since $\alpha - \gamma_\alpha \in \mathbf{C}^\perp$ and $\mathrm{wt}(\gamma_\alpha) \leq \mathrm{wt}(\alpha) \leq k$. Therefore, by Lemma 33 we have that $\Delta_k(\mathcal{Q}, \mathcal{Q}') \leq q^{k/2} \cdot 2\delta$.

5.2 Locally-explainable non-signaling functions

We introduce another natural definition that captures when a non-signaling function $\mathcal{F}$ is "in" a given linear code $\mathbf{C} \subseteq \mathbb{F}^n$. This time we take the perspective of local consistency, namely, we shall require that the output of $\mathcal{F}$ is always consistent with a codeword in $\mathbf{C}$.

▶ **Definition 42.** *Given a k-non-signaling strategy $\mathcal{F}\colon [n] \to \mathbb{F}$, code $\mathbf{C} \subseteq \mathbb{F}^n$, and parameter $k' \leq k$, we say that $\mathcal{F}$ is* **$(\mathbf{C}, k')$-explainable** *if for every set $S \subseteq [n]_{\leq k'}$ it holds that $\Pr[\mathcal{F}(S) \in \mathbf{C}|_S] = 1$.*

Note that $\mathcal{F}$ is $(\mathbf{C}, k')$-explainable if and only if $\Pr[\langle \alpha, \mathcal{F} \rangle = 0] = 1$ for every $\alpha \in \mathbf{C}^\perp_{\leq k'}$. The non-trivial direction of the equivalence is implied by the following lemma.

▶ **Lemma 43.** *Let $\mathbf{C} \subseteq \mathbb{F}^n$ be a linear code, $S \subseteq [n]_{\leq k}$, and $g\colon S \to \mathbb{F}$. If $\langle \alpha, g \rangle = 0$ for every $\alpha \in \mathbf{C}^\perp_{\subseteq S}$, then there is a codeword $f \in \mathbf{C}$ such that $f|_S = g$.*

Proof. Since $\mathbf{C} \subseteq \mathbb{F}^n$ is a linear code, there is a *pivotal set* $P \subseteq [n]$ of size $|P| = \dim(\mathbf{C})$ such that for all $y\colon P \to \mathbb{F}$ there is a unique codeword $f \in \mathbf{C}$ satisfying $f|_P = y$. Such P need not be unique.

Let $P^* \subseteq [n]$ be a pivotal set such that $|P^* \cap S|$ is maximal, and let $P_S := P^* \cap S$. Define $f'\colon P^* \to \mathbb{F}$ by letting $f'(i) = g(i)$ for all $i \in P_S$, and letting $f'(j)$ be arbitrary for all $j \in P^* \setminus P_S$. Since P^* is a pivotal set, there exists a unique $f \in \mathbf{C}$ such that $f|_{P^*} = f'$.

It remains to show that $f|_S = g$. Let $i \in S$. If $i \in P_S$, then $f(i) = f'(i) = g(i)$, as required. Suppose that $i \notin P_S$. Since P^* is maximal, there exists $\alpha \in \mathbf{C}^\perp$ such that $\alpha_i = 1$ and $\mathrm{supp}(\alpha) \subseteq P_S \cup \{i\} \subseteq S$. Indeed, if no such α exists then for any codeword $h \in \mathbf{C}$, $h(i)$ is not determined by $\{h(j) : j \in P_S\}$. Hence, the set $P_S \cup \{i\}$ can be extended into a pivotal set for $\mathbf{C}$, which contradicts the maximality of P^*. Therefore, such an α exists. Since $\langle \alpha, f \rangle = 0$ and $\langle \alpha, g \rangle = 0$, we get that $0 = \langle \alpha, f \rangle - \langle \alpha, g \rangle = f(i) + \sum_{j \in P_S} \alpha_j f(j) - g(i) - \sum_{j \in P_S} \alpha_j g(j) = f(i) + \sum_{j \in P_S} \alpha_j g(j) - g(i) - \sum_{j \in P_S} \alpha_j g(j) = f(i) - g(i)$, and therefore $f(i) = g(i)$. We conclude that $f|_S = g$, as required. ◀

We provide a characterization of the Fourier spectrum of $\mathbf{C}$-explainable non-signaling functions, both in the exact and in the robust cases, as captured by the respective lemmas below. Both lemmas make crucial use of Lemma 29.

▶ **Lemma 44.** *Let $\mathcal{F}\colon [n] \to \mathbb{F}$ be a k-non-signaling function. Then $\mathcal{F}$ is $(\mathbf{C}, k')$-explainable if and only if $\widehat{\mathcal{F}}(\alpha) = \frac{1}{q^n}$ for every $\alpha \in \mathbf{C}^{\perp}_{\leq k'}$.*

Proof. We know that $\mathcal{F}$ is $(\mathbf{C}, k')$-explainable if and only if for every $\alpha \in \mathbf{C}^{\perp}_{\leq k'}$ it holds that $\Pr[\langle \alpha, \mathcal{F}\rangle = 0] = 1$. By Lemma 29, we know that $\Pr[\langle \alpha, \mathcal{F}\rangle = 0] = 1$ if and only if $\widehat{\mathcal{F}}(a\alpha) = \frac{1}{q^n}$ for every $a \in \mathbb{F}$, as M is invertible and maps the distribution $\Pr[\langle \alpha, \mathcal{F}\rangle = 0] = 1$ and $\Pr[\langle \alpha, \mathcal{F}\rangle = b] = 0$ for all other b to the vector 1^q. We conclude the proof by noting that if $\alpha \in \mathbf{C}^{\perp}_{\leq k'}$ then $a\alpha \in \mathbf{C}^{\perp}_{\leq k'}$ for any $a \in \mathbb{F}$. ◀

▶ **Lemma 45.** *Let $\mathcal{F}\colon [n] \to \mathbb{F}$ be a k-non-signaling function, and let $\alpha \in \mathbb{F}^n_{\leq k}$.*

- *If $\Pr[\langle \alpha, \mathcal{F}\rangle = 0] \geq 1 - \varepsilon$, then $|\widehat{\mathcal{F}}(a\alpha) - \frac{1}{q^n}| \leq \frac{2\varepsilon}{q^n}$ for every $a \in \mathbb{F}$.*
- *If $|\widehat{\mathcal{F}}(a\alpha) - \frac{1}{q^n}| \leq \frac{\varepsilon}{q^n}$ for every $a \in \mathbb{F}$, then $\Pr[\langle \alpha, \mathcal{F}\rangle = 0] \geq 1 - \varepsilon$.*

Proof. Suppose that $\Pr[\langle \alpha, \mathcal{F}\rangle = 0] \geq 1 - \varepsilon$. This immediately implies that, for every $a \in \mathbb{F}$, $\Pr[\langle a\alpha, \mathcal{F}\rangle = 0] \geq 1 - \varepsilon$. Therefore,

$$\begin{aligned}
\left|q^n \widehat{\mathcal{F}}(a\alpha) - 1\right| &= \left|-1 + \sum_{b \in \mathbb{F}} \omega^{-\mathrm{Tr}(ab)} \Pr[\langle a\alpha, \mathcal{F}\rangle = b]\right| \\
&\leq |-1 + \Pr[\langle a\alpha, \mathcal{F}\rangle = 0]| + \sum_{b \neq 0} |\omega^{-\mathrm{Tr}(ab)}||\Pr[\langle a\alpha, \mathcal{F}\rangle = b]| \\
&\leq \varepsilon + \sum_{b \neq 0} \Pr[\langle a\alpha, \mathcal{F}\rangle = b] \\
&= \varepsilon + (1 - \Pr[\langle a\alpha, \mathcal{F}\rangle = 0]) \leq 2\varepsilon \ .
\end{aligned}$$

This proves the first direction.

For the second direction, let $v \in \mathbb{C}^q$ be the vector where $v_b = \Pr[\langle \alpha, \mathcal{F}\rangle = b]$ and let $w \in \mathbb{C}^q$ be the vector where $w_a = q^n \widehat{\mathcal{F}}(a\alpha)$. Note that $Mv = w$, where M is the matrix from Lemma 29. Suppose that $|\widehat{\mathcal{F}}(a\alpha) - \frac{1}{q^n}| \leq \frac{\varepsilon}{q^n}$ for every $a \in \mathbb{F}$, so that $|w_a - 1| \leq \varepsilon$ for every $a \in \mathbb{F}$. Then, we have that $\|w - 1^q\|^2_{\ell_2} \leq q\varepsilon^2$, so that $\|w - 1^q\|_{\ell_2} \leq \varepsilon\sqrt{q}$. Let $u \in \mathbb{C}^q$ be the vector where $u_0 = 1$ and $u_b = 0$ for all other $b \in \mathbb{F}$. Observe that $Mu = 1^q$. Since $\frac{1}{\sqrt{q}} M$ is unitary, we have that $\|\frac{1}{\sqrt{q}} M(v - u)\|_{\ell_2} = \|\frac{1}{\sqrt{q}}(w - 1^q)\|_{\ell_2} \leq \frac{1}{\sqrt{q}} \cdot \varepsilon\sqrt{q} = \varepsilon$. Therefore, $|v_b - u_b| \leq \varepsilon$ for all $b \in \mathbb{F}$. In particular, $|v_0 - 1| \leq \varepsilon$, so that $\Pr[\langle \alpha, \mathcal{F}\rangle = 0] \geq 1 - \varepsilon$. ◀

5.3 The relationship between the two definitions

We have given two natural definitions of what it means for a non-signaling function to be in a linear code. Which of the two definitions is more "correct"? Lemma 39 and Lemma 44 show that Definition 38 implies Definition 42, in the sense that if $\mathcal{F}$ is $(\mathbf{C}, k')$-supported then $\mathcal{F}$ is $(\mathbf{C}, k')$-explainable. We prove that, conversely, Definition 42 implies Definition 38 up to a factor of 2 in the locality k'. We conclude that the two definitions are essentially equivalent.

▶ **Lemma 46.** *Let $\mathbf{C} \subseteq \mathbb{F}^n$ be a linear code, and let $\mathcal{F}\colon [n] \to \mathbb{F}$ be a k-non-signaling function.*

- *If $\mathcal{F}$ is $(\mathbf{C}, k')$-supported then $\mathcal{F}$ is $(\mathbf{C}, k')$-explainable.*
- *If $\mathcal{F}$ is $(\mathbf{C}, k')$-explainable then $\mathcal{F}$ is $(\mathbf{C}, k'/2)$-supported.*

▶ Remark 47. For specific choices of $\mathbf{C}$ one can achieve stronger versions of the above lemma. For example, when $\mathbf{C}$ is the Hadamard code (all linear functions), one can prove the lemma with $k' - 1$ in place of $k'/2$. Also, *some* gap in locality is necessary: taking again $\mathbf{C}$ to be the Hadamard code, there exists a non-signaling function $\mathcal{F}$ that is $(\mathbf{C}, k)$-explainable and $(\mathbf{C}, k-1)$-supported but *not* $(\mathbf{C}, k)$-supported. (The foregoing statements are shown implicitly in [4].)

Proof. Lemma 39 and Lemma 44 imply the first direction, as any $(\mathbf{C}, k')$-supported k-non-signaling function $\mathcal{F}$ satisfies $\widehat{\mathcal{F}}(\alpha) = \widehat{\mathcal{F}}(0^n) = \frac{1}{q^n}$ for every $\alpha \in \mathbf{C}^{\perp}_{\leq k'}$, implying that $\mathcal{F}$ is $(\mathbf{C}, k')$-explainable.

We now prove the second direction. Fix $\alpha \in \mathbf{C}^{\perp}_{\leq k'}$, and let $S := \{i \in [n] : \alpha_i \neq 0\}$. Note that $|S| \leq k'$ since $|S| = \mathrm{wt}(\alpha)$. We first show that $\Pr[\langle \alpha, \mathcal{F} \rangle = 0] = 1$. Indeed, since $\mathcal{F}$ is $(\mathbf{C}, k')$-explainable, we have that

$$\begin{aligned}\Pr[\langle \alpha, \mathcal{F} \rangle = 0] &\geq \Pr[\langle \alpha, \mathcal{F} \rangle = 0 \wedge \exists\, f \in \mathbf{C} \text{ s.t. } \mathcal{F}(S) = f|_S] \\ &= \Pr[\langle \alpha, f \rangle = 0 \wedge \exists\, f \in \mathbf{C} \text{ s.t. } \mathcal{F}(S) = f|_S] \\ &= \Pr[\exists\, f \in \mathbf{C} \text{ s.t. } \mathcal{F}(S) = f|_S] = 1 \ ,\end{aligned}$$

and so $\Pr[\langle \alpha, \mathcal{F} \rangle = 0] = 1$.

Now, for any $\alpha, \alpha' \in \mathbb{F}^n_{\leq k'/2}$ with $\alpha - \alpha' \in \mathbf{C}^{\perp}$ we get that for any $b \in \mathbb{F}$,

$$\Pr[\langle \alpha, \mathcal{F} \rangle = b] = \Pr[\langle \alpha', \mathcal{F} \rangle + \langle \alpha - \alpha', \mathcal{F} \rangle = b] = \Pr[\langle \alpha', \mathcal{F} \rangle = b] \ ,$$

since $\Pr[\langle \alpha - \alpha', \mathcal{F} \rangle = 0] = 1$ as $\alpha - \alpha' \in \mathbf{C}^{\perp}$ with $\mathrm{wt}(\alpha - \alpha') \leq k'$. This shows that the vectors $(\Pr[\langle \alpha, \mathcal{F} \rangle = b])_{b \in \mathbb{F}}$ and $(\Pr[\langle \alpha', \mathcal{F} \rangle = b])_{b \in \mathbb{F}}$ are the same. Thus, $\widehat{\mathcal{F}}(\alpha) = \widehat{\mathcal{F}}(\alpha')$, by the definition of $\mathcal{F}$'s Fourier coefficients. By Lemma 39, it follows that $\mathcal{F}$ is $(\mathbf{C}, k'/2)$-supported. ◀

6 Low-degree testing

In this section, we prove Theorem 2. Throughout this section, we let $m, d, k \in \mathbb{N}$ be parameters, and p be a prime where $p \geq d+2$. Let $\mathsf{RS}^{\otimes m}[\mathbb{F}_p, d]$ denote the code of polynomials from $\mathbb{F}_p^m$ to $\mathbb{F}_p$ of individual degree at most d in each variable. We let $\mathcal{F} \colon \mathbb{F}_p^m \to \mathbb{F}_p$ be a k-non-signaling function. For a subset $S \subseteq \mathbb{F}_p$ and $a \in S$, we let $\delta_{a,S}(x) = \frac{\prod_{a' \in S \setminus \{a\}} (x - a')}{\prod_{a' \in S \setminus \{a\}} (a - a')}$ be the degree $|S| - 1$ polynomial satisfying $\delta_a(x) = \begin{cases} 1 & \text{if } x = a \ , \\ 0 & \text{if } x \in S \setminus \{a\} \ . \end{cases}$

We begin by recalling the degree-d evenly-spaced points test.

▶ **Definition 48** (Evenly-spaced points test)**.** *Given a function $f \colon \mathbb{F}_p^m \to \mathbb{F}_p$, the degree-$d$ evenly-spaced points test:*

1. *samples a random point $x \in \mathbb{F}_p^m$ and slope $h \in \mathbb{F}_p^m \setminus \{0^m\}$,*
2. *checks that $\sum_{i=0}^{d+1} c_i f(x + ih) = 0$, where $c_i = (-1)^i \binom{d+1}{i}$.*

▶ **Definition 49** (Evenly-spaced self-correction)**.** *The evenly-spaced self-correction of $\mathcal{F}$, denoted $\hat{\mathcal{F}}$, is a $\lfloor k/(d+1) \rfloor$-non-signaling function defined as follows. Let $w_0, w_1, \ldots, w_m \in \mathbb{F}_p^m$ be independent uniformly random vectors in $\mathbb{F}_p^m$, and for each $x \in \mathbb{F}_p^m$ let $w_x = w_0 + \sum_{i=1}^m x_i w_i$. For an input set S and $g \colon S \to \mathbb{F}_p$, the distribution of $\hat{\mathcal{F}}(S)$ correction is defined as*

$$\Pr[\hat{\mathcal{F}}(S) = g] = \Pr_{w_0, \ldots, w_m, \mathcal{F}}[-\sum_{j=1}^{d+1} c_j \mathcal{F}(x + j w_x) = g(x) \ \forall x \in S] \ .$$

Our main theorem is the following.

▶ **Theorem 50** (Formal version of Theorem 2)**.** *Let p be a prime, and let $m, d \in \mathbb{N}$ with $p \geq d+2$. Let $\mathcal{F} \colon \mathbb{F}_p^m \to \mathbb{F}_p$ be a k-non-signaling function. Suppose that $\mathcal{F}$ passes the degree-d evenly-spaced points test with probability $1 - \varepsilon$. Then there exists a k'-non-signaling function $\mathcal{G} \colon \mathbb{F}_p^m \to \mathbb{F}_p$ that is $(\mathsf{RS}^{\otimes m}[\mathbb{F}_p, d], k')$-supported such that $\Delta_{k'}(\hat{\mathcal{F}}, \mathcal{G}) \leq O(p^{3k'/2}(d+1)^m \varepsilon)$, where $k' = \lfloor \frac{k}{2(d+1)(d+2)} \rfloor - 3$.*

When $\varepsilon = 0$, the theorem can be simplified considerably to the statement below.

▶ **Theorem 51** (Formal version of Theorem 2, zero error case)**.** *Let p be a prime, and let $m, d \in \mathbb{N}$ with $p \geq d+2$. Let $\mathcal{F}\colon \mathbb{F}_p^m \to \mathbb{F}_p$ be a k-non-signaling function. Suppose that $\mathcal{F}$ passes the degree-d evenly-spaced points test with probability 1. Then $\mathcal{F}$ is $(\mathsf{RS}^{\otimes m}[\mathbb{F}_p, d], \lfloor k/(d+2) \rfloor - 1)$-supported.*

We prove Theorem 50 via the following statements, each proved in one of the following sections.

▶ **Lemma 52** (Average to worst case)**.** *Suppose that $\mathcal{F}$ passes the degree-d evenly-spaced points test with probability $1-\varepsilon$, i.e. that $\Pr_{h \neq 0^m, x, \mathcal{F}}[\sum_{i=0}^{d+1} c_i \mathcal{F}(x+ih) = 0] \geq 1-\varepsilon$. Then for every $x, h \in \mathbb{F}_p^m$ with $h \neq 0^m$ it holds that $\Pr_{\hat{\mathcal{F}}}[\sum_{i=0}^{d+1} c_i \hat{\mathcal{F}}(x+ih) = 0] \geq 1-(d+1)\varepsilon$.*

▶ **Lemma 53** (Evenly-spaced points to axis-parallel lines)**.** *Let $\hat{\mathcal{F}}\colon \mathbb{F}_p^m \to \mathbb{F}_p$ be a $\hat{k}$-non-signaling function, with $\hat{k} \geq 2d+3$. Suppose that for every $x, h \in \mathbb{F}_p^m$ with $h \neq 0^m$ it holds that $\Pr_{\hat{\mathcal{F}}}[\sum_{i=0}^{d+1} c_i \hat{\mathcal{F}}(x+ih) = 0] \geq 1-\varepsilon$. Then for every $b_1, \ldots, b_m \in \mathbb{F}_p$, $i \in [m]$, and $S \subseteq \mathbb{F}_p \setminus \{b_i\}$ of size $|S| = d+1 \leq \hat{k}-d-2$ it holds that*

$$\Pr_{\hat{\mathcal{F}}}[\hat{\mathcal{F}}(b_1, \ldots, b_m) = \sum_{a \in S} \delta_{a,S}(b_i) \cdot \hat{\mathcal{F}}(b_1, \ldots, b_{i-1}, a, b_{i+1}, \ldots, b_m)] \geq 1 - p\varepsilon \ .$$

▶ **Lemma 54** (Robust local characterization)**.** *Let $\hat{\mathcal{F}}\colon \mathbb{F}_p^m \to \mathbb{F}_p$ be a $\hat{k}$-non-signaling function, and suppose that for every $i \in [m]$, $b_1, \ldots, b_m \in \mathbb{F}_p$, and $S' \subseteq \mathbb{F}_p \setminus \{b_i\}$ of size $|S'| = d+1$ it holds that*

$$\Pr_{\hat{\mathcal{F}}}[\hat{\mathcal{F}}(b_1, \ldots, b_m) = \sum_{a \in S'} \delta_{a,S'}(b_i) \cdot \hat{\mathcal{F}}(b_1, \ldots, b_{i-1}, a, b_{i+1}, \ldots, b_m)] \geq 1 - \varepsilon \ .$$

Then for every $S \subseteq \mathbb{F}_p^m$ with $|S| \leq \lfloor \hat{k}/(d+2) \rfloor$ it holds that $\Pr[\hat{\mathcal{F}}(S) \in \mathsf{RS}^{\otimes m}[\mathbb{F}_p, d]\big|_S] \geq 1 - 2|S| \cdot (d+1)^{m-1}\varepsilon$.

▶ **Lemma 55.** *Let $\hat{\mathcal{F}}\colon \mathbb{F}_p^m \to \mathbb{F}_p$ be a $\hat{k}$-non-signaling function, and let $k' \leq \hat{k}$. Suppose that for every $S \subseteq \mathbb{F}_p^m$ with $|S| \leq k'$ it holds that $\Pr[\hat{\mathcal{F}}(S) \in \mathsf{RS}^{\otimes m}[\mathbb{F}_p, d]\big|_S] \geq 1-\varepsilon$. Then there exists a $k'/2$-non-signaling function $\mathcal{G}$ that is $(\mathsf{RS}^{\otimes m}[\mathbb{F}_p, d], k'/2)$-supported and $\Delta_{k'/2}(\mathcal{F}, \mathcal{G}) \leq (p^{k'/2}+1) \cdot p^{k'/4} \cdot 2\varepsilon$.*

6.1 Step 1: Average to worst case reduction

We prove Lemma 52. Fix $x, h \in \mathbb{F}_p^m$ with $h \neq 0^m$. Observe that $w_{x+ih} = w_x + i(w_h - w_0)$. Therefore

$$\begin{aligned}
&\Pr_{\hat{\mathcal{F}}}[\sum_{i=0}^{d+1} c_i \hat{\mathcal{F}}(x+ih) = 0] = \Pr_{w_0, \ldots, w_m, \mathcal{F}}[\sum_{i=0}^{d+1} c_i \sum_{j=1}^{d+1} -c_j \mathcal{F}(x+ih+jw_{x+ih}) = 0] \\
&\Pr_{w_0, \ldots, w_m, \mathcal{F}}[\sum_{j=1}^{d+1} -c_j \sum_{i=0}^{d+1} c_i \mathcal{F}(x+ih+jw_x+ij(w_h-w_0)) = 0] \\
&\geq \Pr_{w_0, \ldots, w_m, \mathcal{F}}[\sum_{i=0}^{d+1} c_i \mathcal{F}(x+jw_x+i(h+j(w_h-w_0))) = 0 \ \forall j \in [d+1]] \\
&\geq 1-(d+1)\varepsilon \ ,
\end{aligned}$$

where last inequality is by union bound, using the fact that for $j \neq 0$ the vectors $x + jw_x$ and $h + j(w_h - w_0)$ are independent and uniformly random vectors in $\mathbb{F}_p^m$. Indeed, for $h \neq 0^m$ the vector $h + j(w_h - w_0)$ is equal to $h + \sum_{i=1}^m jh_i w_i$, and hence is uniformly random, and $x + jw_x$ is equal to $x + jw_0 + \sum_{i=1}^m jx_i w_i$, which is also uniformly random and independent of $h + j(w_h - w_0)$ because the term w_0 is independent of all other w_i's.

6.2 Step 2: From evenly-spaced points to axis-parallel lines

We prove Lemma 53. Note that by the assumption for every $x \in \mathbb{F}_p^m$ and $h = e_i$ it holds that $\Pr_{\hat{\mathcal{F}}}[\sum_{\ell=0}^{d+1} c_\ell \hat{\mathcal{F}}(x + \ell e_i) = 0] \geq 1 - \varepsilon$. Fix $i \in [m]$, $b_1, \ldots, b_{i-1}, b_{i+1}, \ldots b_m \in \mathbb{F}_p$, and $S \subseteq \mathbb{F}_p$ of size $|S| = d + 2$. We claim that

$$\Pr[\exists\, g \in \mathsf{RS}[\mathbb{F}_p, d] \text{ s.t. } \hat{\mathcal{F}}(b_1, \ldots, b_{i-1}, s, b_{i+1}, \ldots b_m) \equiv g(s)\ \forall s \in S] \geq 1 - p\varepsilon\ . \tag{1}$$

This clearly implies Lemma 53.

In order to prove Equation (1), let us order the elements of S as $s_1 \leq s_2 \leq \cdots \leq s_{d+2}$ by treating s_i's as integers in $\{0, 1, 2, \ldots, p-1\} \subseteq \mathbb{N}$. For each $j \in \mathbb{F}_p$ define the interval $I_j = \{j, j+1, j+2, \ldots, j+d+1\}$, and denote $Q_j = \{s \in S : s \leq j + d + 2\} \cup I_j$. We claim that with high probability $\hat{\mathcal{F}}$ on the points corresponding to Q_j agrees with some univariate polynomial. Specifically, we prove the following claim.

▷ Claim 56. For each $j = 0, 1 \ldots, p-1$ it holds that

$$\Pr[\exists\, g \in \mathsf{RS}[\mathbb{F}_p, d] \text{ s.t. } \hat{\mathcal{F}}(b_1, \ldots, b_{i-1}, \ell, b_{i+1}, \ldots b_m) \equiv g(\ell)\ \forall \ell \in Q_j] \geq 1 - (j+1)\varepsilon\ .$$

It is clear that the statement of Claim 56 for $j = p - 1$ implies Equation (1), and hence proves Lemma 53.

Proof of Claim 56. The proof is by induction in j. For the base case of $j = 0$ we query $\hat{\mathcal{F}}$ on the set I_0. By the assumption of the lemma we have

$$\Pr_{\hat{\mathcal{F}}}[\exists\, g \in \mathsf{RS}[\mathbb{F}_p, d] \text{ s.t. } \hat{\mathcal{F}}(b_1, \ldots, b_{i-1}, \ell, b_{i+1}, \ldots b_m) = \ell\ \forall \ell \in I_0] \geq 1 - \varepsilon\ ,$$

and the claim hold since for $j = 0$ we have $Q_0 = I_0$.

For the induction step suppose that the statement of the claim holds for $j - 1$, i.e., with probability $1 - j\varepsilon$ there exists a univariate polynomial g of degree d that agrees with $\hat{\mathcal{F}}$ on Q_{j-1}. Note that the set $Q'_{j-1} = Q_{j-1} \setminus \{j-1\}$ contains the $d+1$ consecutive points $\{j, j+1, \ldots, j+d\}$, and these points uniquely define the polynomial g of degree d. Therefore

$$\Pr[\exists\, g \in \mathsf{RS}[\mathbb{F}_p, d] \text{ s.t. } \hat{\mathcal{F}}(b_1, \ldots, b_{i-1}, \ell, b_{i+1}, \ldots b_m) \equiv g(\ell)\ \forall \ell \in Q'_{j-1}] \geq 1 - j\varepsilon\ .$$

On the other hand, by the assumption that $\hat{\mathcal{F}}$ satisfies evenly-spaced constraints with probability $\geq 1 - \varepsilon$ we have

$$\Pr[\exists\, g' \in \mathsf{RS}[\mathbb{F}_p, d] \text{ s.t. } \hat{\mathcal{F}}(b_1, \ldots, b_{i-1}, \ell, b_{i+1}, \ldots b_m) \equiv g'(\ell)\ \forall \ell \in I_j] \geq 1 - \varepsilon\ .$$

Note that the above statement requires that $p \geq d + 2$, as passing the evenly-spaced points test along a line corresponds to being a univariate polynomial only when $p \geq d + 2$.

Query $\hat{\mathcal{F}}$ on the set $Q'_{j-1} \cup I_j$. By union bound, the above events both hold with probability $\geq 1 - (j+1)\varepsilon$, so that there exists $g, g' \in \mathsf{RS}[\mathbb{F}_p, d]$ such that $\hat{\mathcal{F}}(b_1, \ldots, b_{i-1}, \ell, b_{i+1}, \ldots b_m) = g(\ell)$ for all $\ell \in Q'_{j-1}$ and $\hat{\mathcal{F}}(b_1, \ldots, b_{i-1}, \ell, b_{i+1}, \ldots b_m) = g'(\ell)$ for all $\ell \in I_j$. However, since Q'_{j-1} intersects I_j on $d+1$ points it must be the case that $g = g'$. Hence, $\hat{\mathcal{F}}(b_1, \ldots, b_{i-1}, \ell, b_{i+1}, \ldots b_m) = g(\ell)$ for all $\ell \in Q'_{j-1} \cup I_j$. Since $Q_j \subseteq Q'_{j-1} \cup I_j$, this proves the induction step. Since $\left|Q'_{j-1} \cup I_j\right| \leq |S| + d + 2$ and $|Q_j| \leq |S| + d + 2$, it follows that this is valid for all S with $|S| \leq \hat{k} - d - 2$, which completes the proof of Claim 56. ◁

6.3 Step 3: A robust local characterization of low-degree polynomials

We prove Lemma 54. We begin by introducing some notation. For each $i \in [m]$ define $S_i \subseteq \mathbb{F}_p^i$ to be the projection of S to the first i coordinates, i.e. that

$$S_i := \{(a_1, \ldots, a_i) \in \mathbb{F}_p^i : \exists\, b_{i+1}, \ldots b_m \in \mathbb{F}_p \text{ s.t. } (a_1, \ldots, a_i, b_{i+1}, \ldots, b_m) \in S\} \ .$$

Note that $S_m = S$. For any set $R \subseteq \mathbb{F}_p^i$ and $b_{i+1}, \ldots, b_m \in \mathbb{F}_p$, we define the extension of R as

$$R^{(b_{i+1},\ldots,b_m)} := R \times \{(b_{i+1}, \ldots, b_m)\} \ .$$

We prove by induction that for every $i \in [m]$ and every $b_{i+1}, \ldots, b_m \in \mathbb{F}_p$ it holds that

$$\Pr[\exists\, g_i \in \mathsf{RS}^{\otimes i}[\mathbb{F}_p, d] \text{ s.t. } \hat{\mathcal{F}}(a_1, \ldots, a_i, b_{i+1}, \ldots, b_m) = g_i(a_1, \ldots, a_i)\ \forall (a_1, \ldots, a_i) \in S_i] \geq 1 - \varepsilon_i \ , \tag{2}$$

where $\varepsilon_i = |S| \sum_{j=1}^{i} (d+1)^{j-1} \varepsilon$. Note that the above probability is well-defined since we query $\hat{\mathcal{F}}$ on the set $S_i^{(b_{i+1},\ldots,b_m)}$, which contains at most $|S| \leq k' \leq k$ points. Equation (2) proves Lemma 54, as $S_m = S$ and $\varepsilon_m \leq |S| \cdot (\sum_{j=1}^{m} (d+1)^{j-1}) \cdot \varepsilon \leq \frac{(d+1)^m - 1}{d} \cdot |S| \varepsilon \leq 2(d+1)^{m-1} |S| \varepsilon$.

We first show the base case: $i = 1$. Let $R \subseteq S_1$ be a subset of size $\min(d+1, |S_1|)$. Query $\hat{\mathcal{F}}$ on $S_1^{(b_2,\ldots,b_m)}$, and let $g_1 \in \mathsf{RS}^{\otimes 1}[\mathbb{F}_p, d]$ be the univariate polynomial $g_1(x) = \sum_{a \in R} \delta_{a,R}(x) \cdot \hat{\mathcal{F}}(a, b_2, \ldots, b_m)$. Then,

$$\Pr[\exists\, g_1 \in \mathsf{RS}^{\otimes 1}[\mathbb{F}_p, d] \text{ s.t. } \hat{\mathcal{F}}(a_1, b_2, \ldots, b_m) = g_1(a_1) \text{ for all } a_1 \in S_1] \geq 1 - |S_1| \varepsilon = 1 - \varepsilon_1 \ .$$

This is because if $a_1 \in R$, then $\hat{\mathcal{F}}(a, b_2, \ldots, b_m) = g_1(a)$ is trivially true by definition of g_1, and if $a_1 \in S_1 \setminus R$ then this is true because

$$\hat{\mathcal{F}}(a_1, b_2, \ldots, b_m) = \sum_{a \in R} \delta_{a,R}(a_1) \cdot \hat{\mathcal{F}}(a, b_2, \ldots, b_m) = g_1(a_1) \ ,$$

with probability $1 - \varepsilon$, by the assumption on $\hat{\mathcal{F}}$.

We now show the induction step. Suppose that Equation (2) holds for $i-1$ and every $b_i, \ldots, b_m \in \mathbb{F}_p$. Let $R_i = S_i \cup (S_{i-1} \times \{0, \ldots, d\})$. Fix $b_{i+1}, \ldots, b_m \in \mathbb{F}_p$, and query $\hat{\mathcal{F}}$ on $R_i^{(b_{i+1},\ldots,b_m)}$. Note that this is well-defined since $\left|R_i^{(b_{i+1},\ldots,b_m)}\right| \leq |S_i| + (d+1)|S_{i-1}| \leq (d+2)|S| \leq (d+2)k' \leq k$.

We have that $S_{i-1} \times \{0, \ldots, d\} \subseteq R_i$, and so for every $j \in \{0, \ldots, d\}$, the induction hypothesis implies (by setting $b_i = j$) that with probability at least $1 - \varepsilon_{i-1}$ there exists $g_{i-1}^{(j)} \in \mathsf{RS}^{\otimes i-1}[\mathbb{F}_p, d]$ such that $\hat{\mathcal{F}}(a_1, \ldots, a_{i-1}, j, b_{i+1}, \ldots, b_m) = g_{i-1}^{(j)}(a_1, \ldots, a_{i-1})$ for every $(a_1, \ldots, a_{i-1}) \in S_{i-1}$.

For $j \in \{0, \ldots, d\}$, let $\delta_j(x) := \delta_{j,\{0,\ldots,d\}}(x)$. Let $g_i \in \mathsf{RS}^{\otimes i}[\mathbb{F}_p, d]$ be defined as $g_i(x_1, \ldots, x_i) = \sum_{j=0}^{d} \delta_j(x_i) \cdot g_{i-1}^{(j)}(x_1, \ldots, x_{i-1})$. We show that

$$\Pr[\hat{\mathcal{F}}(a_1, \ldots, a_i, b_{i+1}, \ldots, b_m) = g_i(a_1, \ldots, a_i) \text{ for all } (a_1, \ldots, a_i) \in S_i] \geq 1 - \varepsilon_i \ .$$

Indeed, note that for any $(a_1, \ldots, a_i) \in S_i$ we have $(a_1, \ldots, a_{i-1}) \in S_{i-1}$, and so with probability $1 - (d+1)\varepsilon_{i-1}$ all the $g_{i-1}^{(j)}$'s exist, and so we have that for all $(a_1, \ldots, a_i) \in S_i$ it holds that

$$g_i(a_1, \ldots, a_i) = \sum_{j=0}^{d} \delta_j(a_i) \cdot g_{i-1}^{(j)}(a_1, \ldots, a_{i-1}) = \sum_{j=0}^{d} \delta_j(a_i) \cdot \hat{\mathcal{F}}(a_1, \ldots, a_{i-1}, j, b_{i+1}, \ldots, b_m)$$

By the assumption on $\hat{\mathcal{F}}$, we have

$$\Pr[\sum_{j=0}^{d} \delta_j(a_i) \cdot \hat{\mathcal{F}}(a_1, \ldots, a_{i-1}, j, b_{i+1}, \ldots, b_m) \\ = \hat{\mathcal{F}}(a_1, \ldots, a_i, b_{i+1}, \ldots, b_m) \ \forall (a_1, \ldots, a_i) \in S_i] \geq 1 - |S_i|\varepsilon \ .$$

Combining the two equations shows that $g_i(a_1, \ldots, a_i) = \hat{\mathcal{F}}(a_1, \ldots, a_i, b_{i+1}, \ldots, b_m)$ with probability $1 - (d+1)\varepsilon_{i-1} - |S_i|\varepsilon \geq 1 - (d+1)\varepsilon_{i-1} - |S|\varepsilon = 1 - \varepsilon_i$, as required, completing the proof.

6.4 Step 4: Completing the proof

The following generic lemma immediately implies Lemma 55.

▶ **Lemma 57.** *Let $\mathbf{C} \subseteq \mathbb{F}^n$ be a linear code, and let $\mathcal{F}$ be a k-non-signaling function. Suppose that $\Pr[\langle \alpha, \mathcal{F} \rangle = 0] \geq 1 - \varepsilon$ for every $\alpha \in \mathbf{C}^{\perp}_{\leq k}$. Then there exists a $k/2$-non-signaling function $\mathcal{G}$ that is $(\mathbf{C}, k/2)$-supported such that $\Delta_{k/2}(\mathcal{F}, \mathcal{G}) \leq (q^{k/2} + 1) \cdot q^{k/4} \cdot 2\varepsilon$.*

Proof. Let $\alpha, \alpha' \in \mathbb{F}^n_{\leq k/2}$ such that $\alpha - \alpha' \in \mathbf{C}^{\perp}$. Then

$$\left|\widehat{\mathcal{F}}(\alpha) - \widehat{\mathcal{F}}(\alpha')\right| = \left|\frac{1}{q^n} \sum_{j \in \mathbb{F}_p} \omega^{-j} (\Pr[\mathrm{Tr}(\langle \alpha, \mathcal{F} \rangle) = j] - \Pr[\mathrm{Tr}(\langle \alpha', \mathcal{F} \rangle) = j])\right| \\ \leq \frac{1}{q^n} \sum_{j \in \mathbb{F}_p} \left|\omega^{-j}\right| \left|\Pr[\mathrm{Tr}(\langle \alpha, \mathcal{F} \rangle) = j] - \Pr[\mathrm{Tr}(\langle \alpha', \mathcal{F} \rangle) = j]\right| \\ \leq \frac{1}{q^n} \sum_{j \in \mathbb{F}_p} \left|\Pr[\mathrm{Tr}(\langle \alpha, \mathcal{F} \rangle) = j \wedge \langle \alpha - \alpha', \mathcal{F} \rangle \neq 0] - \Pr[\mathrm{Tr}(\langle \alpha', \mathcal{F} \rangle) = j \wedge \langle \alpha - \alpha', \mathcal{F} \rangle \neq 0]\right| \\ \leq \frac{1}{q^n} \sum_{j \in \mathbb{F}_p} \left(\left|\Pr[\mathrm{Tr}(\langle \alpha, \mathcal{F} \rangle) = j \wedge \langle \alpha - \alpha', \mathcal{F} \rangle \neq 0]\right| + \left|\Pr[\mathrm{Tr}(\langle \alpha', \mathcal{F} \rangle) = j \wedge \langle \alpha - \alpha', \mathcal{F} \rangle \neq 0]\right|\right) \\ \leq \frac{2}{q^n} \cdot \Pr[\langle \alpha - \alpha', \mathcal{F} \rangle \neq 0] \leq \frac{2\varepsilon}{q^n} \ ,$$

since $\alpha - \alpha' \in \mathbf{C}^{\perp}$ and $\mathrm{wt}(\alpha - \alpha') \leq k$. By Lemma 41, there exists a quasi-distribution $\mathcal{Q}$ supported on $\mathbf{C}$ such that $\Delta_{k/2}(\mathcal{F}, \mathcal{Q}) \leq q^{k/4} \cdot 2\varepsilon$. By Lemma 26, there exists a quasi-distribution $\mathcal{Q}'$ supported on $\mathbf{C}$ such that $\Delta_{k/2}(\mathcal{Q}, \mathcal{Q}') \leq q^{k/2} \cdot q^{k/4} \cdot 2\varepsilon$. Letting $\mathcal{G}$ be the $k/2$-non-signaling function corresponding to $\mathcal{Q}'$ completes the proof. ◀

7 Local characterizations and linear proofs

We prove Theorem 8 in this section. For this section, we let $k' \leq k$ be an integer. We let $\mathbf{C} \subseteq \mathbb{F}^n$ be a linear code, and $T \subseteq \mathbb{F}^n$ be a set of constraints. Given a k-non-signaling function $\mathcal{F}$, we say that $\mathcal{F}$ satisfies a constraint $\alpha \in \mathbb{F}^n_{\leq k}$ if $\Pr[\langle \alpha, \mathcal{F} \rangle = 0] = 1$.

▶ **Definition 58.** *We let* $\mathsf{Consistent}(T, k)$ *denote the set of k-non-signaling functions $\mathcal{F}$ where $\Pr[\langle \alpha, \mathcal{F} \rangle = 0] = 1$ for every $\alpha \in T$. That is,* $\mathsf{Consistent}(T, k)$ *is the set of k-non-signaling functions that are consistent with T.*

We note that by Lemma 44, $\mathsf{Consistent}(\mathbf{C}^{\perp}_{\leq k'}, k)$ is the set of k-non-signaling functions that are $(\mathbf{C}, k')$-explainable.

With the above definition, the definition of local characterization can be rephrased as follows.

▶ **Definition 59.** *For $\ell \le k' \le k$, a set of constraints $T \subseteq \mathbf{C}^{\perp}_{\le \ell}$ is a ℓ-local characterization of $(\mathbf{C}, k', k)$ if* $\mathsf{Consistent}(T, k)$ *equals the set of k-non-signaling functions that are $(\mathbf{C}, k')$-explainable, i.e. that* $\mathsf{Consistent}(T, k) = \mathsf{Consistent}(\mathbf{C}^{\perp}_{\le k'}, k)$.

In this language, [4] shows that $T = \{e_x + e_y - e_{x+y} : x, y \in \{0,1\}^n\}$ is a 3-local characterization of $(\mathbf{C}, k-1, k)$, where $\mathbf{C}$ is the Hadamard code.

We briefly recall the definition of a k-local linear proof introduced in Section 1.2.

▶ **Definition 60** (k-local linear proof)**.** *Given a constraint set T and $\alpha \in \mathbb{F}^n$, we write $T \vdash_k \alpha$ if there exists a sequence $(\alpha_0{:=}0^n, \alpha_1, \ldots, \alpha_{r-1}, \alpha_r{:=}\alpha)$ with each $\alpha_i \in \mathbb{F}^n$ such that, for every $i \in [r]$, one of the following holds:*

- *$\exists\, j < i$ and $b \in \mathbb{F}$ such that $\alpha_i = b\alpha_j$*
- *$\exists\, j < i$ and $\gamma \in T$ such that $|\mathrm{supp}(\alpha_j) \cup \mathrm{supp}(\gamma)| \le k$ and $\alpha_i = \alpha_j + \gamma$*
- *$\exists\, j_1, j_2 < i$ such that $|\mathrm{supp}(\alpha_{j_1}) \cup \mathrm{supp}(\alpha_{j_2})| \le k$ and $\alpha_i = \alpha_{j_1} + \alpha_{j_2}$.*

Theorem 8 is stated formally as the theorem below.

▶ **Theorem 61** (Formal version of Theorem 8)**.** *k-local linear proofs are complete and sound for k-non-signaling functions. In particular, for $\ell \le k' \le k$, a set of constraints $T \subseteq \mathbf{C}^{\perp}_{\le \ell}$ is a ℓ-local characterization of $(\mathbf{C}, k', k)$ if and only if $T \vdash_k \mathbf{C}^{\perp}_{\le k'}$.*

The proof of Theorem 61 relies on the notion of a k-local subspace, which we define below.

▶ **Definition 62.** *A k-local subspace $\mathcal{V}$ is a subset of $\mathbb{F}^n_{\le k}$ where $\mathcal{V}_{\subseteq S} \subseteq \mathbb{F}^n$ is a linear subspace for every $S \subseteq [n]_{\le k}$.*

We prove Theorem 61 by showing the following three lemmas.

▶ **Lemma 63** (Soundness)**.** *If $T \vdash_k \alpha$, then* $\mathsf{Consistent}(T, k) = \mathsf{Consistent}(T \cup \{\alpha\}, k)$.

▶ **Lemma 64.** *For every k-local subspace $\mathcal{V} \subseteq \mathbb{F}^n_{\le k}$, there exists a k-non-signaling function $\mathcal{F}$ such that $\Pr[\langle \alpha, \mathcal{F} \rangle = 0] = 1$ for every $\alpha \in \mathcal{V}$, and $\Pr[\langle \alpha, \mathcal{F} \rangle = 0] = \frac{1}{|\mathbb{F}|}$ otherwise.*

▶ **Lemma 65.** *$\{\alpha \in \mathbb{F}^n : T \vdash_k \alpha\} \subseteq \mathbb{F}^n_{\le k}$ is a k-local subspace.*

The following corollary follows immediately from Lemma 64 and Lemma 65.

▶ **Corollary 66** (Strong completeness)**.** *There exists a k-non-signaling function such that $\Pr[\langle \alpha, \mathcal{F} \rangle = 0] = 1$ for every α where $T \vdash_k \alpha$, and $\Pr[\langle \alpha, \mathcal{F} \rangle = 0] = \frac{1}{|\mathbb{F}|}$ otherwise.*

Proof of Theorem 61. Completeness and soundness are shown in Corollary 66 and Lemma 63. It remains to show the equivalence for local characterizations.

Suppose that $T \vdash_k \mathbf{C}^{\perp}_{\le k'}$. Then by Lemma 63 we have that $\mathsf{Consistent}(T, k)$ and $\mathsf{Consistent}(T \cup \mathbf{C}^{\perp}_{\le k'}, k)$ are equal. Since $T \subseteq \mathbf{C}^{\perp}_{\le \ell}$ and $\ell \le k'$, we get that $T \subseteq \mathbf{C}^{\perp}_{\le k'}$. Hence, $\mathsf{Consistent}(T, k) = \mathsf{Consistent}(T \cup \mathbf{C}^{\perp}_{\le k'}, k) = \mathsf{Consistent}(\mathbf{C}^{\perp}_{\le k'}, k)$, as required.

Conversely, suppose that T is an ℓ-local characterization of $(\mathbf{C}, k', k)$. By Lemma 64 and Lemma 65, there exists a k-non-signaling function $\mathcal{F}$ such that $\Pr[\langle \alpha, \mathcal{F} \rangle = 0] = 1$ for every $\alpha \in \mathbb{F}^n_{\le k}$ such that $T \vdash_k \alpha$, and $\Pr[\langle \alpha, \mathcal{F} \rangle = 0] = \frac{1}{|\mathbb{F}|}$ otherwise. Since $T \vdash_k \alpha$ for every $\alpha \in T$, it follows that $\mathcal{F} \in \mathsf{Consistent}(T, k)$, which implies that $\mathcal{F} \in \mathsf{Consistent}(\mathbf{C}^{\perp}_{\le k'}, k)$ as T is an ℓ-local characterization of $(\mathbf{C}, k', k)$. This implies that $T \vdash_k \alpha$ for all $\alpha \in \mathbf{C}^{\perp}_{\le k'}$, since for all such α it holds that $\Pr[\langle \alpha, \mathcal{F} \rangle = 0] = 1$, and thus $T \vdash_k \alpha$. Hence, $T \vdash_k \mathbf{C}^{\perp}_{\le k'}$, as required. ◀

7.1 Proof of Lemma 63

It is clear that from the definition that $\mathsf{Consistent}(T,k) \supseteq \mathsf{Consistent}(T \cup \{\alpha\},k)$ for all $\alpha \in \mathbb{F}^n$. Below we prove the containment in the other direction. Suppose that $T \vdash_k \alpha$, and let $(\alpha_0 = 0^n, \alpha_1, \ldots, \alpha_r = \alpha)$ be a k-local proof of α from T. Let $\mathcal{F} \in \mathsf{Consistent}(T,k)$, that is, $\mathcal{F}$ is a k-non-signaling function such that $\forall \gamma \in T$, $\Pr[\langle \gamma, \mathcal{F}\rangle = 0] = 1$. We prove by induction that for $i \in [r]$ it holds that $\Pr[\langle \alpha_i, \mathcal{F}\rangle = 0] = 1$.

For the base case of $i = 0$ it must be the case that $\alpha_0 = 0^n$. Therefore, $\Pr[\langle \alpha_0, \mathcal{F}\rangle = 0] = 1$. For the induction step let $i \geq 1$, and consider the following three cases.

1. There exists $j < i$ and $b \in \mathbb{F} \setminus \{0^n\}$ such that $\alpha_i = b\alpha_j$. Then,

$$\Pr[\langle \alpha_i, \mathcal{F}\rangle = 0] = \Pr[b\langle \alpha_j, \mathcal{F}\rangle = 0] = \Pr[\langle \alpha_j, \mathcal{F}\rangle = 0] = 1 \ ,$$

   where the last equality uses the induction hypothesis.
2. There exist $j < i$ and $\gamma \in T$ such that $\alpha_i = \alpha_j + \gamma$ with $|\mathrm{supp}(\alpha_j) \cup \mathrm{supp}(\gamma)| \leq k$. Since $\mathcal{F} \in \mathsf{Consistent}(T,k)$ we have that $\Pr[\langle \gamma, \mathcal{F}\rangle = 0] = 1$, as $\gamma \in T$. Therefore,

$$\Pr[\langle \alpha_i, \mathcal{F}\rangle = 0] = \Pr[\langle \alpha_j, \mathcal{F}\rangle + \langle \gamma, \mathcal{F}\rangle = 0] \geq \Pr[\langle \alpha_j, \mathcal{F}\rangle = 0 \wedge \langle \gamma, \mathcal{F}\rangle = 0] = 1 \ ,$$

   as required. Note that $\Pr[\langle \alpha_j, \mathcal{F}\rangle = 0 \wedge \langle \gamma, \mathcal{F}\rangle = 0]$ is well-defined because we have $|\mathrm{supp}(\alpha_{j_1}) \cup \mathrm{supp}(\gamma)| \leq k$.
3. There exist $j_1, j_2 < i$ such that $\alpha_i = \alpha_{j_1} + \alpha_{j_2}$ and $|\mathrm{supp}(\alpha_{j_1}) \cup \mathrm{supp}(\alpha_{j_2})| \leq k$. By the induction hypothesis we know that $\Pr[\langle \alpha_{j_1}, \mathcal{F}\rangle = 0] = 1$ and $\Pr[\langle \alpha_{j_2}, \mathcal{F}\rangle = 0] = 1$. Thus,

$$\Pr[\langle \alpha_i, \mathcal{F}\rangle = 0] = \Pr[\langle \alpha_{j_1}, \mathcal{F}\rangle + \langle \alpha_{j_2}, \mathcal{F}\rangle = 0] \geq \Pr[\langle \alpha_{j_1}, \mathcal{F}\rangle = 0 \wedge \langle \alpha_{j_2}, \mathcal{F}\rangle = 0] = 1 \ ,$$

   and therefore $\Pr[\langle \alpha_i, \mathcal{F}\rangle = 0] = 1$. Again, we require $|\mathrm{supp}(\alpha_{j_1}) \cup \mathrm{supp}(\alpha_{j_2})| \leq k$ in order for the last probability to be well-defined.

In particular, this implies that $\Pr[\langle \alpha, \mathcal{F}\rangle = 0] = \Pr[\langle \alpha_r, \mathcal{F}\rangle = 0] = 1$, and hence $\mathcal{F} \in \mathsf{Consistent}(T \cup \{\alpha\},k)$. Therefore $\mathsf{Consistent}(T,k) \subseteq \mathsf{Consistent}(T \cup \{\alpha\},k)$, which completes the proof of Lemma 63.

7.2 Proof of Lemma 64

We define $\mathcal{F}$ specifying its local distributions $\mathcal{F}_S$ for each $S \subseteq [n]_{\leq k}$. We define the function $\mathcal{F}_S \colon \mathbb{F}^S \to \mathbb{C}$ by specifying its (local) Fourier coefficients as follows. We set the Fourier coefficient $\widehat{\mathcal{F}_S}(\alpha)$ to be $\frac{1}{q^{|S|}}$ if $\alpha \in \mathcal{V}$, and 0 otherwise.

We now show that each $\mathcal{F}_S$ is a distribution. For any $f \colon S \to \mathbb{F}$ we have

$$\mathcal{F}_S(f) = \sum_{\alpha \in \mathbb{F}^S} \widehat{\mathcal{F}_S}(\alpha)\chi_\alpha(f) = \sum_{\alpha \in \mathcal{V}_{\subseteq S}} \frac{1}{q^{|S|}}\chi_\alpha(f) = \frac{1}{q^{|S|}} \sum_{\alpha \in \mathcal{V}_{\subseteq S}} \omega^{\mathrm{Tr}(\langle \alpha, f\rangle)} \ .$$

For each $b \in \mathbb{F}$, let $\mathcal{V}_b \subseteq \mathcal{V}_{\subseteq S}$ be the set of $\alpha \in \mathcal{V}_{\subseteq S}$ where $\langle \alpha, f\rangle = b$. Let $\pi \colon \mathcal{V}_{\subseteq S} \to \mathbb{F}$ be the map where $\pi(\alpha) = \langle \alpha, f\rangle$. Since $\mathcal{V}_{\subseteq S}$ is a subspace, π is a homomorphism. It follows that either $\mathcal{V}_0 = \mathcal{V}_{\subseteq S}$ or $|\mathcal{V}_b| = |\mathcal{V}_0|$ for every $b \in \mathbb{F}$. In the first case, $\sum_{\alpha \in \mathcal{V}_{\subseteq S}} \omega^{\mathrm{Tr}(\langle \alpha, f\rangle)} = |\mathcal{V}_{\subseteq S}| \geq 0$. In the second case,

$$\sum_{\alpha \in \mathcal{V}_{\subseteq S}} \omega^{\mathrm{Tr}(\langle \alpha, f\rangle)} = \sum_{b \in \mathbb{F}} \sum_{\alpha \in \mathcal{V}_b} \omega^{\mathrm{Tr}(b)} = \sum_{b \in \mathbb{F}} |\mathcal{V}_b| \omega^{\mathrm{Tr}(b)} = |\mathcal{V}_0| \sum_{b \in \mathbb{F}} \omega^{\mathrm{Tr}(b)} = 0 \ .$$

This implies that in either case, $\mathcal{F}_S(f) \geq 0$, and so $\mathcal{F}_S$ is a distribution.

We now show that the collection of local distributions $\{\mathcal{F}_S\}_{S\subseteq[n]_{\leq k}}$ is indeed non-signaling. This follows from Lemma 35. If $\alpha \in \mathcal{V}$ then we have that $q^{|S|}\widehat{\mathcal{F}_S}(\alpha) = 1 = q^{|R|}\widehat{\mathcal{F}_R}(\alpha)$ for every $S, R \in [n]_{\leq k}$ such that $\mathrm{supp}(\alpha) \subseteq S \cap R$, and otherwise we have $q^{|S|}\widehat{\mathcal{F}_S}(\alpha) = 0 = q^{|R|}\widehat{\mathcal{F}_R}(\alpha)$. Thus, the collection of local distributions is a k-non-signaling function $\mathcal{F}$.

It remains to show that $\mathcal{F}$ satisfies the desired property. Observe that for every α, $q^n\widehat{\mathcal{F}}(\alpha) = q^{|\mathrm{supp}(\alpha)|}\widehat{\mathcal{F}}_{\mathrm{supp}(\alpha)}(\alpha) = 1$ if $\alpha \in \mathcal{V}$, and otherwise $\widehat{\mathcal{F}}(\alpha) = 0$. By Lemma 29 it follows that $\mathcal{F}$ has the desired properties.

7.3 Proof of Lemma 65

Let $\mathcal{V} = \{\alpha \in \mathbb{F}^n : T \vdash_k \alpha\}$. We show that $\mathcal{V}$ is a k-local subspace. Let $S \subseteq [n]_{\leq k}$. We need to show that $\mathcal{V}_{\subseteq S}$ is a linear subspace of $\mathbb{F}^n$. We first observe that 0^n is always in the set, as $T \vdash_k 0^n$ always.

Let $\alpha \in \mathcal{V}_{\subseteq S}$ and let $b \in \mathbb{F} \setminus \{0\}$. Then we have that $T \vdash_k \alpha$ which implies that $T \vdash_k b\alpha$. Since $\mathrm{supp}(b\alpha) = \mathrm{supp}(\alpha) \subseteq S$, it follows that $b\alpha \in \mathcal{V}_{\subseteq S}$.

Let $\alpha, \beta \in \mathcal{V}_{\subseteq S}$. Then, since $|\mathrm{supp}(\alpha) \cup \mathrm{supp}(\beta)| \leq |S| \leq k$ we have that $(\alpha, \beta, \alpha+\beta)$ is a hyperedge in Γ_k. Thus, since $T \vdash_k \{\alpha, \beta\}$ it follows that $T \vdash_k \alpha + \beta$. Since $\mathrm{supp}(\alpha+\beta) \subseteq \mathrm{supp}(\alpha) \cup \mathrm{supp}(\beta) \subseteq S$, it follows that $\alpha + \beta \in \mathcal{V}_{\subseteq S}$.

We have thus shown that $\mathcal{V}_{\subseteq S}$ is a linear subspace of $\mathbb{F}^n$, which completes the proof.

8 Low-degree testing fails for small locality

In this section, we prove Theorem 3. The proof relies heavily on Theorem 8.

We let $\mathbf{C}$ be the linear code of m-variate polynomials $P\colon \mathbb{F}^m \to \mathbb{F}$ of total degree at most d, with $m \geq 2$, and let T be the set of α's in $\mathbf{C}^\perp$ where the $\mathrm{supp}(\alpha)$ is contained in exactly one line.

We define the *rank* of an element in $\mathbf{C}^\perp$ to be

$$\mathsf{rank}_T(\alpha) := \min_{T' \subseteq T : \alpha \in \mathrm{span}(T')} |T'| \ .$$

Note that since $\mathrm{span}(T) = \mathbf{C}^\perp$, the rank of α is well-defined for all $\alpha \in \mathbf{C}^\perp$.

We let T_0 denote the subset of T that only contains elements whose support is *evenly-spaced* along a line and has weight $d+2$. With this notation, the non-signaling evenly-spaced test (i) samples $\alpha \leftarrow T_0$ uniformly at random, and (ii) checks that $\langle \alpha, \mathcal{F} \rangle = 0$.

The main theorem we prove is stated below, and is the formal statement of Theorem 3.

▶ **Theorem 67** (Formal version of Theorem 3). *For every k with $2d+2 \leq k < \frac{3}{16}(d+2)^2$, there exists a k-non-signaling function such that $\Pr[\langle \alpha, \mathcal{F} \rangle = 0] = 1$ for every $\alpha \in T_{\leq k}$, and yet $\Delta_{2d+2}(\mathcal{F}, \mathcal{F}') \geq (1 - \frac{1}{|\mathbb{F}|})$ for every $(2d+2)$-non-signaling function $\mathcal{F}'$ that is $(\mathbf{C}, 2d+2)$-explainable.*

We begin the proof of Theorem 67 by showing the following lemma. This lemma follows from earlier statements, and outlines a sufficient condition to prove Theorem 67

▶ **Lemma 68.** *Suppose that there exists $\alpha^* \in \mathbf{C}^\perp$ with $\mathrm{wt}(\alpha^*) = 2d+2$ such that for every $k < \frac{3}{16}(d+2)^2$ it holds that $T \not\vdash_k \alpha^*$. Then for every k with $2d+2 \leq k < \frac{3}{16}(d+2)^2$ there exists a k-non-signaling function $\mathcal{F}$ such that $\Pr[\langle \alpha, \mathcal{F} \rangle = 0] = 1$ for every $\alpha \in T_{\leq k}$, and yet $\Delta_{2d+2}(\mathcal{F}, \mathcal{F}') \geq 1 - \frac{1}{|\mathbb{F}|}$ for every $(2d+2)$-non-signaling function $\mathcal{F}'$ that is $(\mathbf{C}, 2d+2)$-explainable.*

Proof. Applying Corollary 66, for every k with $2d+2 \leq k < \frac{3}{16}(d+2)^2$, we get that there exists a k-non-signaling function $\mathcal{F}$ such that $\Pr[\langle \alpha, \mathcal{F} \rangle = 0] = 1$ for every $\alpha \in T_{\leq k}$ and $\Pr[\langle \alpha^*, \mathcal{F} \rangle = 0] = \frac{1}{|\mathbb{F}|}$. Let $\mathcal{F}'$ be a $(2d+2)$-non-signaling function that is $(\mathbf{C}, 2d+2)$-explainable. Since for every $S \subseteq \mathbb{F}^m$ with $|S| \leq 2d+2$ we have that $\Pr[\mathcal{F}'(S) \in \mathbf{C}|_S] = 1$ and $\alpha^* \in \mathbf{C}^\perp$ has $\mathrm{wt}(\alpha^*) = 2d+2$, it follows that $\Pr[\langle \alpha^*, \mathcal{F}' \rangle = 0] = 1$. Therefore, $\Delta_{2d+2}(\mathcal{F}, \mathcal{F}') \geq |\Pr[\langle \alpha^*, \mathcal{F} \rangle = 0] - \Pr[\langle \alpha^*, \mathcal{F}' \rangle = 0]| = 1 - \frac{1}{|\mathbb{F}|}$. ◀

By Lemma 68 it suffices to find such an α^*. We let $\alpha^* \in \mathbf{C}^\perp$ be any constraint where $\mathrm{supp}(\alpha)$ has size $2d+2$ and is contained on the curve $x_1^2 - x_2 = 0$ embedded on the plane $x_3 = x_4 = \cdots = x_m = 0$ in $\mathbb{F}^m$. We note that α^* is one of the constraints that checks that $P(t, t^2, 0, \ldots, 0)$ is a univariate polynomial of degree at most $2d$ in t.

We show that α^* satisfies the desired properties in two main lemmas. We first show the following generic lemma, which gives us a way to prove that $T \nvdash_k \alpha^*$.

▶ **Lemma 69** (Interval cut Lemma). *Fix $\alpha \in \mathbf{C}^\perp$. Suppose that there exists $r \in \mathbb{R}$ with $2 \leq r \leq \mathsf{rank}_T(\alpha)$ such that for every $\beta \in \mathbf{C}^\perp$ with $\mathsf{rank}_T(\beta) \in [r/2, r)$ it holds that $T \nvdash_k \beta$. Then $T \nvdash_k \alpha$.*

We then show that every $\beta \in \mathbf{C}^\perp$ of rank in $[(d+2)/4, (d+2)/2)$ must have large weight, implying that they are not provable from T when k is small.

▶ **Lemma 70.** *For every $\beta \in \mathbf{C}^\perp$ with $\mathsf{rank}_T(\beta) \in [(d+2)/4, (d+2)/2)$ it holds that $\mathrm{wt}(\beta) \geq \frac{3}{16}(d+2)^2$. In particular, if $k < \frac{3}{16}(d+2)^2$ then $T \nvdash_k \beta$.*

With the above two lemmas, we now finish the proof of Theorem 67.

Proof of Theorem 67. Let $k < \frac{3}{16}(d+2)^2$. We first show that $\mathsf{rank}_T(\alpha^*) \geq d+1$. Since the curve $x_1^2 - x_2 = 0$ is irreducible in $\mathbb{F}[x_1, x_2, \ldots, x_m]$, any line L intersects the curve on at most 2 distinct points. It follows that $\mathsf{rank}_T(\alpha^*) \geq (2d+2)/2 = d+1$, as any constraint $\beta \in T$ can only have at most 2 points on the curve $x_1^2 - x_2 = 0$.

Since $k < \frac{3}{16}(d+2)^2$, Lemma 70 implies that $T \nvdash_k \beta$ for every β with $\mathsf{rank}_T(\beta) \in [(d+2)/4, (d+2)/2)$. Thus, by Lemma 69 it follows that $T \nvdash_k \alpha^*$. Hence, α^* satisfies the assumptions of Lemma 68, and so applying Lemma 68 completes the proof of Theorem 67. ◀

Next we turn to the proofs of Lemma 69 and Lemma 70.

Proof of Lemma 69. First, observe that by definition of rank, $\mathsf{rank}_T(\alpha_1 + \alpha_2) \leq \mathsf{rank}_T(\alpha_1) + \mathsf{rank}_T(\alpha_2)$. By the assumption of the lemma, there exists $r \in \mathbb{R}$ with $2 \leq r \leq \mathsf{rank}_T(\alpha)$ such that for every $\beta \in \mathbf{C}^\perp$ with $\mathsf{rank}_T(\beta) \in [r/2, r)$ it holds that $T \nvdash_k \beta$. We need to show that $T \nvdash_k \alpha$.

Suppose toward a contradiction that $T \vdash_k \alpha$. Then there exists a path $(\alpha_1, \ldots, \alpha_t = \alpha)$ in $\Gamma_k(\mathbf{C}^\perp, T)$ from 0^n to α. Let S_1 be the set of α_i's such that $\mathsf{rank}_T(\alpha_i) < r/2$, and let S_2 be the set of α_i's such that $\mathsf{rank}_T(\alpha_i) \geq r$. Note that $S_1 \cup S_2 = \{\alpha_1, \ldots, \alpha_t\}$, as otherwise there would exist some i such that α_i has rank in $[r/2, r)$, which would contradict the assumption that $T \vdash_k \alpha_i$ for all $i \in [t]$.

Since $\mathsf{rank}_T(\alpha) \geq r$ it follows that $\alpha \in S_2$, and hence $S_2 \neq \emptyset$. Let ℓ be the smallest index such that $\alpha_\ell \in S_2$. We have that $\alpha_\ell \neq 0^n$ since $\alpha_\ell \in S_2$, and there does not exist $i < \ell$ and $b \in \mathbb{F} \setminus \{0\}$ such that $\alpha_\ell = b\alpha_i$, as then $\mathsf{rank}_T(\alpha_i) = \mathsf{rank}_T(\alpha_\ell) \geq r$, thus contradicting the minimality of ℓ. Suppose that there exists $i < \ell$ and $\gamma \in T$ such that $\alpha_\ell = \alpha_i + \gamma$. By the minimality of ℓ, we must have that $\alpha_i \in S_1$, and hence $r \leq \mathsf{rank}_T(\alpha_\ell) \leq \mathsf{rank}_T(\alpha_i) + \mathsf{rank}_T(\gamma) < r/2 + 1 \leq r/2 + r/2 = r$, which is also a contradiction. Therefore, there must either exist $j_1, j_2 < \ell$ such that $\alpha_\ell = \alpha_{j_1} + \alpha_{j_2}$. By the minimality of ℓ, we must

have that $\alpha_{j_1}, \alpha_{j_2} \in S_1$, and hence $r \leq \mathsf{rank}_T(\alpha_\ell) \leq \mathsf{rank}_T(\alpha_{j_1}) + \mathsf{rank}_T(\alpha_{j_1}) < r/2 + r/2 = r$, which is, again, a contradiction. In all cases we have reached a contradiction to the assumption that $T \vdash_k \alpha$, which completes the proof of Lemma 69. ◀

▶ Remark 71. We note that in the foregoing proof we only required that rank_T is subadditive, i.e., that $\mathsf{rank}_T(\alpha_1 + \alpha_2) \leq \mathsf{rank}_T(\alpha_1) + \mathsf{rank}_T(\alpha_2)$, $\mathsf{rank}_T(\alpha) = 1$ for every $\alpha \in T$, and $\mathsf{rank}_T(0^n) = 0$. Thus, the Interval Cut Lemma holds for any such subadditive function.

Proof of Lemma 70. Let $\beta \in \mathbf{C}^\perp$ be such that $\mathsf{rank}_T(\beta) = r \in [(d+2)/4, (d+2)/2)$. Then there exist lines $L_1, \ldots, L_r$ such that $\beta = \sum_{i=1}^{r} \beta_i$ where $\mathrm{supp}(\beta_i) \subseteq L_i$. The L_i's must be distinct, as otherwise we could add two constraints contained in the same line and we would then get $\mathsf{rank}_T(\beta) < r$. We have that $\mathrm{wt}(\beta_i) \geq d+2$ for each i. Hence, $\mathrm{wt}(\beta) \geq r(d+2) - 2\binom{r}{2}$, since each β_i contributes at least $d+2$ to the weight, and there are at most $\binom{r}{2}$ intersection points as each of the r lines is distinct. The function $f(r) = r(d+2) - r^2$ for $r \in [\frac{d+2}{4}, \frac{d+2}{2})$ is minimized when $r = \frac{d+2}{4}$, and hence $\mathrm{wt}(\beta) \geq r(d+2) - r^2 \geq \frac{3}{16}(d+2)^2$, which completes the proof. ◀

References

1. Samson Abramsky and Adam Brandenburger. The sheaf-theoretic structure of non-locality and contextuality. *New Journal of Physics*, 13(11):113036, 2011.
2. Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *Journal of the ACM*, 45(3):501–555, 1998. Preliminary version in FOCS '92.
3. Manuel Blum, Michael Luby, and Ronitt Rubinfeld. Self-Testing/Correcting with Applications to Numerical Problems. *Journal of Computer and System Sciences*, 47(3):549–595, 1993.
4. Alessandro Chiesa, Peter Manohar, and Igor Shinkar. Testing Linearity against Non-Signaling Strategies. In *Proceedings of the 33rd Annual Conference on Computational Complexity*, CCC '18, pages 17:1–17:37, 2018.
5. Alessandro Chiesa, Peter Manohar, and Igor Shinkar. Probabilistic Checking Against Non-Signaling Strategies from Linearity Testing. In *Proceedings of the 10th Innovations in Theoretical Computer Science Conference*, ITCS '19, pages 25:1–25:17, 2019.
6. Oded Goldreich and Madhu Sudan. Locally testable codes and PCPs of almost-linear length. *Journal of the ACM*, 53:558–655, July 2006. Preliminary version in STOC '02.
7. Yael Kalai, Ran Raz, and Ron Rothblum. Delegation for Bounded Space. In *Proceedings of the 45th ACM Symposium on the Theory of Computing*, STOC '13, pages 565–574, 2013.
8. Yael Tauman Kalai, Ran Raz, and Oded Regev. On the Space Complexity of Linear Programming with Preprocessing. In *Proceedings of the 7th Innovations in Theoretical Computer Science Conference*, ITCS '16, pages 293–300, 2016.
9. Yael Tauman Kalai, Ran Raz, and Ron D. Rothblum. How to delegate computations: the power of no-signaling proofs. In *Proceedings of the 46th ACM Symposium on Theory of Computing*, STOC '14, pages 485–494, 2014. Full version available at `https://eccc.weizmann.ac.il/report/2013/183/`.
10. Leonid A Khalfin and Boris S Tsirelson. Quantum/classical correspondence in the light of Bell's inequalities. *Foundations of physics*, 22(7):879–948, 1992.
11. Anand Natarajan and Thomas Vidick. Low-Degree Testing for Quantum States, and a Quantum Entangled Games PCP for QMA. In *Proceedings of the 59th IEEE Symposium on Foundations of Computer Science*, FOCS '18, pages 731–742, 2018.
12. Sandu Popescu and Daniel Rohrlich. Quantum nonlocality as an axiom. *Foundations of Physics*, 24(3):379–385, 1994.
13. Prasad Raghavendra and David Steurer. Integrality Gaps for Strong SDP Relaxations of UNIQUE GAMES. In *Proceedings of the 50th IEEE Symposium on Foundations of Computer Science*, FOCS '09, pages 575–585, 2009. Full version at `http://people.eecs.berkeley.edu/~prasad/Files/cspgaps.pdf`.

14 Peter Rastall. Locality, Bell's theorem, and quantum mechanics. *Foundations of Physics*, 15(9):963–972, 1985.

15 Ronitt Rubinfeld and Madhu Sudan. Robust Characterizations of Polynomials with Applications to Program Testing. *SIAM Journal on Computing*, 25(2):252–271, 1996.

16 Hanif D. Sherali and Warren P. Adams. A Hierarchy of Relaxations between the Continuous and Convex Hull Representations for Zero-One Programming Problems. *SIAM Journal on Discrete Mathematics*, 3(3):411–430, 1990.

Local Access to Huge Random Objects Through Partial Sampling

Amartya Shankha Biswas
CSAIL, MIT, Cambridge, MA, USA
asbiswas@mit.edu

Ronitt Rubinfeld
CSAIL, MIT, Cambridge, MA, USA
ronitt@csail.mit.edu

Anak Yodpinyanee
CSAIL, MIT, Cambridge, MA, USA
anak@csail.mit.edu

Abstract

Consider an algorithm performing a computation on a *huge random object* (for example a random graph or a "long" random walk). Is it necessary to generate the entire object prior to the computation, or is it possible to provide query access to the object and sample it incrementally "on-the-fly" (as requested by the algorithm)? Such an *implementation* should emulate the random object by answering queries in a manner consistent with an instance of the random object sampled from the true distribution (or close to it). This paradigm is useful when the algorithm is sub-linear and thus, sampling the entire object up front would ruin its efficiency.

Our first set of results focus on undirected graphs with independent edge probabilities, i.e. each edge is chosen as an independent Bernoulli random variable. We provide a general implementation for this model under certain assumptions. Then, we use this to obtain the first efficient local implementations for the Erdös-Rényi $G(n, p)$ model for *all* values of p, and the Stochastic Block model. As in previous local-access implementations for random graphs, we support VERTEX-PAIR and NEXT-NEIGHBOR queries. In addition, we introduce a new RANDOM-NEIGHBOR query. Next, we give the first local-access implementation for ALL-NEIGHBORS queries in the (sparse and directed) Kleinberg's Small-World model. Our implementations require no pre-processing time, and answer each query using $\mathcal{O}(\text{poly}(\log n))$ time, random bits, and additional space.

Next, we show how to implement random Catalan objects, specifically focusing on Dyck paths (balanced random walks on the integer line that are always non-negative). Here, we support HEIGHT queries to find the location of the walk, and FIRST-RETURN queries to find the time when the walk returns to a specified location. This in turn can be used to implement NEXT-NEIGHBOR queries on random rooted ordered trees, and MATCHING-BRACKET queries on random well bracketed expressions (the Dyck language).

Finally, we introduce two features to define a new model that: (1) allows multiple independent (and even simultaneous) instantiations of the same implementation, to be consistent with each other without the need for communication, (2) allows us to generate a richer class of random objects that do not have a succinct description. Specifically, we study uniformly random *valid* q-colorings of an input graph G with maximum degree Δ. This is in contrast to prior work in the area, where the relevant random objects are defined as a distribution with $\mathcal{O}(1)$ parameters (for example, n and p in the $G(n, p)$ model). The distribution over valid colorings is instead specified via a "huge" input (the underlying graph G), that is far too large to be read by a sub-linear time algorithm. Instead, our implementation accesses G through local neighborhood probes, and is able to answer queries to the color of any given vertex in sub-linear time for $q \geq 9\Delta$, in a manner that is consistent with a specific random valid coloring of G. Furthermore, the implementation is memory-less, and can maintain consistency with non-communicating copies of itself.

2012 ACM Subject Classification Theory of computation → Generating random combinatorial structures; Theory of computation → Streaming, sublinear and near linear time algorithms

Keywords and phrases sublinear time algorithms, random generation, local computation

11th Innovations in Theoretical Computer Science Conference (ITCS 2020).
Editor: Thomas Vidick; Article No. 27; pp. 27:1–27:65

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.27

Funding *Amartya Shankha Biswas*: MIT Presidential Fellowship
Ronitt Rubinfeld: NSF grants CCF-1650733, CCF-1733808, IIS-1741137 and CCF-1740751
Anak Yodpinyanee: NSF grants CCF-1650733, CCF-1733808, IIS-1741137 and DPST scholarship, Royal Thai Government

1 Introduction

Consider an algorithm performing a computation on a *huge random object* (for example a huge random graph or a "long" random walk). Is it necessary to generate the entire object prior to the computation, or is it possible to provide local query access to the object and generate it incrementally "on-the-fly" (as requested by the algorithm)? Such an *implementation* would ideally emulate the random object by answering appropriate queries, in a manner that is consistent with a specific instance of the random object, sampled from the true distribution. This paradigm is useful when we wish to simulate a sub-linear algorithm on a random object, since sampling the entire object up front would ruin its efficiency.

In this work, we focus on generating huge random objects in a number of new settings, including basic random graph models that were not previously considered, Catalan objects, and random colorings of graphs. One emerging theme that we develop further is to provide access to random objects through more complex yet natural queries. For example, consider an implementation for Erdös-Rényi $G(n, p)$ random graphs, where the simplest query $\textsc{Vertex-Pair}(u, v)$ would ask about the existence of edge (u, v), which can be answered trivially by flipping a coin with bias p (thus revealing a single entry in the adjacency matrix). However, many applications involving non-dense graphs would benefit from *adjacency list* access, which we provide through $\textsc{Next-Neighbor}$ and $\textsc{Random-Neighbor}$ queries[1]

The problem of sampling partial information about huge random objects was pioneered in [26, 24, 25], through the implementation of *indistinguishable* pseudo-random objects of exponential size. Further work in [41, 18] considers the implementation of different random graph models. [18] introduced the study of $\textsc{Next-Neighbor}$ queries which provide efficient access to the adjacency list representation. In addition to supporting $\textsc{Vertex-Pair}$ and $\textsc{Next-Neighbor}$, we also introduce and implement a new $\textsc{Random-Neighbor}$ query for undirected graphs with independent edge probabilities.

Finally, we define a new model that allows us to generate a richer class of random objects that do not have a succinct description. Specifically, we study uniformly random *valid* q-colorings of an input graph G with max degree Δ. This is in contrast to prior work in the area, where random objects are defined as a distribution with $\mathcal{O}(1)$ parameters (for example, n and p in the $G(n, p)$ model). The distribution over valid colorings is instead specified via a "huge" input (the underlying graph G), that is too large to be read by a sub-linear time algorithm. Instead, our implementation accesses G using local neighborhood probes.

This new model can be compared to *Local Computation Algorithms*, which also implement query access to a consistent valid solution, and read their input using local probes. Inspired by this connection, we extend our model to support *memory-less* implementations. This allows multiple independent (possibly simultaneous) instantiations to agree on the same

[1] $\textsc{Vertex-Pair}(u, v)$ returns whether u and v are adjacent, $\textsc{Next-Neighbor}(v)$ returns a new neighbor of v each time it is invoked (until none is left), and $\textsc{Random-Neighbor}(v)$ returns a uniform random neighbor of v (if v is not isolated).

random object, without any communication. We show how to implement local access to color of any given node in a random coloring, using sub-linear resources. Unlike LCAs which can generate an *arbitrary* valid solution, our model requires a uniformly random one.

Random Graphs (Section 3 and Section E)

Random graphs are one of the most well studied types of random object. We consider the problem of implementing local access to a number of fundamental random graph models, through natural queries, including VERTEX-PAIR, NEXT-NEIGHBOR, and a newly introduced RANDOM-NEIGHBOR query[1], using polylogarithmic resources per query. Our results on random graphs are summarized in Table 1.

Undirected Random Graphs with Independent Edge Probabilities. We implement the aforementioned queries for the generic class of *undirected graphs* with *independent edge probabilities* $\{p_{uv}\}_{u,v\in V}$, where p_{uv} denotes the probability that there is an edge between u and v. Under reasonable assumptions on the ability to compute certain values pertaining to consecutive edge probabilities, our implementations support VERTEX-PAIR, NEXT-NEIGHBOR, and RANDOM-NEIGHBOR queries[1], using $\mathcal{O}(\text{poly}(\log n))$ time, space, and random bits. Note that in this setting, VERTEX-PAIR queries are trivial by themselves, since the existence of an edge depends on an independent random variable. However, maintaining all three types of queries simultaneously is much harder. As in [18] (and unlike many of the implementations presented in [24, 26]), our techniques allow unlimited queries, and the generated random objects are sampled from the true distribution (rather than just being indistinguishable). In particular, our construction yields local-access implementations for the Erdös-Rényi $G(n,p)$ model (for *all* values of p), and the Stochastic Block model with random community assignment.

Table 1 (Local access implementations for random graphs): All the implementations in this table use polylogarithimc time, additional space and random bits per query. A ✗ in the ALL-NEIGHBORS column indicates that the graphs in this model may have un-bounded degree, and it is therefore impossible to sample ALL-NEIGHBORS efficiently. A ?? entry indicates a sampling problem with no known efficient solution.

Model	VERTEX-PAIR	NEXT-NEIGHBOR	RANDOM-NEIGHBOR	ALL-NEIGHBORS
$G(n,p)$ with $p = \frac{\log^{\mathcal{O}(1)} n}{n}$	[41]	[41]	[41]	[41]
$G(n,p)$ for arbitrary p	This paper	This paper	This paper	✗
Stochastic Block Model with $\log^{\mathcal{O}(1)} n$ communities	This paper	This paper	This paper	✗
BA Preferential Attachment	[18]	[18]	??	✗
Small world model	This paper	This paper	This paper	This paper
Random ordered rooted trees	This paper	This paper	This paper	This paper

While VERTEX-PAIR and NEXT-NEIGHBOR queries have been considered in prior work [18, 24, 41], we provide the first implementations of these in non-sparse random graph models. Prior results for implementing queries to $G(n,p)$ focused on the sparse case where $p = \log^{\mathcal{O}(1)} n/n$ [41]. The dense case where $p = \Theta(1)$ is also relatively simple because most of the adjacency matrix is filled, and neighbor queries can be answered by performing $\Theta(1)$ VERTEX-PAIR queries until an edge is found. The case of general p is more involved, and

was not considered previously. For example, when $p = 1/\sqrt{n}$, each vertex has high degree $\mathcal{O}(\sqrt{n})$ but most of the adjacency matrix is empty, thus making it difficult to generate a neighbor efficiently. NEXT-NEIGHBOR queries were introduced in [18] in order to access the neighborhoods of vertices in non-sparse graphs in *lexicographic order*. This query however, does not allow us to efficiently explore graphs in some natural ways, such as via random walks, since the initially returned neighbors are biased by the lexicographic ordering. We address this deficiency by introducing a new RANDOM-NEIGHBOR query, which would be useful, for instance, in sub-linear algorithms that employ random walk processes. We provide the first implementation of all three queries in *non-sparse graphs* as follows:

▶ **Theorem 1.** *Given a random graph model defined by the probability matrix $\{p_{uv}\}_{u,v\in[n]}$, and assuming that we can compute the the quantities $\prod_{u=a}^{b}(1-p_{vu})$ and $\sum_{u=a}^{b} p_{vu}$ in $\mathcal{O}(\log^{\mathcal{O}(1)} n)$ time, there exists an implementation for this model that supports local access through* RANDOM-NEIGHBOR*,* VERTEX-PAIR*, and* NEXT-NEIGHBOR *queries using $\mathcal{O}(\log^{\mathcal{O}(1)} n)$ running time, additional space, and random bits per query.*

We show that these assumptions can be realized in Erdös-Rényi random graphs and the Stochastic Block Model. SBM presents additional challenges for assigning community labels to vertices (Section 2.1.1).

▶ **Corollary 2.** *There exists an algorithm that implements local access to a Erdös-Rényi $G(n,p)$ random graph, through* VERTEX-PAIR*,* NEXT-NEIGHBOR*, and* RANDOM-NEIGHBOR *queries, while using $\mathcal{O}(\log^3 n)$ time, $\mathcal{O}(\log^3 n)$ random bits, and $\mathcal{O}(\log^2 n)$ additional space per query with high probability.*

▶ **Corollary 3.** *There exists an algorithm that implements local access to a random graph from the n-vertex Stochastic Block Model with r randomly-assigned communities, through* VERTEX-PAIR*,* NEXT-NEIGHBOR*, and* RANDOM-NEIGHBOR *queries, using $O(r\,\mathrm{poly}(\log n))$ time, random bits, and space per query w.h.p.*

We remark that while we are able to generate Erdös-Rényi random graphs on-the-fly supporting all three types of queries, our construction still only requires $\widetilde{\mathcal{O}}(m+n)$ time and space to generate a complete $G(n,p)$ graph, which is optimal up to logarithmic factors.

▶ **Corollary 4.** *The final algorithm in Section 3 can generate a complete random graph from the Erdös-Rényi $G(n,p)$ model using overall $\widetilde{\mathcal{O}}(n+m)$ time, random bits and space, which is $\widetilde{\mathcal{O}}(pn^2)$ in expectation. This is optimal up to $\mathcal{O}(\mathrm{poly}(\log n))$ factors.*

Directed Random Graphs – The Small World Model (Section E). We consider local-access implementations for directed graphs through Kleinberg's Small World model, where the probabilities are based on distances in a 2-dimensional grid. This model was introduced in [31] to capture the geographical structure of real world networks, in addition to reproducing observed properties of short routing paths and low diameter. We implement ALL-NEIGHBORS queries for the Small World model, using $\mathcal{O}(\mathrm{poly}(\log n))$ time, space and random bits. Since such graphs are sparse, the other queries follow directly.

▶ **Theorem 5.** *There exists an algorithm that implements* ALL-NEIGHBORS *queries for a random graph from Kleinberg's Small World model, where probability of including each directed non-grid edge (u,v) in the graph is $c/(\mathtt{DIST}(u,v))^2$, where the range of allowed values of c is $\log^{\pm\mathcal{O}(1)} n$ and* DIST *denotes the Manhattan distance, using $O(\log^2 n)$ time and random words per query with high probability.*

Catalan Objects (Section 4)

Catalan objects capture a well studied combinatorial property and admit numerous interpretations that include well bracketed expressions, rooted trees and binary trees, Dyck languages etc. In this paper, we focus on the Dyck path interpretation and implement local access to a uniformly random instance of a Dyck path. Since Dyck paths have natural bijections to other Catalan objects, we can use our Dyck Path implementation to obtain implementations of these random Catalan objects.

A Dyck path is defined as a sequence of n upward $(+1)$ steps, and n downward (-1) steps, with the added constraint that the *sum of any prefix of the sequence is non-negative*. A *random* Dyck path may be viewed as a constrained one dimensional balanced random walk. A natural query on Dyck paths is $\textsc{Height}(t)$ which returns the position of the walk after t steps (equivalently, sum of the first t sequence elements). $\textsc{Height}$ queries correspond to $\textsc{Depth}$ queries on rooted trees and bracketed expressions (Section 4.1).

We also introduce and support $\textsc{First-Return}$ queries, where $\textsc{First-Return}(t)$ returns the first time when the random walk returns to the same *height* as it was at time t, as long as $\textsc{Height}(t+1) = \textsc{Height}(t)+1$ (Section 4.1 presents the rationale for this definition). This allows us to support more involved queries that are widely used; for example, $\textsc{First-Return}$ queries are equivalent to finding the next child of a node in a random rooted tree, and also to finding a matching closing bracket in random bracketed expressions.

$\textsc{Height}$ queries for unconstrained random walks follow trivially from the implementation of interval summable functions in [26, 23]. However, the added non-negativity constraint introduces intricate non-local dependencies on the distribution of positions. We show how to overcome these challenges, and support both queries using $\mathcal{O}(\text{poly}(\log n))$ resources.

▶ **Theorem 6.** *There is an algorithm using $\mathcal{O}(\log^{\mathcal{O}(1)} n)$ resources per query that provides access to a uniformly random Dyck path of length $2n$ by implementing the following queries:*

- $\textsc{Height}(t)$ *returns the position of the walk after t steps.*
- $\textsc{First-Return}(t)$*: If* $\textsc{Height}(t+1) > \textsc{Height}(t)$*, then* $\textsc{First-Return}(t)$ *returns the smallest index t', such that $t' > t$ and* $\textsc{Height}(t') = \textsc{Height}(t)$ *(i.e. the first time the Dyck path returns to the same height). Otherwise, if* $\textsc{Height}(t+1) < \textsc{Height}(t)$*, then* $\textsc{First-Return}(t)$ *is not defined.*

Random Coloring of Graphs – A New Model (Section 5)

So far, all the results in this area have focused on random objects with a small description size; for instance, the $G(n,p)$ model is described with only two parameters n and p. We introduce a new model for implementing random objects with *huge input description*; that is, the distribution is specified as a uniformly random solution to a huge combinatorial problem. The challenge is that now our algorithms cannot read the entire description in sub-linear time.

In this model, we implement query access to random q-colorings of a given huge graph G with maximum degree Δ. A random coloring is generated by proposing $\mathcal{O}(n \log n)$ color updates and accepting the ones that do not create a conflict (Glauber dynamics). This is an inherently sequential process with the acceptance of a particular proposal depending on all preceding neighboring proposals. Moreover, unlike the previously considered random objects, this one has no succinct representation, and we can only uncover the proper distribution by probing the underlying graph (in the manner of *local computation algorithms* [48, 5]). Unlike LCAs which can return an *arbitrary* valid solution, we also have to make sure that we return a solution from the correct distribution. We are able to construct an efficient oracle that returns the final color of a vertex using only a sub-linear number of probes when $q \geq 9\Delta$.

This implementation also has the feature that multiple independent instances of the algorithm having access to the same random bits, will respond to queries in a manner consistent with each other; they will generate exactly the same coloring, regardless of the queries asked. Since these implementations are memory-less, the resulting coloring is *oblivious* to the order of queries, and only depends on common random bits,

▶ **Theorem 7.** *Given neighborhood query access to a graph G with n nodes, maximum degree Δ, and $q = 2\alpha\Delta \geq 9\Delta$ colors, we can generate the color of any given node from a distribution of color assignments that is ϵ-close (in L_1 distance) to the uniform distribution over all valid colorings of G, in a consistent manner, using only $\mathcal{O}((n/\epsilon)^{3.06/\alpha}\Delta\log(n/\epsilon))$ time, probes, and public random bits per query.*

1.1 Related Work

The problem of computing local information of huge random objects was pioneered in [24, 26]. Further work of [41] considers the implementation of sparse $G(n,p)$ graphs from the Erdös-Rényi model [17], with $p = O(\text{poly}(\log n)/n)$, by implementing ALL-NEIGHBORS queries. While these implementations use polylogarithmic resources over their entire execution, they generate graphs that are only guaranteed to *appear random* to algorithms that inspect a *limited portion* of the generated graph.

In [18], the authors construct an oracle for the implementation of recursive trees, and BA preferential attachment graphs. Unlike prior work, their implementation allows for an arbitrary number of queries. Although the graphs in this model are generated via a sequential process, the oracle is able to locally generate arbitrary portions of it and answer queries in polylogarithmic time. Though preferential attachment graphs are sparse, they contain vertices of high degree, thus [18] provides access to the adjacency list through NEXT-NEIGHBOR queries. For additional related work, see Section G.

1.2 Applications

One motivation for implementing local access to huge random objects is so that we can simulate sub-linear time algorithms on them. The standard paradigm of generating the entire (input) object a priori is wasteful, because sub-linear algorithms only inspect a small fraction of the generated input. For example, the greedy routing algorithm on Kleinberg's small world networks [31] only uses $\mathcal{O}(\log^2 n)$ probes to the underlying network. Using our implementation, one can execute this algorithm on a random small world instance in $\mathcal{O}(\text{poly}(\log n))$ time, without incurring the $\mathcal{O}(n)$ prior-sampling overhead, by generating only those parts of the graph that are accessed by the routing algorithm.

Local access implementations may also be used to design parallel generators for random objects. The model in Definition 10 is particularly suited to parallelization; different processors/machines can generate parts of the random object independently, using a *read-only* shared memory containing public random bits.

2 Model and Overview of our Techniques

We begin by formalizing our model of *local-access implementations*, inspired by [18], but we add some aspects that were not addressed in earlier models. First, we define families of random objects.

▶ **Definition 8.** *A **random object family** maps a description* Π *to a distribution* X^Π *over the set* $\mathbb{X}^\Pi$*.*

For example, the *family* of Erdös-Rényi graphs maps $\Pi = (n, p)$ to a distribution over $\mathbb{X}^\Pi$, which is the set of all possible n-vertex graphs, where the probability assigned to any graph containing m edges in the distribution X^Π is exactly $p^m \cdot (1-p)^{\binom{n}{2}-m}$.

▶ **Definition 9.** *Given a **random object family** $\{(\mathsf{X}^\Pi, \mathbb{X}^\Pi)\}$ parameterized by* Π*, a* local access implementation *of a family of query functions $\langle F_1, F_2, \cdots \rangle$ where $F_i : \mathbb{X}^\Pi \to \{0,1\}$, provides an oracle $\mathcal{M}$ with an internal state for storing a partially generated random object. Given a description Π and a query F_i, the oracle returns the value $\mathcal{M}(\Pi, F_i)$, and updates its internal state, while satisfying the following:*

- ***Consistency:** There must be a single $X \in \mathbb{X}^\Pi$, such that for all queries F_i presented to the oracle, the returned value $\mathcal{M}(\Pi, F_i)$ equals the true value $F_i(X)$.*
- ***Distribution equivalence:** The random object $X \in \mathbb{X}^\Pi$ consistent with the responses $\{F_i(X)\}$ must be sampled from some distribution $\hat{\mathsf{X}}^\Pi$ that is ϵ-close to the desired distribution X^Π in L_1-distance. In this work, we focus on supporting $\epsilon = n^{-c}$ for any desired constant $c > 0$.*
- ***Performance:** The computation time, and random bits required to answer a single query must be sub-linear in $|X|$ with high probability, without any initialization overhead.*

In particular, we allow queries to be made adversarially and non-deterministically. The adversary has full knowledge of the algorithm's behavior and its past random bits.

For example, in the $G(n,p)$ family with description $\Pi = (n,p)$, we can define `Vertex-Pair` query functions $\{F_{(u,v)}\}_{u,v\in[n]}$. So, given a graph $G \in \mathbb{X}^\Pi$, the query $F_{(u,v)}(G) = 1$ if and only if $(u,v) \in G$.

In prior work [18, 26, 41] as well as some of our results, the input description Π is of small size (typically $\mathcal{O}(\log n)$), and the oracle $\mathcal{M}$ can read all of Π (for example, $\Pi = (n,p)$ in $G(n,p)$).

Distributions with Huge Description Size. We initiate the study of **random object families** where the description Π is too large to be read by a sub-linear time algorithm. In this setting, the oracle from Definition 9 cannot read the entire input Π, and instead accesses it through local probes. For instance, consider the **random object family** that maps a graph G to the uniform distribution over valid colorings of G. Here, the description Π includes the entire graph G, which is too large to be read by a sub-linear time algorithm. In this case, the oracle can query the underlying graph using neighborhood probes. The number of such probes used to answer a single query must be sub-linear in the input size.

Supporting Independent Query Oracles: Memory-less Implementations

The model in Definition 9 only supports sequential queries, since the response to a future query may depend on the changes in internal state caused by past queries. In some applications, we may want to have multiple independent query oracles whose responses are all consistent with each other. One way to achieve this is to restrict our attention to *memory-less* implementations; ones without any internal state. An important implication of being memory-less is that the responses to each query is oblivious to the order of queries being asked. In fact, the lack of internal state implies that independent implementations that use the same random bits and the same input description must respond to queries in the same way. Thus, instead of using the internal state to maintain consistency, memory-less implementations are given access to the same public random oracle.

For the problem of sampling a random graph coloring, we present an implementation that is memory-less and also accesses the input description through local probes, as elaborated in the following model:

▶ **Definition 10.** *Given a* ***random object family*** $\{(\mathsf{X}^\Pi, \mathbb{X}^\Pi)\}$ *parameterized by input* Π*, a* local access implementation *of a family of query functions* $\langle F_1, F_2, \cdots \rangle$*, provides an oracle* $\mathcal{M}$ *with the following properties.* $\mathcal{M}$ *has query access to the input description* Π*, and a tape of public random bits* $\mathbf{R}$*. Upon being queried with* Π *and* F_i*, the oracle uses sub-linear resources to return the value* $\mathcal{M}(\Pi, \mathbf{R}, F_i)$*, which must equal* $F_i(X)$ *for a specific* $X \in \mathbb{X}^\Pi$*, where the choice of* X *depends only on* $\mathbf{R}$*, and the distribution of* X *(over* $\mathbf{R}$*) is* $\frac{1}{n^c}$*-close to the distribution over* $\mathbb{X}^\Pi$*, for any given constant* c*. Thus, different instances of* $\mathcal{M}$ *with the same description* Π *and the same random bits* $\mathbf{R}$*, must agree on the choice of* X *that is consistent with all answered queries regardless of the order and content of queries that were actually asked.*

We can contrast Definition 10 with the one for *Local Computation Algorithms* [48, 5] which also allow query access to *some* valid solution by reading the input through local probes. The additional challenge in our setting is that we also have to make sure that we return a uniformly random solution, rather than an arbitrary one. We also note that the memory-less property may be achieved for small description size **random object families**. For instance, our implementation for the directed small world model admits such a memory-less implementation using public random bits.

2.1 Undirected Graphs

We consider the generic class of *undirected graphs* with *independent edge probabilities* $\{p_{uv}\}_{u,v \in V}$, (where p_{uv} denotes the probability that there is an edge between u and v), from which the results can be applied to Erdös-Rényi random graphs and the Stochastic Block Model. Throughout, we identify our vertices via their unique IDs from 1 to n, namely $V = [n]$. In this model, `Vertex-Pair` queries by themselves can be implemented trivially, since the existence of any edge (u, v) is an independent Bernoulli random variable, but it becomes harder to maintain consistency when implementing them in conjunction with the other queries. Inspired by [18], we provide an implementation of `next-neighbor` queries, which return the neighbors of any given vertex one by one in lexicographic order. Finally, we introduce a new query: `Random-Neighbor` that returns a uniformly random neighbor of any given vertex. This would be useful for any algorithm that performs random walks. `Random-Neighbor` queries in non-sparse graphs present particularly interesting challenges that are outlined below.

`Next-Neighbor` Queries

In Erdös-Rényi graphs, the (lexicographically) next neighbor of a vertex can be recovered by generating consecutive entries of the adjacency matrix until a neighbor is found, which takes roughly $\Omega(1/p_{uv})$ time. For small edge probabilities $p_{uv} = o(1)$, this implementation is inefficient, and we show how to improve the runtime to $\mathcal{O}(\text{poly}(\log n))$. Our main technique is to sample the number of "non-neighbors" preceding the next neighbor. To do this, we assume that we can estimate the "skip" probabilities $F(v, a, b) = \prod_{u=a}^{b}(1 - p_{vu})$, where $F(v, a, b)$ is the probability that v has no neighbors in the range $[a, b]$. We later show how to compute this quantity efficiently for $G(n, p)$ and the Stochastic Block Model.

This strategy of *skip-sampling* is also used in [18]. However, in our work, the main difficulty arises from the fact that our graph is undirected, and thus we must "inform" all (potentially $\Theta(n)$) non-neighbors once we decide on the query vertex's next neighbor. Concretely, if u' is sampled as the next neighbor of v after its previous neighbor u, we must maintain consistency in subsequent steps by ensuring that none of the vertices in the range (u, u') return v as a neighbor. This update will become even more complicated as we handle Random-Neighbor queries, where we may generate non-neighbors at random locations.

In Section 3.1, we present a randomized implementation (Algorithm 1) that supports Next-Neighbor queries efficiently, but has a complicated performance analysis. We remark that this approach may be extended to support Vertex-Pair queries (Section A) with superior performance (if we do not support Random-Neighbor queries), and also to provide deterministic resource usage guarantee (Section C).

Random-Neighbor Queries

We implement Random-Neighbor queries (Section 3.2) using $\text{poly}(\log n)$ resources. The ability to do so is surprising since: (1) Sampling the degree of vertex, may not be viable for *sub-linear* implementations, because this quantity alone imposes dependence on the existence of *all* of its potential incident edges and consequently on the rest of the graph (since it is undirected). Thus, our implementation needs to return a random neighbor, with probability reciprocal to the query vertex's degree, without resorting to *determining* its degree. (2) Even without committing to the degrees, answers to Random-Neighbor queries affect the conditional probabilities of the remaining adjacencies in a global and non-trivial manner.[2]

We formulate an approach which samples many consecutive edges simultaneously, in such a way that the conditional probabilities of the unsampled edges remain independent and "well-behaved" during subsequent queries. For each vertex v, we divide the potential neighbors of v into consecutive ranges $\{B_v^{(i)}\}$ called blocks, so that each $B_v^{(i)}$ contains $\Theta(1)$ neighbors in expectation (i.e. $\sum_{u \in B_i} p_{vu} = \Theta(1)$). The subroutine of Next-Neighbor is applied to sample the neighbors within a block in expected $\widetilde{\mathcal{O}}(1)$ time. We can now obtain a neighbor of v by picking a random neighbor from a random block, but this introduces a bias because all blocks may not have the same number of neighbors. We remove this bias by rejecting samples from block $B_v^{(i)}$ with probability proportional to the number of neighbors in $B_v^{(i)}$.

2.1.1 Applications to Random Graph Models

We now consider the application of our construction above to actual random graph models, where we must realize the assumption that $\prod_{u=a}^{b}(1 - p_{vu})$ and $\sum_{u=a}^{b} p_{vu}$ can be computed efficiently. For the Erdös-Rényi $G(n, p)$ model, these quantities have simple closed-form expressions. Thus, we obtain implementations of Vertex-Pair, Next-Neighbor, and Random-Neighbor queries, using polylogarithmic resources per query, for *arbitrary* values of p, We remark that, while $\Omega(n + m) = \Omega(pn^2)$ time and space is clearly necessary to generate and represent a full random graph, our implementation supports local-access via all three types of queries, and yet can generate a full graph in $\widetilde{O}(n + m)$ time and space (Corollary 4).

[2] Consider a $G(n, p)$ graph with small p, say $p = 1/\sqrt{n}$, such that vertices will have $\tilde{\mathcal{O}}(\sqrt{n})$ neighbors with high probability. After $\tilde{\mathcal{O}}(\sqrt{n})$ Random-Neighbor queries, we will have uncovered all the neighbors (w.h.p.), so that the conditional probability of the remaining $\Theta(n)$ edges should now be close to zero.

We also generalize our construction to implement the Stochastic Block Model. In this model, the vertex set is partitioned into r *communities* $\{C_1, \ldots, C_r\}$. The probability that an edge exists between $u \in C_i$ and $v \in C_j$ is p_{ij}. A naive solution would be to simply assign communities to contiguous (by index) blocks of vertices, which would easily allow us to calculate the relevant sums/products of probabilities on continuous ranges of indices, with some additional case analysis to check when we are at a community boundary. However, this setup is unrealistic, and not particularly useful in the context of the Stochastic Block model. As communities in the observed data are generally unknown a priori, and significant research has been devoted to designing efficient algorithms for community detection and recovery, these studies generally consider the *random community assignment* condition for the purpose of designing and analyzing algorithms [40]. Thus, we construct implementations where the community assignments are sampled from some given distribution R, or from a collection of specified community sizes $\langle |C_1|, \ldots, |C_r| \rangle$. The main difficulty here is to obtain a uniformly sampled assignment of vertices to communities on-the-fly.

Since the probabilities of potential edges now depend on the communities of their endpoints, we can't obtain closed form expressions for the relevant sums and products of probabilities. However, we observe that it suffices to efficiently count the number of vertices of each community in any range of contiguous vertex indices. We then design a data structure extending a construction of [26], which maintains these counts for ranges of vertices, and determines the partition of their counts only on an as-needed basis. This extension results in an efficient technique to sample counts from the *multivariate hypergeometric distribution* (Section D) which may be of independent interest. For r communities, this yields an implementation with $\mathcal{O}(r \cdot \mathrm{poly}(\log n))$ overhead in required resources for each operation.

2.2 Directed Graphs

Lastly, we consider Kleinberg's Small World model ([31, 37]) in Section E. While small world models are proposed to capture properties of observed data such as small shortest-path distances and large clustering coefficients [55], this important special case of Kleinberg's model, defined on two-dimensional grids, demonstrates the underlying geographical structures of networks.

In this model, each vertex is identified via its 2D coordinate $v = (v_x, v_y) \in [\sqrt{n}]^2$. Defining the Manhattan distance as $\textsc{dist}(u, v) = |u_x - v_x| + |u_y - v_y|$, the probability that each directed edge (u, v) exists is $c/(\textsc{dist}(u, v))^2$. A common choice for c is given by normalizing the distribution such that the expected out-degree of each vertex is 1 ($c = \Theta(1/\log n)$). We can also support a range of values of $c = \log^{\pm\Theta(1)} n$. Since the degree of each vertex in this model is $\mathcal{O}(\log n)$ with high probability, we implement `All-Neighbor` queries, which in turn can emulate `Vertex-Pair`, `Next-Neighbor` and `Random-Neighbor` queries. In contrast to our previous cases, this model imposes an underlying two-dimensional structure of the vertex set, which governs the distance function as well as complicates the individual edge probabilities.

We implement `All-Neighbors` queries in the small world model by listing all neighbors from closest to furthest away from the queried vertex, using $\mathrm{poly}(\log n)$ resources per query. The main challenge is to sample for the next closest neighbor, when the probabilities are a function of the Manhattan distance on the lattice. Rather than sampling for a neighbor directly, we partition the nodes based on their distance from v (there are $\Theta(d)$ vertices at distance d). We first choose the next smallest distance partition with a neighbor using rejection sampling techniques (Lemma 11) on the appropriate distribution. In the second step, we generate all the neighbors within that partition using *skip-sampling*.

2.3 Random Catalan Objects

Many important combinatorial objects can be interpreted as Catalan objects. One such interpretation is a Dyck path; a one dimensional random walk on the line with n up and n down steps, starting from the origin, with the constraint that the height is always non-negative. We implement $\textsc{Height}(t)$, which returns the position of the walk at time t, and $\textsc{First-Return}(t)$, which returns the first time when the random walk returns to the same position as it was at time t. These queries are natural for several types of Catalan objects. As noted previously, we can use standard bijections to translate the Dyck path query implementations into natural queries for *bracketed expressions* and *ordered rooted trees*. Specifically, `Height` values in Dyck paths are equivalent to *depth* in bracket expressions and trees. The `First-Return` queries are more involved, and are equivalent to finding the *matching bracket* in bracket expressions, and alternately to finding the *next child* of a node in an ordered rooted tree (see Section 4.1).

Over the course of the execution, our algorithm will determine the height of a random Dyck path at many different positions $\{x_1, x_2, \cdots, x_m\}$ (with $x_i < x_{i+1}$), both directly as a result of user given `Height` queries, and indirectly through recursive calls to `Height`. These positions divide the sequence into contiguous *intervals* $[x_i, x_{i+1}]$, where the height of the endpoints y_i, y_{i+1} have been determined, but none of the intermediate heights are known. The important observation is that the unknown section of the path within an *interval* is entirely determined by the positions and heights of the endpoints, and in particular is completely independent of all other *intervals*. So, each interval $[x_i, x_{i+1}]$ along with corresponding heights $\{y_i, y_{i+1}\}$, represents a generalized Dyck problem with U up steps, D down steps, and a constraint that the path never dips more than y_i units below the starting height.

`Height` Queries

General $\textsc{Height}(x)$ queries can then be answered by recursively halving the *interval* containing x, by repeatedly sampling the height at the midpoint, until the height of position x is sampled. We start by implementing a subroutine that given an *interval* $[x_i, x_{i+1}]$ of length $2B$, containing $2U$ up and $2D$ down steps, determines the number of up steps $U' = U + d$ assigned to the first half of the *interval* (we parameterize U' with d in order to make the analysis cleaner). Note that this is equivalent to answering the query $\textsc{Height}(x_i + B)$. This is done by sampling the parameter d from a distribution $\{p_d\}$ where $p_d \equiv S_{left}(d) \cdot S_{right}(d)/S_{total}$. Here, $S_{left}(d)$ (respectively $S_{right}(d)$) is the number of possible paths in the left (resp. right) half of the *interval* when $U + d$ up steps and $D - d$ down steps are assigned to the first half, and S_{total} is the number of possible paths in the original $2B$-interval.

The problem of determining the number of up steps in the first half of the *interval* was solved for the unconstrained case (where the sequence is just a random permutation of up and down steps) in [26]. Adding the non-negativity constraint introduces further difficulties as the distribution over d has a CDF that is difficult to compute. We construct a different distribution $\{q_d\}$ that approximates $\{p_d\}$ pointwise to a factor of $\mathcal{O}(\log n)$ and has an efficiently computable CDF. This allows us to sample from $\{q_d\}$ and leverage rejection sampling techniques (see Lemma 11 in Section 2.5) to obtain samples from $\{p_d\}$.

`First-Return` Queries

Note that $\textsc{First-Return}(x)$ is only of interest when the first step after position x is upwards (i.e. $\textsc{Height}(x+1) > \textsc{Height}(x)$), since this situation is important for complex queries to random bracketed expressions and random rooted trees. Section 4.1 details the rationale

for this definition based on bijections between these objects. FIRST-RETURN queries are challenging because we need to find the *interval* containing the first return to HEIGHT(x). Since there could be up to $\Theta(n)$ intervals, it is inefficient to iterate through all of them. To circumvent this problem, we allow each interval $[x_i, x_{i+1}]$ to sample and maintain its own boundary constraint k_i instead of using the global non-negativity constraint. This implies that the path within the interval $[x_i, x_{i+1}]$ never reaches the height $y_i - k_i$ or lower. Additionally, we maintain a crucial invariant that states that this boundary is achieved by the endpoint of lower height i.e. $\min(y_i, y_{i+1})$. If the invariant holds, we can find the interval containing FIRST-RETURN(x) by finding the smallest determined position $x_j > x$ whose sampled height $y_j \leq$ HEIGHT(x), and considering the interval $[x_{j-1}, x_j]$ preceding x_j. We use an interval tree to update and query for the known heights.

Unfortunately, every HEIGHT query creates new intervals by sub-dividing existing ones, potentially breaking the invariant. We re-establish the invariant for $[x_i, x_{i+1}]$ by generating a "mandatory boundary" h (a boundary constraint with the added restriction that some position within the interval *must* touch the boundary), and then sampling a position $x_{mid} \in [x_i, x_{i+1}]$ such that HEIGHT(x_{mid}) $= h$ (Figure 6). This creates new intervals $[x_i, x_{mid}]$ and $[x_{mid}, x_{i+1}]$, both of which have a boundary constraint at h.

The first step of sampling the *mandatory boundary* is performed by binary searching on the possible boundary locations using an appropriate CDF (Section 4.4.2). To find an intermediate position touching this boundary, we parameterize the position with d, and find the distribution $\{p_d\}$ associated with the various possibilities. Since we cannot directly sample from this complicated distribution, we define a *piecewise continuous* probability distribution $\hat{q}(\delta)$ such that $\hat{q}(\delta)$ approximates $p_{\lfloor \delta \rfloor}$ (Section 4.4.3). We then use this to define a discrete distribution $\{q_d\}$ where $q_d = \int_d^{d+1} \hat{q}(\delta)$, where we can efficiently compute the CDF of $\{q_d\}$ by integrating the piecewise continuous $\hat{q}(\delta)$. The challenge here is to construct an appropriate $\hat{q}(\delta)$ that only has $\mathcal{O}(\log^{\mathcal{O}(1)} n)$ continuous pieces. This allows us to again use the rejection sampling technique (Lemma 11) to indirectly obtain a sample from $\{p_d\}$.

2.4 Random Coloring of a Graph

Finally, we introduce a new model (Definition 10) for implementing huge random objects, where the distribution is specified as a uniformly random solution to a huge combinatorial problem. In this new setting, we will implement local query access to random q-colorings of a given huge graph G of size n with maximum degree Δ. Since the implementation has to run in sub-linear time, it is not possible to read the entire input G during a single query execution.

`Color` Queries

Given a graph G with maximum degree Δ, and the number of colors $q \geq 9\Delta$, we are able to construct an efficient implementation for COLOR(v) that returns the final color of v in a uniformly random q-coloring of G using only a sub-linear number of probes. Random colorings of a graph are sampled using $\mathcal{O}(n \log n)$ iterations of a Markov chain [22]. Each step of the chain proposes a random color update for a random vertex, and accepts the update if it does not create a conflict. This is an inherently sequential process, with the acceptance of a particular proposal depending on all preceding neighboring proposals.

To make the runtime analysis simpler, we define a modified version of Glauber Dynamics that proceeds in $\mathcal{O}(\log n)$ epochs. In each epoch, all of the n vertices propose a random color and update themselves if their proposals do not conflict with any of their neighbors. This Markov chain is a special case of the one presented in [20] for distributed graph coloring, and mixes in $\mathcal{O}(\log n)$ epochs when $q \geq 9\Delta$. In order to implement the query COLOR(v), it suffices

to implement $\textsc{Accepted}(v, t)$ that indicates whether the proposal for v was accepted in the t^{th} epoch. This depends on the prior colors of the potentially Δ neighbors of v. Determining the prior colors of all the neighbors w using recursive calls would result in Δ invocations of $\textsc{Accepted}(w, t-1)$ (at the preceding epoch $t-1$). Naively, this gives a bound of Δ^t on the number of invocations. We can prune the recursions by only considering neighbors who proposed the color c during *some* past epoch. This reduces the expected number of recursive calls to $t\Delta/q$, since there are $t\Delta$ potential proposals and each one is c with probability $1/q$. If q is larger than $t\Delta$, the number of recursive calls is less than 1, which gives a sub-linear bound on the total number of resulting invocations. Since the number of epochs t can be as large as $\Theta(\log n)$, this strategy will only work when $q = \Omega(\Delta \log n)$.

The improvement to $q = \Omega(\Delta)$ follows from the observation that for a neighbor w that proposed color c at epoch t', the recursive call corresponding to w can directly jump to epoch t'. If the conflicting color c was indeed accepted at that epoch, we then step *forwards* through epochs $t'+1, t'+2, \cdots$, to check whether c was overwritten by some future accepted proposal. This strategy dramatically reduces the recursive sub-problem size (given by the epoch number t'), and furthermore we show that we do not have to step through too many future epochs in order to check whether c was overwritten. This allows us to bound the runtime by $\widetilde{\mathcal{O}}\left(t\Delta(n/\epsilon)^{6.12\Delta/q}\right)$, where the overall coloring is sampled from a distribution that is ϵ-close to uniform (see Definition 10).

One requirement for our strategy is the ability to access a *valid initial coloring* (the initial state of the Markov Chain) through local probes, in addition to local probes to the underlying graph structure. This assumption can be removed by using a result of [10], that presents an LCA for $\Delta+1$ graph coloring using $\Delta^{\mathcal{O}(1)} \log n$ graph probes. Alternately, we can assign random initial colors to the vertices, which may result in an *invalid* final coloring. However, the Markov Chain will transform the initial invalid coloring into a valid one with high probability.

2.5 Basic Tools for Efficient Sampling

In this section, we describe the main techniques used to sample from a distribution $\{p_i\}_{i\in[n]}$, which differs based on the type of access to $\{p_i\}$ provided to the algorithm. If the algorithm is given cumulative distribution function (CDF) access to $\{p_i\}$, then it is well known that via $\mathcal{O}(\log n)$ CDF evaluations, one can sample according to a distribution that is at most n^{-c} far from $\{p_i\}$ in L_1 distance.

Sampling can be more challenging when when we can only access the probability distribution function (PDF) of $\{p_i\}$, The approach that we use in this work is to construct an auxiliary distribution $\{q_i\}$ such that: (1) $\{q_i\}$ has an efficiently computable CDF, and (2) q_i approximates p_i pointwise to within a polylogarithmic multiplicative factor for "most" of the support of $\{p_i\}$. The following Lemma inspired by [26] formalizes this concept.

▶ **Lemma 11.** *Let $\{p_i\}_{i\in[n]}$ and $\{q_i\}_{i\in[n]}$ be distributions on $[n]$ satisfying the following conditions:*

1. *There is a $\log^{\mathcal{O}(1)} n$ time algorithm to approximate p_i and q_i up to a multiplicative $\left(1 \pm \frac{1}{n^c}\right)$ factor.*
2. *We can generate an index i according to a distribution $\{\hat{q}_i\}$, where $\hat{q}_i$ is a $\left(1 \pm \frac{1}{n^c}\right)$ multiplicative approximation to q_i.*
3. *There exists a poly$(\log n)$-time recognizable set S such that*
   - $1 - \sum_{i\in S} p_i < \frac{1}{n^c}$
   - *For every $i \in S$, it holds that $p_i \le \log^{\mathcal{O}(1)} n \cdot q_i$*

Then, with high probability we can use only $\log^{\mathcal{O}(1)} n$ samples from $\{\hat{q}_i\}$ to generate an index i according to a distribution that is $\mathcal{O}\left(\frac{1}{n^c}\right)$-close to $\{p_i\}$ in L_1 distance.

Proof. We begin by setting an upper bound $U = \log^{\mathcal{O}(1)} n$ on p_i/q_i for all $i \in S$. The sampling proceeds in iterations, such that in each iteration we obtain an index i according to the distribution $\{\hat{q}_i\}$. If $i \notin S$, this index is returned with probability $\tilde{p}_i/U\tilde{q}_i$, where $\tilde{p}_i$ and $\tilde{q}_i$ are the $\left(1 \pm \frac{1}{n^c}\right)$ multiplicative approximations to p_i and q_i, Otherwise, we repeat this process until some output is returned.

The probability of returning index $i \in S$ in a particular iteration is $\hat{q}_i \cdot (\tilde{p}_i/U\tilde{q}_i)$, which is in turn a $\left(1 \pm \frac{1}{n^c}\right)$ multiplicative approximation to p_i/U. Hence, the probability of success in a single iteration is roughly $1/U$, and therefore we only need $\mathcal{O}(U \log n) = \log^{\mathcal{O}(1)} n$ iterations (and the same number of samples from $\{\hat{q}_i\}$) in order to succeed with high probability. The resulting distribution of indices approximates $\{p_i\}$ pointwise on the domain S, up to a factor of $\left(1 \pm \frac{1}{n^c}\right)$. Since the remainder of the domain contains at most $\frac{1}{n^c}$ probability mass, the output distribution is $\mathcal{O}\left(\frac{1}{n^c}\right)$ close to $\{p_i\}$ in L_1 distance. ◀

3 Local-Access Implementations for Random Undirected Graphs

In this section, we provide an efficient local access implementations for random undirected graphs when the probabilities $p_{uv} = \mathbb{P}[(u, v) \in E]$ are given, and we can efficiently approximate the following quantities: (1) the probability that there is no edge between a vertex u and a range of consecutive vertices $[a, b]$, namely $\prod_{u=a}^{b}(1 - p_{vu})$, and (2) the sum of the edge probabilities (i.e., the expected number of edges) between u and vertices from $[a, b]$, namely $\sum_{u=a}^{b} p_{vu}$. In Section 1.2, we provide subroutines for computing these values for the Erdös-Rényi model and the Stochastic Block model. We also begin by assuming perfect-precision arithmetic, which we relax in Section B.1.

First, consider the adjacency matrix $\mathbf{A}$ of G, where each entry $\mathbf{A}[u][v]$ can exist in three possible states: $\mathbf{A}[u][v] = 1$ or 0 if the algorithm has determined that $\{u, v\} \in E$ or $\{u, v\} \notin E$ respectively, and $\mathbf{A}[u][v] = \phi$ if whether $\{u, v\} \in E$ or not will be determined by future random choices (in fact, the marginal probability of $\mathbb{P}[(u, v) \in E]$ conditioned on all prior samples is still p_{uv}). Our implementation also maintains the vector **last** (used in the same sense as [18]), where $\mathbf{last}[v]$ records the neighbor of v returned in the last call NEXT-NEIGHBOR(v), or $\mathbf{last}[v] = 0$ if no such call has been invoked. All cells of $\mathbf{A}$ and **last** are initialized to ϕ and 0, respectively.

We use the Bernoulli random variable $X_{uv} \sim \mathsf{Bern}(p_{uv})$ when sampling the value of $\mathbf{A}[u][v] = \phi$. For the sake of analysis, we will frequently view our random process as if the *entire* table of random variables X_{uv} has been sampled *up-front*, and the algorithm simply "uncovers" these variables instead of making coin-flips. Thus, every cell $\mathbf{A}[u][v]$ is originally ϕ, but will eventually take the value X_{uv}.

Obstacles for maintaining A explicitly. Consider a naive implemention that fills out the cells of $\mathbf{A}$ one-by-one as required by each query; equivalently, we perform VERTEX-PAIR queries on successive vertices until a neighbor is found. There are two problems with this approach. Firstly, the algorithm only finds a neighbor, for a RANDOM-NEIGHBOR or NEXT-NEIGHBOR query, with probability p_{uv}: for $G(n, p)$ this requires $1/p$ iterations, which is already infeasible for $p = o(1/\text{poly}(\log n))$. Secondly, the algorithm may generate a large number of non-neighbors in the process, possibly in random or arbitrary locations.

3.1 `Next-Neighbor` Queries via Run-of-0's Sampling

We implement NEXT-NEIGHBOR(v) by sampling for the first index $u > \mathbf{last}[v]$ such that $X_{vu} = 1$, from a sequence of Bernoulli RVs $\{X_{v,u}\}_{u>\mathbf{last}[v]}$. To do so, we sample a consecutive "run" of 0's with probability $\prod_{u=\mathbf{last}[v]+1}^{u'}(1-p_{vu})$: this is the probability that there is no edge between a vertex v and any $u \in (\mathbf{last}[v], u']$, which can be computed efficiently by our assumption. The problem is that, some entries $\mathbf{A}[v][u]$'s in this run may have already been determined (to be 1 or 0) by queries NEXT-NEIGHBOR(u) for $u > \mathbf{last}[v]$. To mitigate this issue, we give a succinct data structure that determines the value of $\mathbf{A}[v][u]$ for $u > \mathbf{last}[v]$ and, more generally, captures the state $\mathbf{A}$, in Section 3.1.1. Using this data structure, we ensure that our sampled run does not skip over any 1. Next, for the sampled index u of the first occurrence of 1, we check against this data structure to see if $\mathbf{A}[v][u]$ is already assigned to 0, in which case we re-sample for a new candidate $u' > u$. Section 3.1.2 discusses the subtlety of this issue.

We note that we do not yet try to handle other types of queries here yet. We also do not formally bound the number of re-sampling iterations of this approach here, because the argument is not needed by our final algorithm. Yet, we remark that $O(\log n)$ iterations suffice with high probability, even if the queries are adversarial. This method can be extended to support VERTEX-PAIR queries (but unfortunately not RANDOM-NEIGHBOR queries). See Section A for full details.

3.1.1 Data structure

From the definition of X_{uv}, NEXT-NEIGHBOR(v) is given by $\min\{u > \mathbf{last}[v] : X_{vu} = 1\}$. Let $P_v = \{u : \mathbf{A}[v][u] = 1\}$ be the set of known neighbors of v, and $w_v = \min\{(P_v \cap (\mathbf{last}[v], n])\}$ be its first known neighbor not yet reported by a NEXT-NEIGHBOR(v) query, or equivalently, the next occurrence of 1 in v's row on $\mathbf{A}$ after $\mathbf{last}[v]$. If there is no known neighbor of v after $\mathbf{last}[v]$, we set $w_v = n + 1$. Consequently, $\mathbf{A}[v][u] \in \{\phi, 0\}$ for all $u \in (\mathbf{last}[v], w_v)$, so NEXT-NEIGHBOR($v$) is either the index u of the first occurrence of $X_{vu} = 1$ in this range, or w_v if no such index exists.

We keep track of $\mathbf{last}[v]$ in a dictionary, to avoid any initialization overhead. Each P_v is maintained as an ordered set, which is also instantiated when it becomes non-empty. When NEXT-NEIGHBOR(v) returns u, we add v to P_u and u to P_v. We do not attempt to maintain $\mathbf{A}$ explicitly, as updating it requires replacing up to $\Theta(n)$ ϕ's to 0's for a single NEXT-NEIGHBOR query in the worst case. Instead, we argue that $\mathbf{last}$ and P_v's provide a succinct representation of $\mathbf{A}$ via the following observation.

▶ **Lemma 12.** *The data structures* $\mathbf{last}$ *and* P_v*'s together provide a succinct representation of* $\mathbf{A}$ *when only* NEXT-NEIGHBOR *queries are allowed. In particular,* $\mathbf{A}[v][u] = 1$ *if and only if* $u \in P_v$*. Otherwise,* $\mathbf{A}[v][u] = 0$ *when* $u < \mathbf{last}[v]$ *or* $v < \mathbf{last}[u]$*. In all remaining cases,* $\mathbf{A}[v][u] = \phi$*.*

Proof. The condition for $\mathbf{A}[v][u] = 1$ clearly holds by constuction. Otherwise, observe that $\mathbf{A}[v][u]$ becomes *decided* (i.e. its value is changed from ϕ to 0) during the first call to NEXT-NEIGHBOR(v) that returns a value $u' > u$ thereby setting $\mathbf{last}[v] = u' \implies u < \mathbf{last}[v]$, or vice versa. ◀

3.1.2 Queries and Updates

We now present Algorithm 1, and discuss the correctness of its sampling process. The argument here is rather subtle and relies on viewing the process as an "uncovering" of the table of RVs X_{uv} (introduced in Section 3). Consider the following strategy to find

Algorithm 1 Sampling NEXT-NEIGHBOR.

```
1:  procedure NEXT-NEIGHBOR(v)
2:      u ← last[v]
3:      w_v ← min{(P_v ∩ (u, n]) ∪ {n + 1}}
4:      repeat
5:          sample F ~ F(v, u, w_v)
6:          u ← F
7:      until u = w_v or last[u] < v
8:      if u ≠ w_v
9:          P_v ← P_v ∩ {u}
10:         P_u ← P_u ∩ {v}
11:     last[v] ← u
12:     return u
```

NEXT-NEIGHBOR(v) in the range $(\mathbf{last}[v], w_v)$. Suppose that we generate a sequence of $w_v - \mathbf{last}[v] - 1$ independent coin-tosses, where the i^{th} coin C_{vu} corresponding to $u = \mathbf{last}[v]+i$ has bias p_{vu}, regardless of whether X_{vu} is decided or not. Then, we use the sequence $\langle C_{vu}\rangle$ to assign values to *undecided* random variables X_{vu}. The main observation here is that, the *decided* random variables $X_{vu} = 0$ do not need coin-flips, and the corresponding coin result C_{vu} can be discarded. Thus, we generate coin-flips until we encounter some u satisfying both $C_{vu} = 1$ and $\mathbf{A}[v][u] = \phi$.

Let $\mathsf{F}(v, a, b)$ denote the probability distribution of the occurrence u of the first coin-flip $C_{vu} = 1$ among the neighbors in (a, b). More specifically, $F \sim \mathsf{F}(v, a, b)$ represents the event that $C_{v,a+1} = \cdots = C_{v,F-1} = 0$ and $C_{v,F} = 1$, which happens with probability $\mathbb{P}[F = f] = \prod_{u=a+1}^{f-1}(1 - p_{vu}) \cdot p_{vf}$. For convenience, let $F = b$ denote the event where all $C_{vu} = 0$. Our algorithm samples $F_1 \sim \mathsf{F}(v, \mathbf{last}[v], w_v)$ to find the first occurrence of $C_{v,F_1} = 1$, then samples $F_2 \sim \mathsf{F}(v, F_1, w_v)$ to find the second occurrence $C_{v,F_2} = 1$, and so on. These values $\{F_i\}$ are iterated as u in Algorithm 1. This process generates u satisfying $C_{vu} = 1$ in increasing order, until we find one that also satisfies $\mathbf{A}[u][v] = \phi$ (this outcome is captured by the condition $\mathbf{last}[u] < v$), or until the next generated u is equal to w_v. Note that once the process terminates at some u, we make no implications on the results of any uninspected coin-flips after C_{vu}.

Obstacles for extending beyond `Next-Neighbor` queries. There are two main issues that prevent this method from supporting RANDOM-NEIGHBOR queries. Firstly, while one might consider applying NEXT-NEIGHBOR starting from some random location u to find the minimum $u' \geq u$ where $\mathbf{A}[v][u'] = 1$, the probability of choosing u' will depend on the probabilities p_{vu}'s, and is generally not uniform. Secondly, in Section 3.1.1, we observe that $\mathbf{last}[v]$ and P_v together provide a succinct representation of $\mathbf{A}[v][u] = 0$ only for contiguous cells $\mathbf{A}[v][u]$ where $u \leq \mathbf{last}[v]$ or $v \leq \mathbf{last}[u]$: they cannot handle 0 anywhere else. Unfortunately, in order to support RANDOM-NEIGHBOR queries, we will likely need to assign $\mathbf{A}[v][u]$ to 0 in random locations beyond $\mathbf{last}[v]$ or $\mathbf{last}[u]$, which cannot be captured by the current data structure. Specifically, to speed-up the sampling process for small p_{vu}'s, we must generate many random non-neighbors at once, but we cannot afford to spend time linear in the number of created 0's to update our data structure. We remedy these issues via the following approach.

3.2 Final Implementation Using Blocks

We begin this section by focusing first on RANDOM-NEIGHBOR queries, then extend the construction to the remaining queries. In order to handle RANDOM-NEIGHBOR(v), we divide the neighbors of v into *blocks* $\mathbf{B}_v = \{B_v^{(1)}, B_v^{(2)}, \ldots, B_v^{(i)}, \ldots\}$, so that each block contains, in expectation, roughly the same number of neighbors of v. We implement RANDOM-NEIGHBOR(v) by randomly selecting a block $B_v^{(i)}$, filling in entries $\mathbf{A}[v][u]$ for $u \in B_v^{(i)}$ with 1's and 0's, and then reporting a random neighbor from this block. As the block size may be large when the probabilities are small, instead of using a linear scan, our FILL subroutine will be implemented using the "run-of-0s" sampling from Algorithm 1 (see Section 3.1). Since the number of iterations required by this subroutine is roughly proportional to the number of neighbors, we choose to allocate a constant number of neighbors in expectation to each block: with constant probability the block contains some neighbors, and with high probability it has at most $O(\log n)$ neighbors.

As the actual number of neighbors appearing in each block will be different, we balance out these discrepancies by performing *rejection sampling*. This equalizes the probability of choosing any neighbor implicitly, without the knowledge of $\deg(v)$. Leveraging the fact that the maximum number of neighbors in any block is $\mathcal{O}(\log n)$, we show not only that the probability of success in the rejection sampling process is at least $1/\mathrm{poly}(\log n)$, but the number of iterations required by NEXT-NEIGHBOR is also bounded by $\mathrm{poly}(\log n)$, achieving the overall $\mathrm{poly}(\log n)$ complexities. Here, we will extensively rely on the assumption that the expected number of neighbors for consecutive vertices, $\sum_{u=a}^{b} p_{vu}$, can be approximated efficiently.

3.2.1 Partitioning and Filling the Blocks

We fix a sufficiently large constant L, and assign the vertex u to the $\lceil \sum_{i=1}^{u} p_{vi}/L \rceil^{\text{th}}$ block of v. Essentially, each block represents a contiguous range of vertices, where the expected number of neighbors of v in the block is $\approx L$ (for example, in $G(n,p)$, each block contains $\approx L/p$ vertices). We define $\Gamma^{(i)}(v) = \Gamma(v) \cap B_v^{(i)}$, the neighbors appearing in block $B_v^{(i)}$. Our construction ensures that $L - 1 < \mathbb{E}\left[|\Gamma^{(i)}(v)|\right] < L + 1$ for every $i < |\mathbf{B}_v|$ (i.e., the condition holds for all blocks except possibly the last one).

Now, we show that with high probability, all the block sizes $|\Gamma^{(i)}(v)| = \mathcal{O}(\log n)$, and at least a 1/3-fraction of the blocks are non-empty (i.e., $|\Gamma^{(i)}(v)| > 0$), via the following lemmas (proven in Section B).

▶ **Lemma 13.** *With high probability, the number of neighbors in every block, $|\Gamma^{(i)}(v)|$, is at most $\mathcal{O}(\log n)$.*

▶ **Lemma 14.** *With high probability, for every v such that $|\mathbf{B}_v| = \Omega(\log n)$ (i.e., $\mathbb{E} = \Omega(\log n)$), at least a 1/3-fraction of the blocks $\{B_v^{(i)}\}_{i \in [|\mathbf{B}_v|]}$ are non-empty.*

We consider blocks to be in two possible states – filled or unfilled. Initially, all blocks are considered unfilled. In our algorithm we will maintain, for each block $B_v^{(i)}$, the set $P_v^{(i)}$ of known neighbors of u in block $B_v^{(i)}$; this is a refinement of the set P_v in Section 3.1. We define the behaviors of the procedure FILL(v,i) as follows. When invoked on an unfilled block $B_v^{(i)}$, FILL(v,i) decides whether each vertex $u \in B_v^{(i)}$ is a neighbor of v (implicitly setting $\mathbf{A}[v][u]$ to 1 or 0) unless X_{vu} is already decided; in other words, update $P_v^{(i)}$ to $\Gamma^{(i)}(v)$. Then $B_v^{(i)}$ is marked as filled. We postpone the description of our implementation of FILL to Section 3.3, instead using it as a black box.

3.2.2 Putting it all together: `Random-Neighbor` queries

Algorithm 2 Block sampling.

procedure RANDOM-NEIGHBOR(v)
  while True
    sample $B_v^{(i)} \sim_{\mathcal{U}} \mathbf{B}_v$ u.a.r.
    if $B_v^{(i)}$ is not *filled*
      FILL(v, i)
    with probability $\frac{|P_v^{(i)}|}{M}$
      return $u \sim_{\mathcal{U}} P_v^{(i)}$ u.a.r

Consider Algorithm 2 for sampling a random neighbor via rejection sampling. For simplicity, throughout the analysis, we assume $|\mathbf{B}_v| = \Omega(\log n)$; otherwise, invoke FILL$(v, i)$ for all $i \in [|\mathbf{B}_v|]$ to obtain the entire neighbor list $\Gamma(v)$.

To obtain a random neighbor, we first choose a block $B_v^{(i)}$ uniformly at random, and invoke FILL(v, i) if the block is *unfilled*. Then, we *accept* the sampled block for generating our random neighbor with probability proportional to $|P_v^{(i)}|$. More specifically, if $M = \Theta(\log n)$ is an upper bound on the maximum number of neighbors in any block (see Lemma 13), we accept block $B_v^{(i)}$ with probability $|P_v^{(i)}|/M$, which is well-defined (i.e., does not exceed 1) with high probability. Note that if $P_v^{(i)} = \emptyset$, we sample another block. If we choose to accept $B_v^{(i)}$, we return a random neighbor from $P_v^{(i)}$. Otherwise, *reject* this block and repeat the process again.

Since the returned vertex is always a member of $P_v^{(i)}$, a valid neighbor is always returned. We now show that the algorithm correctly samples a uniformly random neighbor and bound the number of iterations required for the rejection sampling process.

▶ **Lemma 15.** *Algorithm 2 returns a uniformly random neighbor of vertex v.*

Proof. It suffices to show that the probability that any neighbor in $\Gamma(v)$ is return with uniform positive probability, within the same iteration. Fixing a single iteration and consider a vertex $u \in P_v^{(i)}$, we compute the probability that u is accepted. The probability that $B_v^{(i)} i$ is chosen is $1/|\mathbf{B}_v|$, the probability that $B_v^{(i)}$ is accepted is $|P_v^{(i)}|/M$, and the probability that u is chosen among $P_v^{(i)}$ is $1/|P_v^{(i)}|$. Hence, the overall probability of returning u in a single iteration of the loop is $1/(|\mathbf{B}_v| \cdot M)$, which is positive and independent of u. Therefore, each vertex is returned with the same probability. ◀

▶ **Lemma 16.** *Algorithm 2 terminates in $\mathcal{O}(\log n)$ iterations in expectation, or $\mathcal{O}(\log^2 n)$ iterations w.h.p.*

Proof. Using Lemma 14, we conclude that the probability of choosing a non-empty block is at least $1/3$. Since $M = \Theta(\log n)$ by Lemma 13, the success probability of each iteration is at least $1/(3M) = \Omega(1/\log n)$, Thus, the number of iterations required is $O(\log^2 n)$ with high probability. ◀

3.3 Implementation of `Fill`

Lastly, we describe the implementation of the FILL procedure, employing the approach of skipping non-neighbors, as developed for Algorithm 1. We aim to simulate the following process: perform coin-tosses C_{vu} with probability p_{vu} for every $u \in B_v^{(i)}$ and update $\mathbf{A}[v][u]$'s

Algorithm 3 Filling a block.

```
procedure FILL(v, i)
    (a, b) ← B_v^(i)
    while a < b
        sample u ~ F(v, a, b)
        B_u^(j) ← block containing v
        if B_u^(j) is not filled
            P_v^(i) ← P_v^(i) ∪ {u}
            P_u^(j) ← P_u^(j) ∪ {v}
        a ← u
    mark B_u^(j) as filled
```

according to these coin-flips unless they are decided (i.e., $\mathbf{A}[v][u] \neq \phi$). We directly generate a sequence of u's where the coins $C_{vu} = 1$, then add u to P_v and vice versa if X_{vu} has not previously been decided. Thus, once $B_v^{(i)}$ is filled, we will obtain $P_v^{(i)} = \Gamma^{(i)}(v)$ as desired.

As discussed in Section 3.1, while we have recorded all occurrences of $\mathbf{A}[v][u] = 1$ in $P_v^{(i)}$, we need an efficient way of checking whether $\mathbf{A}[v][u] = 0$ or ϕ. In Algorithm 1, **last** serves this purpose by showing that $\mathbf{A}[v][u]$ for all $u \leq \mathbf{last}[v]$ are decided as shown in Lemma 12. Here instead, we maintain a single bit marking whether each block is filled or unfilled: a filled block implies that $\mathbf{A}[v][u]$ for all $u \in B_v^{(i)}$ are decided. The block structure along with the mark bits, unlike **last**, is capable of handling intermittent ranges of intervals, which is sufficient for our purpose, as shown in the following lemma. This yields the implementation of Algorithm 3 for the FILL procedure fulfilling the requirement previously given in Section 3.2.1.

▶ **Lemma 17.** *The data structures $P_v^{(i)}$'s and the block marking bits together provide a succinct representation of $\mathbf{A}$ as long as modifications to $\mathbf{A}$ are performed solely by the* FILL *operation in Algorithm 3. In particular, let $u \in B_v^{(i)}$ and $v \in B_u^{(j)}$. Then, $\mathbf{A}[v][u] = 1$ if and only if $u \in P_v^{(i)}$. Otherwise, $\mathbf{A}[v][u] = 0$ when at least one of $B_v^{(i)}$ or $B_u^{(j)}$ is marked as* filled. *In all remaining cases, $\mathbf{A}[v][u] = \phi$.*

Proof. The condition for $\mathbf{A}[v][u] = 1$ still holds by construction. Otherwise, observe that $\mathbf{A}[v][u]$ becomes decided precisely during a FILL(v, i) or a FILL(u, j) operation, which thereby marks one of the corresponding blocks as filled. ◀

Note that $P_v^{(i)}$'s, maintained by our implementation, are initially empty but may not still be empty at the beginning of the FILL function call. These $P_v^{(i)}$'s are again instantiated and stored in a dictionary once they become non-empty. Further, observe that the coin-flips are simulated independently of the state of $P_v^{(i)}$, so the number of iterations of Algorithm 3 is the same as the number of coins $C_{vu} = 1$ which is, in expectation, a constant (namely $\sum_{u \in B_v^{(i)}} \mathbb{P}[C_{vu} = 1] = \sum_{u \in B_v^{(i)}} p_{vu} \leq L + 1$).

By tracking the resource required by Algorithm 3 we obtain the following lemma; note that "additional space" refers to the enduring memory that the implementation must allocate and keep even after the execution, not its computation memory. The $\log n$ factors in our complexities are required to perform binary-search for the range of $B_v^{(i)}$, or for the value u from the CDF of $\mathsf{F}(u, a, b)$, and to maintain the ordered sets $P_v^{(i)}$ and $P_u^{(j)}$.

▶ **Lemma 18.** *Each execution of Algorithm 3 (the* FILL *operation) on an* unfilled *block $B_v^{(i)}$, in expectation:*

- *terminates within $\mathcal{O}(1)$ iterations (of its* **repeat** *loop);*
- *computes $\mathcal{O}(\log n)$ quantities of $\prod_{u \in [a,b]}(1 - p_{vu})$ and $\sum_{u \in [a,b]} p_{vu}$ each;*
- *uses an additional $\mathcal{O}(\log n)$ time, $\mathcal{O}(1)$ random $\log n$-bit words, and $\mathcal{O}(1)$ additional space.*

Observe that the number of iterations required by Algorithm 3 only depends on its random coin-flips and independent of the state of the algorithm. Combining with Lemma 16, we finally obtain polylogarithimc resource bound for our implementation of RANDOM-NEIGHBOR.

▶ **Corollary 19.** *Each execution of Algorithm 2 (the* RANDOM-NEIGHBOR *query), with high probability,*

- *terminates within* $\mathcal{O}(\log^2 n)$ *iterations (of its* **repeat** *loop);*
- *computes* $\mathcal{O}(\log^3 n)$ *quantities of* $\prod_{u\in[a,b]}(1-p_{vu})$ *and* $\sum_{u\in[a,b]} p_{vu}$ *each;*
- *uses an additional* $\mathcal{O}(\log^3 n)$ *time,* $\mathcal{O}(\log^2 n)$ *random words, and* $\mathcal{O}(1)$ *additional space.*

Supporting Other Query Types along with `Random-Neighbor`

- VERTEX-PAIR(u,v): We simply need to make sure that Lemma 17 holds, so we first apply FILL(u, j) on block $B_u^{(j)}$ containing v (if needed), then answer accordingly.
- NEXT-NEIGHBOR(v): We maintain **last**, and keep invoking FILL until we find a neighbor. Recall that by Lemma 14, the probability that a particular block is empty is a small constant. Then with high probability, there exists no $\omega(\log n)$ *consecutive* empty blocks $B_v^{(i)}$'s for any vertex v, and thus NEXT-NEIGHBOR only invokes up to $\mathcal{O}(\log n)$ calls to FILL.

We summarize the results so far with through the following theorem.

▶ **Theorem 1.** *Given a random graph model defined by the probability matrix* $\{p_{uv}\}_{u,v\in[n]}$*, and assuming that we can compute the the quantities* $\prod_{u=a}^{b}(1-p_{vu})$ *and* $\sum_{u=a}^{b} p_{vu}$ *in* $\mathcal{O}(\log^{\mathcal{O}(1)} n)$ *time, there exists an implementation for this model that supports local access through* RANDOM-NEIGHBOR*,* VERTEX-PAIR*, and* NEXT-NEIGHBOR *queries using* $\mathcal{O}(\log^{\mathcal{O}(1)} n)$ *running time, additional space, and random bits per query.*

We have also been implicitly assuming perfect-precision arithmetic and we relax this assumption in Section B.1. In the following Section 3.4, we show applications of Theorem 1 to the $G(n,p)$ model, and the Stochastic Block model under random community assignment, by providing formulas and by constructing data structures for computing the quantities specified in Theorem 1.

3.4 Applications to Erdös-Rényi Model and Stochastic Block Model

In this section we demonstrate the application of our techniques to two well known, and widely studied models of random graphs. That is, as required by Theorem 1, we must provide a method for computing the quantities $\prod_{u=a}^{b}(1-p_{vu})$ and $\sum_{u=a}^{b} p_{vu}$ of the desired random graph families in logarithmic time, space and random bits. Our first implementation focuses on the well known Erdös-Rényi model – $G(n,p)$: in this case, $p_{vu} = p$ is uniform and our quantities admit closed-form formulas.

Next, we focus on the Stochastic Block model with randomly-assigned communities. Our implementation assigns each vertex to a community in $\{C_1, \ldots, C_r\}$ identically and independently at random, according to some given distribution R over the communities. We formulate a method of sampling community assignments locally. This essentially allows us to sample from the *multivariate hypergeometric distribution*, using $\text{poly}(\log n)$ random bits, which may be of independent interest. We remark that, as our first step, we sample for the number of vertices of each community. That is, our construction can alternatively support the community assignment where the number of vertices of each community is given, under the assumption that the *partition* of the vertex set into communities is chosen uniformly at random.

3.4.1 Erdös-Rényi Model

As $p_{vu} = p$ for all edges $\{u, v\}$ in the Erdös-Rényi $G(n, p)$ model, we have the closed-form formulas $\prod_{u=a}^{b}(1-p_{vu}) = (1-p)^{b-a+1}$ and $\sum_{u=a}^{b} p_{vu} = (b-a+1)p$, which can be computed in constant time according to our assumption, yielding the following corollary.

▶ **Corollary 2.** *There exists an algorithm that implements local access to a Erdös-Rényi $G(n, p)$ random graph, through* Vertex-Pair, Next-Neighbor, *and* Random-Neighbor *queries, while using $\mathcal{O}(\log^3 n)$ time, $\mathcal{O}(\log^3 n)$ random bits, and $\mathcal{O}(\log^2 n)$ additional space per query with high probability.*

We remark that there exists an alternative approach that picks $F \sim \mathsf{F}(v, a, b)$ directly via a closed-form formula $a + \lceil \frac{\log U}{\log(1-p)} \rceil$ where U is drawn uniformly from $[0, 1)$, rather than binary-searching for U in its CDF. Such an approach may save some $\mathrm{poly}(\log n)$ factors in the resources, given the prefect-precision arithmetic assumption. This usage of the log function requires $\Omega(n)$-bit precision, which is not applicable to our computation model.

While we are able to generate our random graph on-the-fly supporting all three types of queries, our construction still only requires $\mathcal{O}(m + n)$ space ($\log n$-bit words) in total at any state; that is, we keep $\mathcal{O}(n)$ words for **last**, $\mathcal{O}(1)$ words per neighbor in P_v's, and one marking bit for each block (where there can be up to $m + n$ blocks in total). Hence, our memory usage is nearly optimal for the $G(n, p)$ model:

▶ **Corollary 4.** *The final algorithm in Section 3 can generate a complete random graph from the Erdös-Rényi $G(n, p)$ model using overall $\widetilde{\mathcal{O}}(n + m)$ time, random bits and space, which is $\widetilde{\mathcal{O}}(pn^2)$ in expectation. This is optimal up to $\mathcal{O}(\mathrm{poly}(\log n))$ factors.*

3.4.2 Stochastic Block model

In the Stochastic Block model, each vertex is assigned to some community C_i, $i \in [r]$. By partitioning the product by communities, we may rewrite the desired formulas, for $v \in C_i$, as $\prod_{u=a}^{b}(1 - p_{vu}) = \prod_{j=1}^{r}(1 - p_{ij})^{|[a,b]\cap C_j|}$ and $\sum_{u=a}^{b} p_{vu} = \sum_{j=1}^{r} |[a, b] \cap C_j| \cdot p_{ij}$. Thus, it suffices to design a data structure that is able to efficiently count the number of occurrences of vertices of each community in any contiguous range (namely the value $|[a, b] \cap C_j|$ for each $j \in [r]$), where the vertices are assigned communities according to a given distribution R. To this end, we use the following lemma, yielding an implementation for the Stochastic Block model using $O(r\,\mathrm{poly}(\log n))$ resources per query.

▶ **Theorem 20.** *There exists a data structure that samples a community for each vertex independently at random from R with $\frac{1}{\mathrm{poly}(n)}$ error in the L_1-distance, and supports queries that ask for the number of occurrences of vertices of each community in any contiguous range, using $O(r\,\mathrm{poly}(\log n))$ time, random bits, and additional space per query. Further, this data structure may be implemented in such a way that requires no overhead for initialization.*

▶ **Corollary 3.** *There exists an algorithm that implements local access to a random graph from the n-vertex Stochastic Block Model with r randomly-assigned communities, through* Vertex-Pair, Next-Neighbor, *and* Random-Neighbor *queries, using $O(r\,\mathrm{poly}(\log n))$ time, random bits, and space per query w.h.p.*

We provide the full details of the construction in Section D. Our construction extends a similar implementation in the work of [26] which only supports $r = 2$. The overall data structure is a balanced binary tree, where the root corresponds to the entire range of indices $[n]$, and the children of each vertex correspond to the first and second half of the parent's range.

Each node[3] holds the number of vertices of each community in its range. The tree initially contains only the root, with the number of vertices of each community $\langle |C_1|, |C_2|, \cdots, |C_r| \rangle$ sampled according to the multinomial distribution (for n samples from the distribution R). The children are generated top-down on an as-needed basis according to the given queries. The technical difficulties arise when generating the children, where one needs to sample the counts assigned to either child from the correct marginal distribution. We show how to sample such a count from the *multivariate hypergeometric distribution*, below in Theorem 21 (proven in Section D).

▶ **Theorem 21.** *Given B marbles of r different colors, such that there are C_i marbles of color i, there exists an algorithm that samples $\langle s_1, s_2, \cdots, s_r \rangle$, the number of marbles of each color appearing when drawing l marbles from the urn without replacement, in $O(r \cdot \mathrm{poly}(\log B))$ time and random words.*

Proof of Theorem 20. Recall that R denotes the given distribution over integers $[r]$ (namely, the random distribution of communities for each vertex). Our algorithm generates and maintains random variables $X_1, \ldots, X_n$ (denoting the community assignment), each of which is drawn independently from R. Given a pair (a, b), it uses Theorem 20 to sample the vector $\mathbf{C}(a, b) = \langle c_1, \ldots, c_r \rangle$, where c_k counts the number of variables in $\{X_a, \ldots, X_b\}$ that take on the value k.

We maintain a complete binary tree whose leaves corresponds to indices from $[n]$. Each node represents a range and stores the vector $\mathbf{C}$ for the corresponding range. The root represents the entire range $[n]$, which is then halved in each level. Initially the root samples $\mathbf{C}(1, n)$ from the multinomial distribution according to R (see e.g., Section 3.4.1 of [32]). Then, the children are generated on-the-fly as described above. Thus, each query can be processed within $O(r \,\mathrm{poly}(\log n))$ time, yielding Theorem 20. ◀

Then, by embedding the information stored by the data structure into the state (as in the proof of Lemma 51), we obtain the desired Corollary 3.

4 Implementing Random Catalan Objects

In the previous Section 3.4.2 on the Stochastic Block Model, we considered random sequences of colored marbles. Next, we focus on an important variant of these sequences as Catalan objects, which impose a global constraint on the types of allowable sequences. Specifically, consider a sequence of n white and n black marbles, such that every *prefix* of the sequence has at least as many white marbles as black ones. Our goal will be to support queries to a uniformly random instance of such an object.

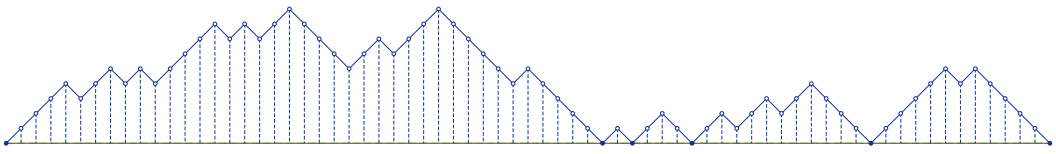


Figure 1 Simple Dyck path with $n = 35$ up and down steps.

[3] For clarity, "vertex" is only used in the sampled graph, and "node" is only used in the internal data structures.

One interpretation of Catalan objects is given by Dyck paths (Figure 1). A Dyck path is essentially a $2n$ step *balanced* one-dimensional walk with exactly n up and down steps. In Figure 1, each step moves one unit along the positive x-axis (time) and one unit up or down the positive y-axis (position). The prefix constraint implies that the y-coordinate of any point on the walk is ≥ 0 i.e. the walk never crosses the x-axis. The number of possible Dyck paths (see Theorem 62) is the n^{th} Catalan number $C_n = \frac{1}{n+1} \cdot \binom{2n}{n}$. Many important combinatorial objects occur in Catalan families of which these are an example.

We will approach the problem of partially sampling Catalan objects through Dyck paths. This, in turn, will allow us to implement access other random Catalan objects such as rooted trees, and bracketed expressions. Specifically, we will want to answer the following queries:

- DIRECTION(t): Returns the value of the t^{th} step in the Dyck path (whether the step is up or down).
- HEIGHT(t): Returns the y-position of the path after t steps (the number of up steps minus the number of down steps among the first t steps). Since a DIRECTION(t) query can be simulated using the queries HEIGHT(t) and HEIGHT($t-1$), we will not explicitly discuss the DIRECTION queries in what follows.
- FIRST-RETURN(t): If the $(t+1)^{th}$ step is upwards i.e. HEIGHT($t+1$) = HEIGHT(t) + 1, it returns the smallest index $t' > t$ such that HEIGHT(t') = HEIGHT(t). While it may not be clear why this query is important, it will be useful for querying bracketed expressions and random trees. (see Section 4.1).

4.1 Bijections to other Catalan objects

The HEIGHT query is natural for Dyck paths, but the FIRST-RETURN query is important in exploring other Catalan objects. For instance, consider a random well bracketed expression; equivalently an uniform distribution over the Dyck language. One can construct a trivial bijection between Dyck paths and words in this language by replacing up and down steps with opening and closing brackets respectively (Figure 2). The HEIGHT query corresponds to asking for the nesting depth at a certain position in the word, and FIRST-RETURN(x) returns the position after the matched closing bracket for the step $(x \to x+1)$.

There is also a natural bijection between Dyck paths and ordered rooted trees (Figure 2), by viewing the Dyck path as a transcript of the tree's DFS traversal. Starting with the root, for each "up-step" we move to a new child of the current node, and for each "down-step", we backtrack towards the root. Thus, the HEIGHT query returns the depth of a node. Also, since the Dyck path is a DFS transcript of the tree, a FIRST-RETURN query on the path can be used to find successive children of a tree node (each return produces the *next child*). For instance, in Figure 2, we can invoke FIRST-RETURN thrice starting at the first *yellow path position* to reveal the corresponding three children of the *yellow tree node*.

By definition, FIRST-RETURN(x) is meaningful only when the step from x to $x+1$ is upwards, i.e. when HEIGHT($x+1$) = HEIGHT(x)+1. We can also implement a REVERSE-FIRST-RETURN query, which is just a standard FIRST-RETURN query on the reversed Dyck path (consider a reversal of the green dashed arrows in Figure 2). The reversal implies that REVERSE-FIRST-RETURN(x) is only meaningful when HEIGHT($x-1$) = HEIGHT(x) + 1. In terms of bijections, REVERSE-FIRST-RETURN is equivalent to finding a matching opening bracket in bracketed expressions, and a PREVIOUS-CHILD query in rooted trees. We show how to implement this query in Section 4.4.7. In the case where the height at x is larger than both the heights at $x-1$ and $x+1$ (boxed nodes in Figure 2), there is no meaningful "first return" from the context of the bijections. Specifically, these nodes correspond to leaf nodes in rooted trees, or to a terminal nesting level in the bracket expression.

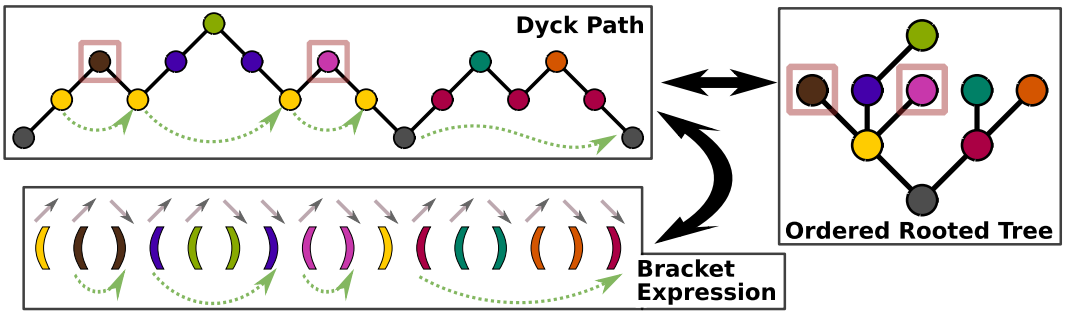


Figure 2 Bijections from Dyck paths to bracketed expressions and ordered rooted trees. Note that the color coded *positions* on the path correspond to tree nodes, and the path itself is a DFS traversal of the tree. The bijection to bracketed expressions proceeds by directly replacing up and down steps with opening and closing brackets respectively (we color the brackets with the *higher* endpoint of the corresponding step). Green dashed arrows show successive FIRST-RETURN queries on the path. Using the bijection, this is equivalent to revealing three child sub-trees in order from left to right for the corresponding tree node, and also to finding matching brackets in bracketed expressions. FIRST-RETURN is not defined for the red boxed nodes in either forward or reverse direction, since this position corresponds to a leaf node in the tree, or to a terminal nesting level in the bracket expression.

Moving forwards, we will focus on Dyck paths for the sake of simplicity.

4.2 Catalan Trapezoids and Generalized Dyck Paths

In order to implement local access to random Dyck paths, we will need to analyze more general Catalan objects. Specifically, we consider a sequence of U up-steps and D down-steps, such that any prefix of the sequence containing U' up and D' down steps satisfies $U' - D' \geq 1 - k$. This means that we start our Dyck path at a height of $k - 1$, and we are never allowed to cross below zero (Figure 3). Note that the case $k = 1$ corresponds to the standard description of Dyck paths, as mentioned previously (Figure 1).

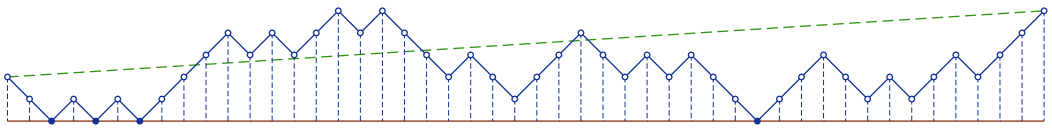


Figure 3 Generalized Dyck path with $U = 25$, $D = 22$ and $k = 3$. Note that the boundary is $k - 1 = 2$ units below the starting height.

We will denote the set of such *generalized Dyck paths* as $\mathbb{C}_k(U, D)$ and the number of paths as $C_k(U, D) = |\mathbb{C}_k(U, D)|$, which is an entry in the *Catalan Trapezoid* of order k [47]. We also use $\mathsf{C}_k(U, D)$ to denote the uniform distribution over $\mathbb{C}_k(n, m)$. Now, we state a result from [47] without proof:

$$C_k(U,D) = \begin{cases} \binom{U+D}{D} & 0 \leq D < k \\ \binom{U+D}{D} - \binom{U+D}{D-k} & k \leq D \leq U + k - 1 \\ 0 & D > U + k - 1 \end{cases} \tag{1}$$

For $k = 1$ and $n = m$, these represent the vanilla Catalan numbers i.e. $C_n = C_1(n, n)$ (number of simple Dyck paths). Our goal is to sample from the distribution $\mathsf{C}_1(n, n)$.

Consider the situation after a sequence of HEIGHT queries to the Dyck path at various locations $\langle x_1, x_2, \cdots, x_m \rangle$, such that the corresponding heights were sampled to be $\langle y_1, y_2, \cdots, y_m \rangle$. These revealed locations partition the path into disjoint *intervals* $[x_i, x_{i+1}]$, where the heights of the endpoints of each interval have been determined (as $y_i = \textsc{Height}(x_i)$). We notice that these intervals can be generated independently of each other. Specifically, the path within the interval $[x_i, x_{i+1}]$ will be sampled from $\mathsf{C}_k(U, D)$, where $k - 1 = y_i$, $U + D = x_{i+1} - x_i$, and $U - D = y_{i+1} - y_i$. Moreover, since the heights of the endpoints y_i and y_{i+1} are known, this choice is independent of any samples outside the interval. Next, in Section 4.3, we will show how one can determine heights within such an interval, and in Section 4.4 we will move on to the more complicated FIRST-RETURN queries.

4.3 Implementing Height queries

We implement $\textsc{Height}(t)$ by showing how to efficiently determine the height of the path at the midpoint of an existing interval $[x_i, x_{i+1}]$ (with corresponding endpoint heights y_i, y_{i+1}), which results in two sub-intervals that are half the size. Next, we extend this strategy to determine the heights of arbitrary positions by recursively sub-dividing the relevant interval (binary search). If the interval in question has odd length, we sample a single step from an endpoint, and proceed with a shortened even length interval. Sampling a single step is easy since there are only two outcomes (see proof of Theorem 28).

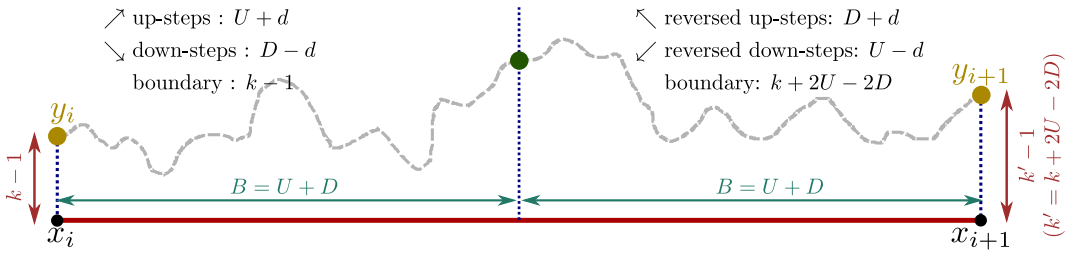


Figure 4 The $2B$-interval is split into two equal parts resulting in two separate Dyck problems. The green node (center) is the sampled height of the midpoint parameterized by the value of d. The path considered in both sub-intervals starts at a yellow node (left and right edges) and ends at the green node. From this perspective, the path on the right is reversed with up and down steps being swapped. A possible path is shown in gray.

Our general recursive step is as follows. We consider an interval of length $2B$ comprising of $2U$ up-steps and $2D$ down-steps where the sum of any prefix cannot be less than $k - 1$ i.e. the path within this interval should be sampled from $\mathsf{C}_k(2U, 2D)$. In order to make the analysis simpler, we have assumed that the number of up and down steps are both even. The case of sampling according to $\mathsf{C}_k(2U + 1, 2D + 1)$ works similarly with slightly different formulae. Without loss of generality, we assume that $D \leq U$; if this were not the case, we could simply flip the interval, swap the up and down steps, and modify the prefix constraint to $k' = k + 2U - 2D$ (Figure 4). This ensures that the overall path in the interval is non-decreasing in height, which will simpify our analysis.

We determine the height of the path $B = U + D$ steps into the interval at the midpoint (see Figure 4). This is equivalent to finding the number of up or down steps that get assigned to the first half of the interval. We parameterize the possibilities by d and define p_d to be the probability that exactly $U + d$ up-steps and $D - d$ down steps get assigned to the first half (with the remaining $U - d$ up steps and $D + d$ down steps being assigned to the second half).

$$p_d = \frac{S_{left}(d) \cdot S_{right}(d)}{S_{total}(d)} \tag{2}$$

Here, $S_{left}(d)$ denotes the number of possible paths in the first half (using $U + d$ up steps) and $S_{right}(d)$ denotes the number of possible paths in the second half (using $U - d$ up steps). Note that all of these paths have to respect the k-boundary constraint (cannot dip more than $k - 1$ units below the starting height), where $k = y_i + 1$. Moving forwards, we will drop the d when referring to the path counts. We (conceptually) flip the second half of the interval, such that the corresponding path begins from the end of the $2B$-interval and terminates at the midpoint (Figure 4). This results in a different starting point, and the prefix/boundary constraint will also be different. Hence, we define $k' = k + 2U - 2D$ to represent the new boundary constraint (since the final height of the $2B$-interval is $k' - 1$). Finally, S_{total} is the total number of possible paths in the $2B$ interval.

We cannot directly sample from this complicated distribution $\{p_d\}$. Instead, we use the rejection sampling strategy from Lemma 11. An important point to note is that in order to apply this lemma, we must be able to approximate the p_d values. However, we cannot naively use the formula from Equation 2, since the values of $\{S_{left}, S_{right}, S_{total}\}$ are too large to compute explicitly. Lemma 67 in Section F.3 shows how to indirectly compute the probabilty approximations. We also use the following lemma to bound the deviation of the path with high probability. A proof is presented in Section F.2.

▶ **Lemma 22.** *Consider a contiguous* sub-path *of a simple Dyck path of length* $2n$ *where the sub-path is of length* $2B$ *comprising of* U *up-steps and* D *down-steps (with* $U + D = 2B$*). Then there exists a constant* c *such that the quantities* $|B - U|$, $|B - D|$, *and* $|U - D|$ *are all* $< c\sqrt{B \log n}$ *with probability at least* $1 - 1/n^2$ *for every possible sub-path.*

This lemma allows us to ignore potential midpoint heights that cause a deviation greater than $c\sqrt{B \log n}$. A direct implication is that with high probability, the correctly sampled value for d will be $\mathcal{O}(\sqrt{B \log n})$. In other words, the height of the midpoint takes on one of only $\mathcal{O}(\sqrt{B \log n})$ distinct values with high probability. This immediately suggests a $\widetilde{\mathcal{O}}(\sqrt{B})$ time algorithm for determining the midpoint height, by explicitly computing the probabilities of each of these potential heights, and directly sampling from the resulting distribution. However, we can go further and obtain a $\mathcal{O}(poly(\log n))$ time algorithm.

4.3.1 The Simple Case: Far Boundary

We first consider the case when the boundary constraint is far away from the starting point, i.e. k is large. The following lemma (proof in Section F.2) shows that in this case, we can safely *ignore* the constraint. Intuitively, this is because the boundary is so far away, that with high probability, we do not hit it even if we choose a random *unconstrained* path.

▶ **Lemma 23.** *Given a Dyck path sampling problem of length* B *with* U *up,* D *down steps, and a boundary at* k*, there exists a constant* c *such that if* $k > c\sqrt{B \log n}$*, then the distribution of paths sampled without a boundary* $\mathsf{C}_\infty(U, D)$ *is* $\mathcal{O}(1/n^2)$*-close in* L_1 *distance to the distribution of Dyck paths* $\mathsf{C}_k(U + D)$*.*

By Lemma 23, the problem of sampling from $C_k(2U, 2D)$ reduces to sampling from the hypergeometric distribution $C_\infty(2U, 2D)$ when $k > \mathcal{O}(\sqrt{B \log n})$ i.e. the probabilities p_d can be approximated by:

$$q_d = \frac{\binom{B}{D-d} \cdot \binom{B}{D+d}}{\binom{2B}{2D}}$$

This problem of sampling from the hypergeometric distribution is implemented using $\mathcal{O}(poly(\log n))$ resources in [26] (see Lemma 54 in Section D). We also used this result earlier in the paper in order to find the community assignments in the Stochastic Block Model (Section 3.4.2).

4.3.2 The Difficult Case: Intervals Close to Zero

The difficult case is when $k = \mathcal{O}(\sqrt{B \log n})$, and the previous approximation due to Lemma 23 no longer works. In this case, we cannot just ignore the boundary constraint, and instead we have to analyze the true probability distribution given by p_d. We obtain an expression for p_d by substituting the formula for generalized Catalan numbers as follows: (Equation 1) into Equation 2.

$$S_{left} = C_k(U + d, D - d) \qquad S_{right} = C_{k'}(U - d, D + d) \qquad S_{total} = C_k(2U, 2D) \tag{3}$$

Since the right interval is flipped in our analysis, this changes the prefix/boundary constraint, and hence, the expression for S_{right} uses $k' = k + 2U - 2D$. This also implies that $k' = \mathcal{O}(\sqrt{B \log n})$ (using Lemma 22). We can now use Equation 2 to evaluate the probabilties $p_d = S_{left} \cdot S_{right}/S_{total}$. Recall that S_{left} and S_{right} are the number of possible paths in the left and right half of the interval, when exactly $U + d$ up steps are assigned to the first half, and S_{total} is the total number of possible paths in the interval.

We will invoke the rejection sampling technique (Lemma 11), by constructing a different distribution q_d that approximates p_d up to logarithmic factors over the vast majority of its support (we ignore all $|d| > \Theta(\sqrt{B \log n})$ since the associated probability mass is negligible by Lemma 22). In order to perform rejection sampling, we also need good approximations of p_d, which is acheved by Lemma 67 in Section F.3. Next, we define an appropriate q_d that approximates p_d and also has an *efficiently computable CDF*. Surprisingly, as in Section 4.3.1, we will be able to use the hypergeometric distribution for q_d,

$$q_d \equiv \frac{\binom{B}{D-d} \cdot \binom{B}{D+d}}{\binom{2B}{2D}} = \frac{\binom{B}{D-d} \cdot \binom{B}{U-d}}{\binom{2B}{2D}}$$

However, the argument for why this q_d is a good approximation to p_d is far less straightforward.

First, we consider the case where $k \cdot k' \leq 2U + 1$. In this case, we use loose bounds for $S_{left} < \binom{B}{D-d}$ and $S_{right} < \binom{B}{U-d}$. These are true because $\binom{B}{D-d}$ and $\binom{B}{U-d}$ are the total number of *unconstrained* paths in the left and right half respectively, and adding the boundary constraint can only reduce the number of paths. We also prove the following lemma in Section F.4 to bound the value of S_{total}.

▶ **Lemma 24.** *When $kk' > 2U + 1$, $S_{total} > \frac{1}{2} \cdot \binom{2B}{2D}$.*

Combining the three bounds, we conclude that $p_d < \frac{1}{2} q_d$. Intuitively, in this case the Dyck boundary is far away, and therefore the number of possible paths is only a constant factor away from the number of unconstrained paths (see Section 4.3.1). The case where the boundaries are closer (i.e. $k \cdot k' \leq 2U + 1$) is trickier, since the individual counts need not be close to the corresponding binomial counts. However, in this case we can still ensure that the sampling probability is within poly-logarithmic factors of the binomial sampling probability. We use the following lemmas to bound S_{left} and S_{right} (proofs in Section F.4).

▶ **Lemma 25.** *$S_{left} \leq c_1 \frac{k \cdot \sqrt{\log n}}{\sqrt{B}} \cdot \binom{B}{D-d}$ for some constant c_1.*

▶ **Lemma 26.** *$S_{right} \leq c_2 \frac{k' \cdot \sqrt{\log n}}{\sqrt{B}} \cdot \binom{B}{U-d}$ for some constant c_2.*

Finally, we obtain a diferent bound on S_{total} for the *near boundary* case.

▶ **Lemma 27.** *When $kk' \leq 2U + 1$, $S_{total} \geq c_3 \frac{k \cdot k'}{B} \cdot \binom{2B}{2D}$ for some constant c_3.*

Multiplying these inequalities together allows us to bound $p_d = S_{left} \cdot S_{right}/S_{total} \leq \Theta(q_d \log n)$, implying $p_d/q_d \leq \Theta(\log n)$. Consequently, we can leverage Lemma 11 to obtain a sample from $\{p_d\}$ using $\mathcal{O}(\log^{\mathcal{O}(1)} n)$ samples from $\{q_d\}$, thus determining the height of the Dyck path at the midpoint of the interval.

▶ **Theorem 28.** *Given an interval $[x_i, x_{i+1}]$ (and the associated endpoint heights y_i, y_{i+1}) in a Dyck path of length $2n$, with the guarantee that no position between x_i and x_{i+1} has been sampled yet, there is an algorithm that returns the height of the path at the midpoint of x_i and x_{i+1} (or next to the midpoint if $x_{i+1} - x_i$ is odd). Moreover, this algorithm only uses $\mathcal{O}(poly(\log n))$ resources.*

Proof. If $x_{i+1} - x_i$ is even, we can set $B = (x_{i+1} - x_i)/2$. Otherwise, we first sample a single step from x_i to $x_i + 1$, and then set $B = (x_{i+1} - x_i - 1)/2$. Since there are only two possibilities for a single step, we can explicitly approximate the corresponding probabilities using the strategy outlined in the proof of Lemma 67 (see Section F.3), and then sample accordingly. This allows us to apply the rejection sampling from Lemma 11 using $\{q_d\}$ to obtain samples from $\{p_d\}$ as defined above. ◀

▶ **Theorem 29.** *There is an algorithm that provides sample access to a Dyck path of length $2n$, by answering queries of the form* HEIGHT(x) *with the correctly sampled height of the Dyck path at position x using only $\mathcal{O}(poly(\log n))$ resources per query.*

Proof. The algorithm maintains a successor-predecessor data structure (e.g. Van Emde Boas tree) to store all positions x that have already been determined. Each newly determined position is added to this structure. Given a query HEIGHT(x), the algorithm first finds the successor and predecessor (say a and b) of x among the already queried positions. This provides us the guarantee required to apply Theorem 28, which allows us to query the height at the midpoint of a and b. We then binary search by updating either the successor or predecessor of x and repeat until we determine the height of position x. ◀

4.4 Supporting "First Return" Queries

In this section, we discuss more complex queries to a Dyck path. Specifically, we implement FIRST-RETURN(x) to allow the user to query the next time the path returns to HEIGHT(x). The utility of this kind of query is illustrated through bijections to other random Catalan objects in Section 4.1. An important detail here is that if the first step from x to $x + 1$ is a down-step, there is no well defined FIRST-RETURN.

4.4.1 Maintaining a Boundary Invariant

Notice that after performing a set of HEIGHT queries $\langle x_1, x_2, \cdots, x_m \rangle$ to the Dyck path, many different positions are revealed (possibly in adversarial locations). This partitions the path into at most $m + 1$ disjoint and independent *intervals* with known boundary conditions. The first step towards finding the FIRST-RETURN from position t would be to locate the *interval* where the first return occurs. Even if we had an efficient technique to filter intervals, we would want to avoid considering all $\Theta(m)$ intervals to find the correct one. In addition, the fact that a specific interval *does not* contain the first return implies dependencies for all subsequent samples.

We resolve these difficulties by maintaining a new invariant. Consider all positions whose heights have have already been determined, either directly as a result of user given HEIGHT queries, or indirectly due to recursive HEIGHT calls; $\langle x_1, x_2, \cdots, x_m \rangle$ in increasing order i.e. $x_i < x_{i+1}$. Let the corresponding heights be $\langle y_1, y_2, \cdots, y_m \rangle$ i.e. HEIGHT$(x_i) = y_i$.

▶ **Invariant 30.** *For any interval $[x_i, x_{i+1}]$ where the heights of the endpoints have been determined to be y_i and y_{i+1}, and every other height in the interval has yet to determined, the section of the Dyck path between positions x_i and x_{i+1} is constrained to lie above $min(y_i, y_{i+1})$.*

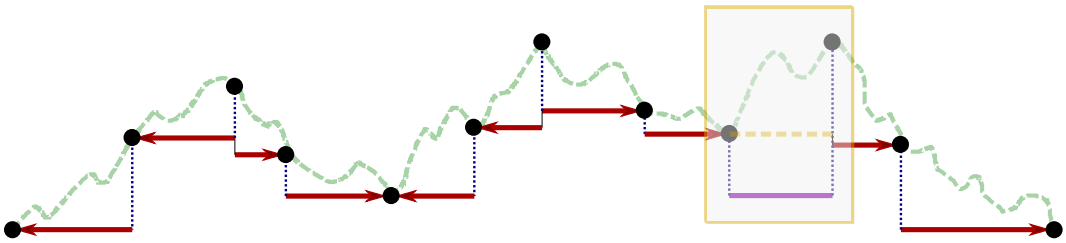


Figure 5 The set of intervals formed by a set of height samples. Each interval also has its own boundary constraint (red). Invariant 30 implies that each boundary must coincide with one of the interval endpoints. Note that the only interval which violates the invariant is the third last one (shown in yellow box).

How can one maintain such an invariant? After determining the height of a particular position x_i as y_i (with $x_{i-1} < x_i < x_{i+1}$), the invariant is potentially broken on either side of x_i. We re-establish the invariant by determining the height of an additional point on either side (see Section 4.4.6 for details). Specifically, we use the following *two step strategy* to find the additional point for some violating interval $[x_i, x_{i+1}]$ (example violation in Figure 5):

1. Sample the lowest height h achieved by the walk between x_i and x_{i+1} according to the uniform distribution over all possible paths that respect the current boundary constraint (see Section 4.4.2).
2. Find an intermediate position x such that $x_i < x < x_{i+1}$ and $\textsc{Height}(x) = h$ (see Section 4.4.3).

The newly determined position x produces two sub-intervals $[x_i, x]$ and $[x, x_{i+1}]$. Since $h = \textsc{Height}(x)$ has been determined to be the minimum height in the overall range $[x_i, x_{i+1}]$, the invariant is preserved in the new intervals (see Figure 6). Lemma 37 in Section 4.4.5 shows how this invariant can help us efficiently search for the interval containing the first return.

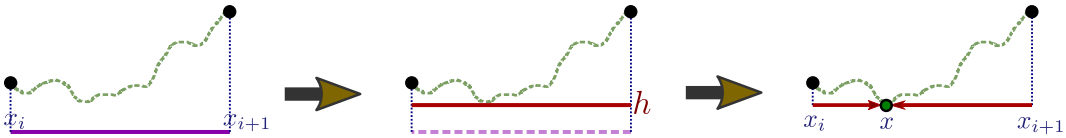


Figure 6 An interval $[x_i, x_{i+1}]$ that violates the invariant is "fixed" by first sampling the lowest height h achieved within the interval, and then generating a position $x \in [x_i, x_{i+1}]$ such that $Height(x) = h$.

4.4.2 Sampling the Lowest Achievable Height: Mandatory Boundary

First, we need to sample the lowest height h of the walk between x_i and x_{i+1} (with corresponding heights y_i and y_{i+1}). We will refer to h as the *"mandatory boundary"* in this interval; i.e. no height in the interval may be lower than the boundary, but at least one point *must* touch the boundary (have height h). We assume that $y_i \leq y_{i+1}$ without loss of generality; if this is not the case, swap x_i and x_{i+1} and consider the reversed path. Say this interval defines a generalized Dyck problem with U up steps, D down steps, and a boundary that is $k - 1$ units below y_i.

Given any two boundaries k_{lower} and k_{upper} on this interval (with $k_{lower} < k_{upper} \leq y_i$), we can count the number of possible generalized Dyck paths that violate the k_{upper} boundary but *not* the k_{lower} boundary as follows (see Figure 7):

$$P_{k_{lower}}^{k_{upper}} = C_{k_{lower}}(U, D) - C_{k_{upper}}(U, D)$$

We define the current lower and upper boundaries as $k_{low} = k, k_{up} = 0$, and set $k_{mid} =$

Algorithm 4 Finding the Mandatory boundary.

```
1:  function MANDATORY-BOUNDARY(U, D, k)
2:      k_low ← k
3:      k_up ← 0
4:      while k_low < k_up − 1
5:          k_mid ← ⌊(k_low + k_up)/2⌋
6:          P_total ← C_{k_low}(U, D) − C_{k_up}(U, D)
7:          P_{k_low}^{k_mid} ← C_{k_low}(U, D) − C_{k_mid}(U, D)
8:          with probability P_{k_low}^{k_mid} / P_total
9:              k_up ← k_mid
10:         else
11:             k_low ← k_mid
12:     return k_low
```

$(k_{low} + k_{up})/2$. Since we can compute the quantities $P_{k_{mid}}^{k_{up}}$, $P_{k_{low}}^{k_{mid}}$, and $P_{total} = P_{k_{low}}^{k_{up}}$, we can sample a single bit to decide if the *"lower boundary"* should move up or if the *"upper boundary"* should move down. We then repeat this binary search until we find $k' = k_{low} = k_{up} - 1$ and k' becomes the *"mandatory boundary"* (i.e. the walk reaches the height exactly $k' - 1$ units below y_i but no lower.

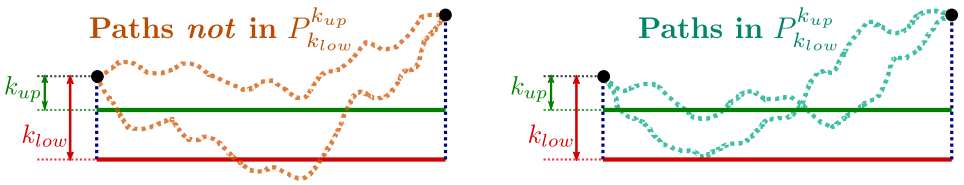


Figure 7 $P_{k_{low}}^{k_{up}}$ counts all the paths that dip at least k_{up} units below the starting point, but **do not** dip k_{low} units below the starting point. k_{mid} would lie halfway between these boundaries.

4.4.3 Determining First Position that Touches the *"Mandatory Boundary"*

Now that we have a "*mandatory boundary*" k, we just need to generate a position $x \in [x_i, x_{i+1}]$ with $\textsc{Height}(x) = y_i - k + 1$, according to the uniform distribution over all paths that touch, but do not violate the boundary at k. In fact, we will do something stronger by determining the *first* time the walk touches the boundary after x_i. As before, we assume that this interval contains U up steps and D down steps.

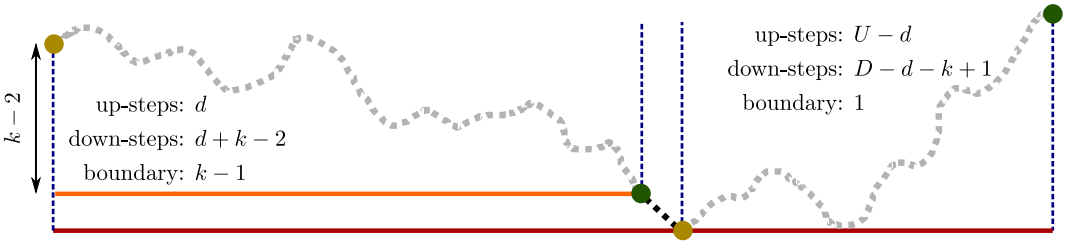


Figure 8 Zooming into the error in Figure 5. We generate a position x (yellow) on the boundary (red), such that the section of the path to the left of x never approaches the red boundary (it respects the orange boundary).

We will parameterize the position x the number of up-steps d between x_i and x (See Figure 8). implying that $x = x_i + 2d + k - 1$. Given a specific d, we want to compute the number of valid paths that result in d up-steps before the first approach to the boundary. Note that unlike Section 4.3, d is used here to parameterize the (horizontal) x-position of the desired point. We will calculate the probability p_d associated with a particular position by counting the total number of paths to the left and right of the first approach and multiplying them together.

As in Section 4.3.2, we define S_{left} to be the number of paths in the sub-interval before the first approach (left side of Figure 8), S_{right} to be the number of paths following the first approach, and S_{total} to be the total number of paths that touch the "mandatory boundary" at k (note that these quantities are functions of U, D, k and d, but we drop the parameters for the sake of clarity):

$$S_{left} = C_k(d, d+k-2) \quad S_{right} = C_1(U-d, D-d-k+1) \quad S_{total} = C_k(U, D) - C_{k-1}(U, D)$$

Our goal is to generate d from the distribution $\{p_d\}$ where $p_d = S_{left} \cdot S_{right}/S_{total}$. The application of Lemma 11 requires us to approximate $\{p_d\}$ with a "well behaved" $\{q_d\}$ (one whose CDF can be efficiently estimated). Since q_d only needs to approximate p_d up to $\text{poly}(\log n)$ factors, we focus on obtaining $\text{poly}(\log n)$ approximations to the path counts $\{S_{left}, S_{right}, S_{total}\}$. Using Equation 1, we obtain approximations for S_{left} and S_{right} (Lemma 31 and Lemma 32 with proofs in Section F.5).

▶ **Lemma 31.** *If $d > \log^4 n$, then $S_{left}(d) = \Theta\left(\frac{2^{2d+k}}{\sqrt{d}} e^{-r_{left}(d)} \cdot \frac{k-1}{d+k-1}\right)$ where $r_{left}(d) = \frac{(k-2)^2}{2(2d+k-2)}$. Furthermore, $r_{left}(d) = \mathcal{O}(\log^2 n)$.*

▶ **Lemma 32.** *If $U + D - 2d - k > \log^4 n$, then $S_{right}(d) = \Theta\left(\frac{2^{U+D-2d-k}}{\sqrt{U+d-2d-k}} e^{-r_{right}(d)} \cdot \frac{U-D+k}{U-d+1}\right)$ where $r_{right}(d) = \frac{(U-D-k-1)^2}{4(U+D-2d-k+1)}$. Furthermore, $r_{right}(d) = \mathcal{O}(\log^2 n)$.*

Our general strategy will be to integrate continuous versions of these approximations in order to obtain a CDF of some approximating distribution. Unfortunately, the continuous approximation functions obtained do not admit closed form integrals. The main culprit is the exponential term in both expressions. We tackle this issue by noticing that the values of the exponents are bounded by $\mathcal{O}(\log^2 n)$ over the majority of the range of d. Within this range of d values, the exponential term may be further simplified by taking a piecewise constant approximation, where each of the pieces corresponds to a fixed value of the *floor of the corresponding exponent*. This technique is elaborated in Section 4.4.4.

We start by considering the *"problematic"* values of d that are outside the range of the two preceding lemmas. These values are the ones where $d < \log^4 n$ or $2d > U + D - k - \log^4 n$. Since d is the number of up steps in the left sub-interval, $d \geq 0$. Further, since the length of the right sub-interval nust be non-negative (see Figure 8), we get $U + D - 2d - k + 1 \geq 0$. Thus, we define the *"problematic"* set:

$$\mathcal{R} = \left\{ d \;\middle|\; 0 \leq d < \log^4 n \textbf{ or } -1 < 2d - U - D + k < \log^4 n \right\} \tag{4}$$

Clearly, we can bound the size of this set as $|\mathcal{R}| = \mathcal{O}(\log^4 n)$. An immediate consequence of Lemma 31 and Lemma 32 is the following.

▶ **Corollary 33.** *When $d \notin \mathcal{R}$, $S_{left}(d) \cdot S_{right}(d) = \Theta\left(\frac{2^{U+D}}{\sqrt{d(U+D-2d-k)}} \cdot e^{-r(d)} \cdot \frac{k-1}{d+k-1} \cdot \frac{U-D+k}{U-d+1}\right)$ where $r(d) = \mathcal{O}(\log^2 n)$.*

4.4.4 Estimating the CDF

We use these observations to construct a suitable $\{q_d\}$ that can be used to invoke the rejection sampling lemma (Lemma 11). We will achieve this by constructing a piecewise continuous function $\hat{q}$, such that $\hat{q}(\delta)$ approximates $p_{\lfloor \delta \rfloor}$, and then use the integral of $\hat{q}$ to define the discrete distribution $\{q_d\}$. As stated in the previous section, we can leverage the fact that when $d \notin \mathcal{R}$, the floor of the exponent $\lfloor r(d) \rfloor$ only takes $\mathcal{O}(\log^2 n)$ distinct values (consequence of Corollary 33). Since the "problematic" set $\mathcal{R}$ only has $\mathcal{O}(\log^4 n)$ values, we can also deal with these remaining values by simply creating $|\mathcal{R}|$ additional continuous pieces in the function $\hat{q}$. We begin by rewriting $p_d = \Theta\left(\mathcal{K} \cdot f(d) \cdot e^{-r(d)}\right)$ where:

$$\mathcal{K} = \frac{2^{U+D}}{S_{total}} = \frac{2^{U+D}}{C_k(U,D) - C_{k-1}(U,D)} \qquad f(d) = \frac{(k-1)(U-D+k)}{\sqrt{d(U+D-2d-k)}(d+k-1)(U-d+1)} \tag{5}$$

Notice that $\mathcal{K}$ is a constant and $f(d)$ is a function whose integral has a closed form. Using the fact that $r(d) = \mathcal{O}(\log^2 n)$ (Corollary 33), and $|\mathcal{R}| = \mathcal{O}(\log^4 n)$, we obtain the following lemma:

▶ **Lemma 34.** *Given the piecewise continuous function*

$$\hat{q}(\delta) = \begin{cases} p_{\lfloor \delta \rfloor} & \text{if } \lfloor \delta \rfloor \in \mathcal{R} \\ \mathcal{K} \cdot f(\delta) \cdot exp\left(-\lfloor r(\lfloor \delta \rfloor) \rfloor\right) & \text{if } \lfloor \delta \rfloor \notin \mathcal{R} \end{cases} \qquad \Longrightarrow \qquad p_d = \Theta\left(\int_d^{d+1} \hat{q}(\delta)\right)$$

Furthermore, $\hat{q}(\delta)$ has $\mathcal{O}(\log^4 n)$ continuous pieces.

Proof. For $d \in \mathcal{R}$, the integral trivially evaluates to exactly p_d. For $d \notin \mathcal{R}$, it suffices to show that $p_d = \Theta\left(\hat{q}(\delta)\right)$ for all $\delta \in [d, d+1)$. We already know that $p_d = \Theta\left(\mathcal{K} \cdot f(d) \cdot e^{-r(d)}\right)$. Moreover, for any $\delta \in [d, d+1)$, the exponential term $e^{-\lfloor r(\lfloor \delta \rfloor) \rfloor}$ in $\hat{q}(\delta)$ is within a factor of e of the original $e^{-r(\lfloor \delta \rfloor)}$ term.

For all $\mathcal{O}(\log^4 n)$ values $d \in \mathcal{R}$, $\hat{q}(\delta)$ is constant on the interval $[d, d+1]$. Since $r(d) = \mathcal{O}(\log^2 n)$ by Corollary 33, the exponential term $e^{-\lfloor r(\lfloor \delta \rfloor) \rfloor}$ in $\hat{q}(\delta)$ taken on at most $\mathcal{O}(\log^2 n)$ values. Thus, $\hat{q}$ is continuous for a range of δ correspoding to a fixed value of $\lfloor r(\lfloor \delta \rfloor) \rfloor$, and so, we conclude that $\hat{q}$ is piecewise continuous with $\mathcal{O}(\log^2 n)$ pieces. ◀

Now, we have everything in place to define the distribution $\{q_d\}$ that we will be sampling from. Specifically, we will define q_d and it's CDF Q_d as follows ($\mathcal{N}$ is the normalizing factor):

$$q_d = \left(\int_d^{d+1} \hat{q}(\delta)\right) \cdot \frac{1}{\mathcal{N}} \qquad Q_d = \left(\int_0^{d+1} \hat{q}(\delta)\right) \cdot \frac{1}{\mathcal{N}} \qquad \text{where } \mathcal{N} = \int_0^{d_{max}+1} \hat{q}(\delta) \tag{6}$$

To show that these can be computed efficiently, it suffices to show that any integral of $\hat{q}(\delta)$ can be efficiently evaluated. This follows from the fact that $\hat{q}$ is piecewise continuous with $\mathcal{O}(\log^4)$ pieces (Lemma 34), each of which has a closed form integral (since $f(d)$ defined in Equation 5 has an integral).

▶ **Lemma 35.** *Given the function $\hat{q}_d$ defined in Lemma 34, it is possible to approximate the integral $\int_{d_1}^{d_2+1} \hat{q}(\delta)$ to a multiplicative factor of $\left(1 \pm \frac{1}{n^2}\right)$, in $\mathrm{poly}(\log n)$ time for any valid $d_1, d_2 \in \mathbb{Z}$ (the bounds must be such that $d_i \geq 0$ and $U + D - 2d_i - k + 1 \geq 0$).*

Proof. We will compute the integral by splitting it up into $\mathcal{O}(\log^4 n)$ continuous pieces and then approximating the integral over each piece. The pieces corresponding to values of $d \in \mathcal{R}$ can be explicitly computed as $\int_d^{d+1} \hat{q}(\delta) = p_d$ (recall that we can compute p_d using Lemma 67 from Section F.3).

The more interesting pieces are over the range of δ where $\lfloor \delta \rfloor \notin \mathcal{R}$. Such a piece is given by a continuous range of values $[\delta_{min}, \delta_{max}] \subseteq [d_1, d_2 + 1]$, such that for any $\delta \in [\delta_{min}, \delta_{max}]$, the value of $\lfloor r(\lfloor \delta \rfloor) \rfloor$ is a constant E. We begin the calculation of $\hat{q}(\delta) = \mathcal{K} \cdot f(\delta) \cdot exp(-E)$, by first computing a $\left(1 \pm 1/n^3\right)$ multiplicative approximation to $\ln \mathcal{K} = (U + D) \ln 2 - \ln S_{total}$, using the strategy in Lemma 67 (Section F.3). Since $f(\delta)$ has a closed form integral (Equation 5), we can also compute $F = \int_{\delta_{min}}^{\delta_{max}} f(\delta)$. Using the fact that the exponent is constant over the range $[\delta_{min}, \delta_{max}]$, we can write the logarithm of the integral as:

$$\ln \left(\int_{d_{min}}^{d_{max}} \hat{q}(\delta) \right) = \ln \left(\mathcal{K} \cdot e^{-E} \cdot F \cdot \right) = (U + D) \cdot \ln 2 - \ln S_{total} + \ln F - E$$

If this value is smaller than $-3 \ln n$, we can safely ignore it since it contrbuted less than $1/n^3$ to the probability mass. On the flipside, the logarithm is guaranteed to be bounded by $\mathcal{O}(1)$. This upper bound is a result of the fact that $\int \hat{q}(\delta) = \mathcal{O}(\sum p_d) = \mathcal{O}(1)$, where both the sum and the integral are taken over the entire *valid* range of d. This means that we can exponentiate to obtain the true value of the piecewise integral up to a multiplicative approximation of $(1 \pm 1/n^3)$. Adding the integrals of all the $\mathcal{O}(\log^4 n)$ pieces together produces the final desired value of the integral. ◀

Now, we are finally ready to use Lemma 11 to sample d from the distribution $\{p_d\}$, using the efficient sampling procedure for $\{q_d\}$. The only other requirement is the ability to approximate the p_d values, which follows from Lemma 67.

▶ **Theorem 36.** *Given a sub-interval $[x_i, x_{i+1}]$ of a random Dyck path of length $2n$, such that the only determined heights in the interval are $y_i = \textsc{Height}(x_i)$ and $y_{i+1} = \textsc{Height}(x_{i+1})$, and a mandatory boundary constraint at k, there exists an algorithm that generates a point x within the interval such that $\textsc{Height}(x) = y_i - k + 1$, using $\mathrm{poly}(\log n)$ time, random bits, and additional space.*

4.4.5 Finding the Correct Interval: `First-Return` Query

As before, consider all positions that have been queried already $\langle x_1, x_2, \cdots, x_m \rangle$ (in increasing order) along with their corresponding heights $\langle y_1, y_2, \cdots, y_m \rangle$. We wish to find the first return to height y_i after x_i (where $y_i = \textsc{Height}(x_i)$). Our strategy begins by using Invariant 30 to find the interval $\mathcal{I} = [x_k, x_{k+1}]$ containing $\textsc{First-Return}(x_i)$.

▶ **Lemma 37.** *Assuming that Invariant 30 holds, the interval $(x_{j-1}, x_j]$ containing $\textsc{First-Return}(x_i)$ is obtained by setting j to be either the smallest index $f > i$ such that $y_f \leq y_i$ or setting $j - 1 = i$.*

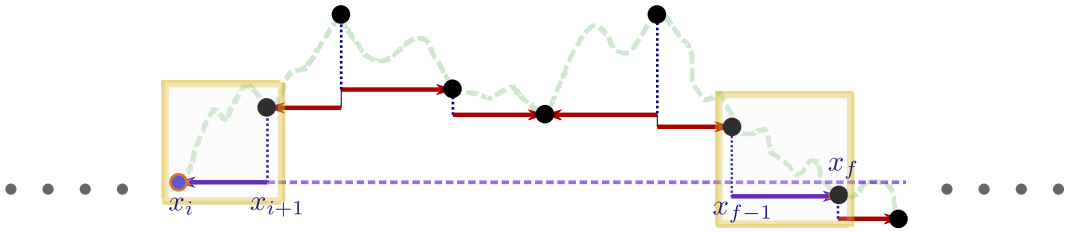


Figure 9 There are only two possible intervals (yellow boxes) that could contain $\textsc{First-Return}(x_i)$; either the interval adjacent to x_i, or the interval $[x_{f-1}, x_f]$, where f is the smallest index such that $y_f < x_i$.

Proof. We assume the contrary i.e. there exists some $k \neq f$ and $k \neq i+1$ such that the correct interval is $(x_{k-1}, x_k]$. Since $y_f < y_i$, the position of first return to y_i happens in the range $(x_i, x_f]$. So, the only possibility is $i+1 < k \leq f-1$. By the definition of y_f, we know that both y_k and y_{k-1} are strictly larger than y_i. Invariant 30 implies that the boundary for this interval $(y_{k-1}, y_k]$ is at $\min(y_{k-1}, y_k) > y_i$. So, it is not possible for the first return to be in this interval. ◀

The good news is that there are only two intervals that we need to worry about, one of which is just the adjacent one $[x_i, x_{i+1}]$. The problem of finding the other interval that may contain the first return boils down to finding the smallest index $f > i$ such that $y_f \leq y_i$. To this end, we define $M_{[a,b]}$ as the minimum determined height in the range of positions $[a, b]$.

One solution is to maintain a range tree $\mathbf{R}$ [14] over the range $[2n]$. Assuming that $2n = 2^l$, we can view $\mathbf{R}$ as a complete binary tree with depth r. Every non-leaf node is denoted by $\mathbf{R}_{[a,b]}$, and corresponds to a range $[a, b] \subseteq [2n]$ that is the union of the ranges of its children. Each $\mathbf{R}_{[a,b]}$ stores the value $M_{[a,b]}$ which is is the minimum determined height in the range of positions $[a, b]$, or ∞ if none of the heights have been revealed. The leaf nodes are denoted as $\{\mathbf{R}_i\}_{i\in[2n]}$, and correspond to the singleton range corresponding to position $i \in [2n]$. Note that a node at depth l' will correspond to a range of size $2^{l-l'}$, with the root being associated with the entire range $[2n]$.

We say that the range $[a, b]$ is *canonical* if it corresponds to a range of some $\mathbf{R}_{[a,b]}$ in $\mathbf{R}$. By the property of range trees, any arbitrary range can be decomposed into a disjoint union of $O(\log n)$ canonical ranges. We implement $\mathbf{R}$ to support the following operations:

- $\textsc{Update}(x, y)$: Update the height of the position x to y.
  This update affects all ranges $[a_i, b_i]$ containing x. So, for each $[a_i, b_i]$ we set $M_{[a_i,b_i]} = \min\left(M_{[a_i,b_i]}, y\right)$.
- $\textsc{Query}(a, b)$: Return the minimum boundary height in the range $[a, b]$.
  We decompose $[a, b]$ into $\mathcal{O}(\log n)$ *canonical* ranges $\langle r_1, r_2, \cdots\rangle$, and return the minimum of all the M_{r_i} values as $M_{[a,b]}$ (since $[a, b]$ is the union of all r_i).

Now, we can binary search for f by guessing a value f' and checking if $\textsc{Query}(x_i, x_{f'}) \leq y_i$. Overall, this requires $\mathcal{O}(\log n)$ calls to $\textsc{Query}$, each of which makes $\mathcal{O}(\log n)$ probes to the range tree. To avoid an initialization overhead, we only create the node $\mathbf{R}_{[a,b]}$ during the first $\textsc{Update}$ affecting a position $x \in [a, b]$. Since a call to $\textsc{Update}$ can create at most $\mathcal{O}(\log n)$ new nodes in $\mathbf{R}$, the additional space required for each $\textsc{Height}$ or $\textsc{First-Return}$ query is still bounded.

▶ **Theorem 38.** *There is an algorithm using $\mathcal{O}(\log^{\mathcal{O}(1)} n)$ resources per query that provides access to a random Dyck path of length $2n$, by answering queries of the form $\textsc{First-Return}(x_i)$ with the correctly sampled smallest position $x' > x_i$, where the Dyck path first returns to $\textsc{Height}(x_i)$.*

Proof. In order to make the presentation simpler, we ensure that the next determined position after x_i is x_i+1 (in other words $x_{i+1} = x_i+1$). This can be done by invoking $\textsc{Height}(x_i+1)$, if needed. If $\textsc{Height}(x_i+1) = y_{i+1} < y_i$, we can terminate because $\textsc{First-Return}(x_i)$ is not defined. Otherwise, we notice that in this setting, the first return cannot lie in the adjacent interval $[x_i, x_{i+1}] = [x_i, x_i+1]$.

Hence, we proceed to finding the smallest value f such that $y_f \le y_i$, by using the range tree data structure described above. Since $\textsc{Height}(x_{f-1}) > y_i \le \textsc{Height}(x_f)$ by definition, the interval $(x_{f-1}, x_f]$ must contain at least one position at height y_i. We determine the height at the midpoint of this interval, and then fix the potential violation of Invariant 30 by finding another point along with its height. This essentially breaks up the interval $[x_{f-1}, x_f]$ into $\mathcal{O}(1)$ sub-intervals, each at most half the size of the original. Based on the newly revealed heights, we again find the (newly created) sub-interval containing the first return in $\mathcal{O}(1)$ time. We repeat up to $\mathcal{O}(\log n)$ times, reducing the size of the intervals in consideration, until the position of the first return is revealed. ◀

4.4.6 Maintaining `Height` Queries under Invariant 30

Finally, we show that the boundary constraints introduced in order to maintain Invariant 30 do not interfere with the implementation of $\textsc{Height}$ queries. As before, we consider the currently revealed heights $\langle y_1, y_2, \cdots, y_m\rangle$, along with the corresponding positions $\langle x_1, x_2, \cdots, x_m\rangle$ (in increasing order). Say that we are now presented with a query $\textsc{Height}(x)$, where $x_i < x < x_{i+1}$. As in Section 4.3, we swap x_i and x_{i+1} if necessary in order to ensure that $y_i < y_{i+1}$. Due to Invariant 30, we know that the lowest achievable height in the interval $[x_i, x_{i+1}]$ is y_i, i.e. the boundary constraint for the left half becomes $k = 1$ instead of $k = y_i + 1$, since the constrained boundary is at height y_i. Similarly, the boundary constraint for the right half becomes $k' = 2U - 2D + 1$.

The algorithm for determining the height at the midpoint x_{mid} of $[x_i, x_{i+1}]$ can proceed as described in Section 4.3. Of course, in this scenario, the boundary is never far away, and therefore we should always use the strategy in Section 4.3.2. The second step is to re-establish Invariant 30 in the newly created sub-interval $[x_{mid}, x_{i+1}]$[4], using the two step strategy from Section 4.4.1. This involves determining the height of one additional point $x' \in [x_{mid}, x_{i+1}]$. Finally, we can continue with the height sampling algorithm from Theorem 29, by recursively halving one of the newly created intervals (the one containing x). Figure 10 illustrates the aforementioned three steps.

Hence, we can combine results from Theorem 29 and Theorem 38 to obtain the following:

▶ **Theorem 6.** *There is an algorithm using $\mathcal{O}(\log^{\mathcal{O}(1)} n)$ resources per query that provides access to a uniformly random Dyck path of length $2n$ by implementing the following queries:*

- $\textsc{Height}(t)$ *returns the position of the walk after t steps.*
- $\textsc{First-Return}(t)$*: If* $\textsc{Height}(t+1) > \textsc{Height}(t)$*, then* $\textsc{First-Return}(t)$ *returns the smallest index t', such that $t' > t$ and* $\textsc{Height}(t') = \textsc{Height}(t)$ *(i.e. the first time the Dyck path returns to the same height). Otherwise, if* $\textsc{Height}(t+1) < \textsc{Height}(t)$*, then* $\textsc{First-Return}(t)$ *is not defined.*

[4] Note that the invariant is not broken for the $[x_i, x_{mid}]$ sub-interval since $y_i + 1$ was the original boundary constraint. In general, the invariant will be automatically preserved for one of the sub-intervals; the side corresponding to $\min(y_i, y_{i+1})$.

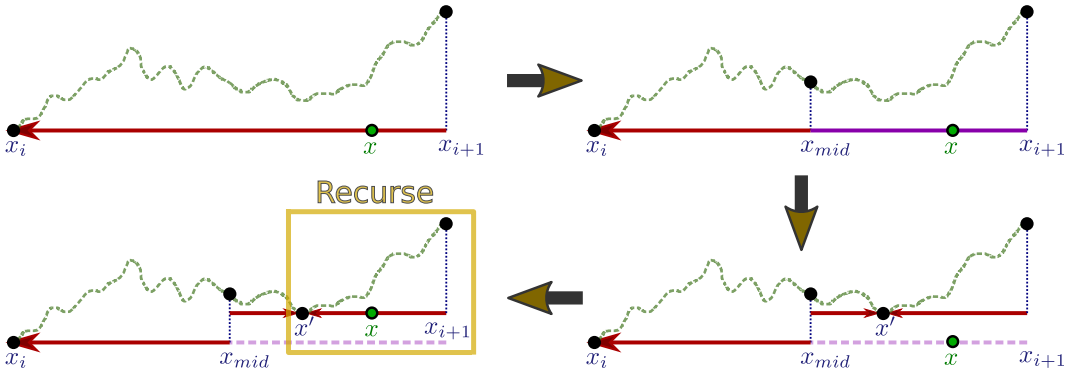


Figure 10 Performing HEIGHT queries while maintaining Invariant 30. The first step is to determine the height at the midpoint x_{mid} of an existing interval $[x_i, x_{i+1}]$ that contains x. Next, we re-establish Invariant 30 by sampling an additional point x' within the violating interval Finally, we recurse on the appropriate sub-interval (yellow box).

4.4.7 Reverse-First-Return Queries

For the sake of completeness, we also sketch how to implement a REVERSE-FIRST-RETURN query, which is just a standard FIRST-RETURN query on the reversed Dyck path. This type of query was motivated in Section 4.1, and can be applied to positions x where $\textsc{Height}(x-1) = \textsc{Height}(x) + 1$.

Note that Invariant 30 remains unchanged as it is symmetric to reversal of the Dyck path. Specifically, the invariant on a particular interval does not change if we swap the endpoints of the interval. First, we can find the correct interval containing $\textsc{Reverse-First-Return}(x)$, by using an analog of Lemma 37.

▶ **Lemma 39.** *Assuming Invariant 30, the interval* $[x_j, x_{j+1})$ *containing* $\textsc{Reverse-First-Return}(x_i)$ *is obtained by setting* j *to be either the largest index* $f < i$ *such that* $y_f \leq y_i$ *or setting* $j + 1 = i$.

Since the range tree defined in Section 4.4.5 allows us to query the minimum in arbitrary contiguous ranges, we can also use it to find the appropriate f for Lemma 39 through binary search. Once we have found an interval that is known to contain the REVERSE-FIRST-RETURN, we can follow the strategy from the proof of Theorem 38; recursively sub-dividing the relevant interval and finding the newly created sub-interval that contains the return, until we converge on the correct return position.

5 Random Coloring of a Graph

A *valid* q-coloring of a graph $G = (V, E)$ is a vector of colors $\mathbf{X} \in [q]^V$, such that for all $(u, v) \in E$, $\mathbf{X}_u \neq \mathbf{X}_v$. We present a sub-linear time algorithm to provide local access to a uniformly random valid q-coloring of an input graph. Specifically, we implement $\textsc{Color}(v)$, which returns the color $\mathbf{X}_v$ of node v, where $\mathbf{X}$ is a uniformly random valid coloring. The implementation can access the input graph G through a sub-linear number of *neighborhood queries*. A neighborhood query of the form $\textsc{All-Neighbors}(v)$ returns a list of neighbors of v. The implementation can also access a tape of public random bits $\mathbf{R}$.

Moreover, multiple independent instances of COLOR that are given access to the same public tape of random bits $\mathbf{R}$, should output color values consistent with a single $\mathbf{X}$, regardless of the order and content of the queries received. Unlike our previous results, the choice of

$\mathbf{X}$ only depends on $\mathbf{R}$, and the COLOR implementations do not need to use any additional memory to maintain consistency. For a formal description of this model, see Definition 10. This model is essentially a generalization of *Local Computation Algorithms* [48]. Given a computation problem with input x and a set of valid outputs $F(x)$, LCAs provide query access to some $y \in F(x)$ using only a sub-linear number of probes to x. This makes it feasible to consider sub-linear algorithms for problems where the output size is super-linear. Our model has the additional restriction that the output must be drawn uniformly at random from $F(x)$, rather than just being an arbitrary member. Since the number of possible outputs ($|F(x)|$) may be exponential, it is not possible to encode all the requisite random bits in sub-linear space. Therefore, we use a source of public randomness $\mathbf{R}$.

Sequential Algorithm for Random Coloring

We consider graphs with max degree Δ, and $q = \Theta(\Delta)$, since this is the regime where this problem is feasible [22]. In the sequential setting, [22] used the technique of path coupling to show that for $q > 2\Delta$, one can sample an uniformly random coloring by using a simple Markov chain. The Markov chain proceeds in T steps. The state of the chain at time t is given by $\mathbf{X}^t \in [q]^{|V|}$. Specifically, the color of vertex v at step t is $\mathbf{X}^t_v$. In each step of the Markov process, a vertex and a color are sampled uniformly at random i.e. a pair $(v, c) \sim_{\mathcal{U}} V \times [q]$. Subsequently, if the recoloring of vertex v with color c does not result in a conflict with v's neighbors, i.e. $c \notin \{\mathbf{X}^t_u : u \in \Gamma(v)\}$, then the vertex is recolored i.e. $\mathbf{X}^{t+1}_v \leftarrow c$. After running this chain for $T = \mathcal{O}(n \log(n/\epsilon))$ steps, the Markov chain is mixed, implying that the distribution of resulting colors is ϵ close to the uniform distribution in L_1 distance.

5.1 Modified Glauber Dynamics based on a Distributed Algorithm

Now we define a modified Markov chain as a special case of the *Local Glauber Dynamics* presented in [20]. The modified Markov chain proceeds in epochs. We denote the initial coloring of the graph by the vector $\mathbf{X}^0$ and the state of the coloring after the k^{th} epoch by $\mathbf{X}^k$. In the k^{th} epoch, every node attempts to recolor itself simultaneously in a conservative manner, as described below:

- Sample $|V|$ colors $\langle c_1, c_2, \cdots, c_n \rangle$ from $[q]$, where c_v is the color proposed by vertex v.
- For each vertex v, we set $\mathbf{X}^k_v$ to c_v, if and only if for all neighbors w of v, $\mathbf{X}^{k-1}_w \neq c_v$ and $c_w \neq c_v$. Specifically, a vertex v is recolored if and only if its proposed color c_v does not conflict with any of it's neighbors current colors (at the end of the previous epoch), or their current proposals.

This procedure is a special case of the *Local Glauber Dynamics*, which was presented in [20] as a distributed algorithm for sampling a random coloring.[5] In the distributed setting, our epochs correspond to synchronous rounds, where many vertices recolor themselves simultaneously.

In order to bound the mixing time of this Markov chain, [20] uses the standard technique of *path coupling*, introduced in [8]. The argument begins by considering two initial states of the Markov Chains, say two colorings $\mathbf{X}^0$ and $\mathbf{Y}^0$, that differ at only one vertex. Formally, we can define the distance between two colorings $d(\mathbf{X}, \mathbf{Y})$ as the number of vertices v such

[5] Note that [20] also uses a marking probability γ, which indicates the likelihood of any vertex participating in a given round. For our purposes, it suffices to set $\gamma = 1$.

that $\mathbf{X}_v \neq \mathbf{Y}_v$, which results in the condition $d(\mathbf{X}^0, \mathbf{Y}^0) = 1$. A *coupling* is a joint evolution rule for a pair of states $(\mathbf{X}^0, \mathbf{Y}^0) \to (\mathbf{X}^1, \mathbf{Y}^1)$, such that both of the individual evolutions $(\mathbf{X}^0 \to \mathbf{X}^1)$ and $(\mathbf{Y}^0 \to \mathbf{Y}^1)$ have the same transition probabilities as the original Markov Chain. We can directly use the result from the coupling defined in [20].

▶ **Lemma 40.** *If $q = 2\alpha\Delta$, then there exists a coupling $(\mathbf{X}^0, \mathbf{Y}^0) \to (\mathbf{X}^1, \mathbf{Y}^1)$, such that if $d(\mathbf{X}^0, \mathbf{Y}^0) = 1$, then $\mathbb{E}[d(\mathbf{X}^1, \mathbf{Y}^1)] \leq 1 - \left(1 - \frac{1}{2\alpha}\right) e^{-3/\alpha} + \frac{1/2\alpha}{1-1/\alpha}$*

▶ **Corollary 41.** *If $q \geq 9\Delta$ and $d(\mathbf{X}^0, \mathbf{Y}^0) = 1$, then $\mathbb{E}[d(\mathbf{X}^1, \mathbf{Y}^1)] < \frac{1}{e^{1/3}}$*

The *path coupling* lemma from [8] uses a coupling on adjacent states to bound the mixing time.

▶ **Lemma 42** (Simplified Path Coupling from [8])**.** *If there exists a coupling $(\mathbf{X}^0, \mathbf{Y}^0) \to (\mathbf{X}^1, \mathbf{Y}^1)$ defined for states where $d(\mathbf{X}^0, \mathbf{Y}^0) = 1$, such that $\mathbb{E}[d(\mathbf{X}^1, \mathbf{Y}^1) \mid \mathbf{X}^0, \mathbf{Y}^0] \leq \beta$ (for $\beta < 1$), then, the mixing time $\tau_{mix}(\epsilon) = \mathcal{O}(\ln(n\epsilon^{-1})/\ln \beta^{-1})$.*

▶ **Corollary 43.** *If $q \geq 9\Delta$, then the chain is mixed after $\tau_{mix}(\epsilon) = 3\left(\ln n + \ln(\frac{1}{\epsilon})\right)$ epochs.*

5.1.1 Naive Reduction from Distributed Algorithms

Using the technique of Parnas and Ron [46], one can modify a distributed algorithm for a graph problem to construct a Local Computation Algorithm for the same problem. Specifically, given a k-round distributed algorithm on a network of max degree Δ, we can simulate the behaviour and outcome of a single node v by simulating the full algorithm on the k-neighborhood of v. The simulation only requires us to probe this k-neighborhood, which contains $\mathcal{O}(\Delta^k)$ nodes. However, the aforementioned distributed algorithm for Glauber Dynamics used $\mathcal{O}(\log n)$ rounds, implying a probe complexity of $\Delta^{\mathcal{O}(\log n)}$, which is super-linear. We show how to reduce the probe complexity by appropriately pruning the k-neighborhood.

5.2 Local Coloring Algorithm

Given query access to the adjacency matrix of a graph G with maximum degree Δ and a vertex v, the algorithm has to output the color assigned to v after running $t = \mathcal{O}(\ln n)$ epochs of *Modified Glauber Dynamics*. We want to be able to answer such queries in sub-linear time, without simulating the entire Markov Chain. We will define the number of colors as $q = 2\alpha\Delta$ where $\alpha > 1$.

The proposals at each epoch are a vector of color samples $\mathbf{C}^t \sim_{\mathcal{U}} [q]^n$, where $\mathbf{C}^t_v$ is the color proposed by v in the t^{th} epoch. Since each of these $\mathbf{C}^t_v$ values are independent uniform samples from $[q]$, instances of our algorithm will be able to access them in a consistent manner using the public random bits $\mathbf{R}$.

We also use $\mathbf{X}^t$ to denote the final vector of vertex colors at the end of the t^{th} epoch. Finally, we define indicator variables $\boldsymbol{\chi}^t_v$ to indicate whether the color $\mathbf{C}^t_v$ proposed by vertex v was accepted at the t^{th} epoch: $\boldsymbol{\chi}^t_v = 1$ if and only if for all neighbors $w \in \Gamma(v)$, we satisfy the condition $\mathbf{C}^t_v \neq \mathbf{X}^{t-1}_w$ and $\mathbf{C}^t_v \neq \mathbf{C}^t_w$ (i.e. the proposed color does not conflict with any neighboring proposal or any neighbor's color from the preceding epoch). So, the color of a vertex v after the t^{th} epoch $\mathbf{X}^t_v$ is set to be $\mathbf{C}^i_v$ where $i \leq t$ is the largest index such that $\boldsymbol{\chi}^i_v = 1$. While the proposals $\mathbf{C}^t_v$ can simply be read off the public random tape $\mathbf{R}$, it is not clear how we can determine the $\boldsymbol{\chi}^t_v$ values efficiently. Computing $\mathbf{X}^t_v$ is quite simple if we know the values $\boldsymbol{\chi}^i_v$ for all $i \leq t$. So, we focus our attention on the query $\textsc{Accepted}(v, t)$ that returns $\boldsymbol{\chi}^t_v$.

5.2.1 Local Access to an Initial Valid Coloring

One caveat that we have not addressed is how we should initialize the Markov Chain. The starting state can be any valid coloring of G, but we have to be able to access the initial colors of requisite vertices in sub-linear time. One option is to simply assume that we have oracle access to an arbitrary valid coloring. We can also invoke a result from [10] that provides a *Local Computation Algorithm* for $(\Delta+1)$-coloring of a graph. Specifically, they provide local access to the color of any vertex using $\mathcal{O}(\Delta^{\mathcal{O}(1)}\log n)$ queries to the underlying graph, such that the returned colors are consistent with some valid coloring. Integrating their routine into our algorithm incurs a multiplicative poly(Δ) overhead for each query.

Another option is to (uniformly) randomly intialize the colors of the vertex independently of its neighbors. Although the initial coloring may be invalid, one can show that with high probability, the final coloring after running the Markov Chain is valid. The intuitive reasoning is that each vertex attempts to recolor itself $\mathcal{O}(\log n)$ times, and each attempt suceeds with constant probabiltity at least $1-1/\alpha$ (Lemma 44). Furthermore, we can union bound to claim that *all* the vertices get re-colored at least once with high probability. After a vertex is re-colored once, it can no longer be in conflict with any of its neighbor's colors, and therefore we obtain a final coloring that is valid. For the sake of simplicity, we will assume that our algorithm can access the initial colors of any vertex v through a public $\mathbf{C}_v^0$.

5.2.2 Naive Coloring Implementations

Our general strategy to determine $\boldsymbol{\chi}_v^t$ will be to check for all neighbors w of v, whether w causes a conflict with v's proposed color in the t^{th} epoch. One naive way to achieve this, is to iterate backwards from epoch t, querying to find out whether w's proposal was accepted, until the most recent accepted proposal (latest epoch $t' < t$ such that $\boldsymbol{\chi}_w^{t'} = 1$) is found. At this point, if $\mathbf{C}_w^{t'} = \mathbf{C}_v^t$, then the current color of w conflicts with v's proposal. Otherwise there is no conflict, and we can proceed to the next neighbor. However, this process potentially makes Δ recursive calls to a sub-problem that is only slightly smaller i.e. $T(t) \leq \Delta \cdot T(t-1)$. This leads to a running time upper bound of Δ^t which is superlinear for the desired number of epochs $t = \Omega(\log n)$ (the mixing time).

We can prune the number of recursive calls by only processing the neighbors w which actually proposed the color $\mathbf{C}_v^t$ during *some* epoch. In this case, the expected number of neighbors that have to be probed recursively is less than $t\Delta/q$ (since the total number of neighbor proposals over t epochs is at most $t\Delta$, and there are q possible colors). So, the overall runtime is upper bounded by $(t\Delta/q)^t$. For this algorithm, if we allow $q > t\Delta = \Omega(\Delta \log n)$ colors, the runtime becomes sub-linear. So, we can use this simple algorithm only when q is sufficiently large. However, we want a sub-linear time algorithm for $q = \mathcal{O}(\Delta)$.

5.2.3 A Sub-linear Time Algorithm for $q = \mathcal{O}(\Delta)$

The expected number of neighbors that need to be checked recursively can always be $t\Delta/q$ in the worst case. The crucial observation is that even though these recursive calls seem unavoidable, we can aim to reduce the size of the recursive sub-problem, and thus bound the number of levels of recursion.

Algorithm 5 shows our final procedure for generating $\boldsymbol{\chi}_v^t$, where $c = \mathbf{C}_v^t$ is v's proposal at epoch t. We iterate through all neighbors w of v, checking for conflicts (Line 3). The condition $c \neq \mathbf{C}_w^t$ can be checked by reading $\mathbf{C}_w^t$ off the public random tape (Line 4). If no conflict is seen, we proceed to check whether $c \neq \mathbf{X}_w^{t-1}$.

Algorithm 5 Checking if proposal is accepted.

```
1:  procedure ACCEPTED(v, t)
2:      c ← C_v^t                          ▷ Find the current proposal using the public random bits
3:      for w ← Γ(v)                       ▷ Iterate through the neighbors, checking for conflicts
4:          if C_w^t = c                   ▷ Check for conflict with neighbor's current proposal
5:              return 0
6:          for t' ← [t, t−1, t−2, ··· , 1, 0]   ▷ Check for conflict with neighbor's prior state
7:              if C_w^{t'} = c and ACCEPTED(w, t')   ▷ Potential conflict with neighbor's color
8:                  overwritten ← false    ▷ Check if color c is overwritten by a future proposal
9:                  for t̃ ← [t'+1, t'+2, t'+3, ··· , t−1]
10:                     if ACCEPTED(w, t̃)
11:                         overwritten ← true
12:                         break
13:                 if not overwritten
14:                     return 0           ▷ Conflict! This proposal is not accepted
15:                 break
16:     return 1                           ▷ No conflicts! This proposal is accepted
```

To achieve this, we iterate through all the epochs in reverse order (line 6) to check whether the color c was ever proposed for vertex w. If not, we can ignore w. Otherwise let's say that the most recent proposal for c was at epoch t' i.e. $\mathbf{C}_w^{t'} = c$. Now, we directly "jump" to the $(t')^{th}$ epoch and recursively check if this proposal was accepted (line 7). If the proposal $\mathbf{C}_w^{t'}$ was not accepted, we keep iterating back in time until we find the next most recent epoch when c was proposed by w, or until we run out of epochs. When we find the most recent epoch t' in which c was accepted i.e. $\boldsymbol{\chi}_w^{t'} = 1$, we successively consider epochs $t'+1, t'+2, t'+3, \cdots, t-1$, to see whether, the color c was overwritten (line 9) by an accepted proposal in a future epoch. This is done by recursively invoking $\textsc{Accepted}(w, t'+i)$ in order to compute $\boldsymbol{\chi}_w^{t'+i}$ (line 10). If at any of these subsequent iterations, we see that a different proposal was accepted (thus overwriting the color c), then neighbor w does not cause a conflict, and we can move on to the next neighbor. Otherwise, we have seen that $\boldsymbol{\chi}_w^{t'} = 1$ (color c was accepted) and every subsequent proposal until the current epoch t was rejected; this implies that color c *survived* as the color of neighbor w, i.e. $\mathbf{X}_w^{t-1} = c$. This leads to a conflict with v's current proposal for color c (line 14) and hence $\boldsymbol{\chi}_v^t = 0$. If we exhaust all the neighbors and don't find any conflicts (line 16) then $\boldsymbol{\chi}_v^t = 1$.

Now we analyze the runtime of $\textsc{Accepted}$ by constructing and solving a recurrence relation. We will use the following lemma to evaluate the expectation of products of relevant random variables.

▶ **Lemma 44.** *The probability that any given proposal is rejected* $\mathbb{P}[\boldsymbol{\chi}_v^t = 0]$ *is at most* $1/\alpha$. *Moreover, this upper bound holds even if we condition on all the values in* $\mathbf{C}$ *except* $\mathbf{C}_v^t$.

Proof. A rejection can occur due to a conflict with at most 2Δ possible values in $\{\mathbf{C}_w^t, \mathbf{X}_w^{t-1} | w \in \Gamma(v)\}$. Since there are $2\alpha\Delta$ colors, the rejection probability is at most $1/\alpha$. ◀

▶ **Definition 45.** *We define* T_t *to be a random variable indicating the number of recursive calls performed during the execution of* $\textsc{Accepted}(v, t)$ *while computing a single* $\boldsymbol{\chi}_v^t$.

▶ **Definition 46.** *We define $W^t_{t'}$ to be a random variable indicating the number of calls to* ACCEPTED *that are required, to check whether a color c assigned at epoch t' was overwritten at some epoch before t.*

Using $\mathcal{B}(p)$ to denote the Bernoulli random variable with bias p, we obtain an expression for $W^t_{t'}$ (using $\leq$ to denote stochastic dominance).

$$W^t_{t'} \leq \left[T_{t'+1} + \mathcal{B}\left(\frac{1}{\alpha}\right) \cdot T_{t'+2} + \mathcal{B}\left(\frac{1}{\alpha^2}\right) \cdot T_{t'+3} + \cdots + \mathcal{B}\left(\frac{1}{\alpha^{t-t'-2}}\right) \cdot T_{t-1} \right] \tag{7}$$

The above Equation 7 conservatively assumes that the call to ACCEPTED$(v, t'+1)$ in line 10 is always invoked (resulting in $T_{t'+1}$ invocations of ACCEPTED). However, the next call to ACCEPTED$(v, t'+2)$ occurs only if the previous one was not accepted, which happens with probability $\leq 1/\alpha$ (Lemma 44). This produces the $\mathcal{B}(1/\alpha) \cdot T_{t'+2}$ term in the expression. In general, ACCEPTED$(v, t'+i)$ is only invoked if the preceding $i-1$ calls to ACCEPTED all returned 0. This event happens with probability at most $1/\alpha^{i-1}$.

Next, we prove our main lemma that bounds the number of recursive calls to ACCEPTED.

▶ **Lemma 47.** *Given graph G and $q = 2\alpha\Delta$ colors, for $\alpha > 4.5$, the expected number of recursive calls to the procedure* ACCEPTED *while computing a single $\boldsymbol{\chi}^t_v =$* ACCEPTED(v, t) *is $\mathbb{E}[T_t] = \mathcal{O}\left(e^{1.02t/\alpha}\right)$.*

Proof. We start by constructing a recurrence for the expected number of calls to ACCEPTED used by the algorithm. When checking a single neighbor w, the algorithm iterates through all the epochs t' such that $\mathbf{C}^{t'}_w = c$ (in reality, only the last occurence matters, but we are looking for an upper bound). If such a t' is found (which happens with probability $1/q$ independently for each trial), there is a recursive call to ACCEPTED(w, t'), which in turn results in $T_{t'}$ recursive calls to ACCEPTED. If we find that $\boldsymbol{\chi}^{t'}_w = 1$ (i.e. w was colored to c at epoch t'), we will need to proceed to check whether the color was subsequently overwritten, which requires an additional $W^t_{t'}$ calls to ACCEPTED. Summing up over all neighbors and epochs, we obtain the following bound:

$$T_t \leq \sum_{w \in \Gamma(v)} \sum_{t'=1}^{t} \mathbb{P}\left[\mathbf{C}^{t'}_w = c\right] \cdot \left[T_{t'} + W^t_{t'}\right] \tag{8}$$

$$\leq \Delta \cdot \sum_{t'=1}^{t} \mathcal{B}\left(\frac{1}{q}\right) \cdot \left[T_{t'} + T_{t'+1} + \mathcal{B}\left(\frac{1}{\alpha}\right) \cdot T_{t'+2} + \mathcal{B}\left(\frac{1}{\alpha^2}\right) \cdot T_{t'+3} + \cdots \right.$$

$$\left. \cdots + \mathcal{B}\left(\frac{1}{\alpha^{t-t'-2}}\right) \cdot T_{t-1} \right] \tag{9}$$

$$\leq \Delta \cdot \mathcal{B}\left(\frac{1}{q}\right) \cdot \left[\sum_{t'=1}^{t-1} T_{t'} + \sum_{t'=1}^{t-1} T_{t'} \cdot \left(1 + \mathcal{B}\left(\frac{1}{\alpha}\right) + \mathcal{B}\left(\frac{1}{\alpha^2}\right) + \cdots\right) \right] \tag{10}$$

In the last step, we just grouped together all the terms corresponding to the same epoch (note that we include additional terms since it's an upper bound). Using Lemma 44 and the fact that $\mathbb{P}[\mathbf{C}^{t'}_w = c]$ is independent of all other events, we can write a recurrence for the expected number of probes.

$$\mathbb{E}[T_t] \leq \Delta \cdot \frac{1}{2\alpha\Delta} \left[\sum_{t'=1}^{t-1} T_{t'} + \sum_{t'=1}^{t-1} T_{t'} \cdot \left(1 + \frac{1}{\alpha} + \frac{1}{\alpha^2} + \cdots\right) \right] \leq \frac{1}{2\alpha} \cdot \sum_{t'=1}^{t-1} T_{t'} \cdot \left[1 + \frac{\alpha}{\alpha - 1}\right] \tag{11}$$

Now, we make the assumption that $\mathbb{E}[T_{t'}] \leq e^{kt/\alpha}$, and show that this satisfies the expectation recurrence for the desired value of k. First, we sum the geometric series:

$$\sum_{t'=1}^{t-1} \mathbb{E}[T_{t'}] = \sum_{t'=1}^{t-1} e^{kt'/\alpha} < \frac{e^{kt/\alpha} - 1}{e^{k/\alpha} - 1} < \frac{e^{kt/\alpha}}{e^{k/\alpha} - 1}$$

The expectation recurrence to be satisfied then becomes:

$$\mathbb{E}[T_t] \leq \frac{1}{2\alpha} \cdot \frac{e^{kt/\alpha}}{e^{k/\alpha} - 1} \cdot \left[1 + \frac{\alpha}{\alpha - 1}\right] = e^{kt/\alpha} \cdot \frac{2\alpha - 1}{2\alpha(\alpha - 1)(e^{k/\alpha} - 1)} = e^{kt/\alpha} \cdot f(\alpha, k)$$

We notice that for $k = 1.02$ and $\alpha > 4.5$, $f(\alpha) < 1$. This can easily be verified by checking that $f(\alpha, 1.02)$ decreases monotonically with α in the range $\alpha > 4.5$. Thus, our recurrence is satisfied for $k = 1.02$, and therefore the expected number of calls is $\mathcal{O}(e^{1.02t/\alpha})$. ◀

▶ **Theorem 7.** *Given neighborhood query access to a graph G with n nodes, maximum degree Δ, and $q = 2\alpha\Delta \geq 9\Delta$ colors, we can generate the color of any given node from a distribution of color assignments that is ϵ-close (in L_1 distance) to the uniform distribution over all valid colorings of G, in a consistent manner, using only $\mathcal{O}((n/\epsilon)^{3.06/\alpha}\Delta \log(n/\epsilon))$ time, probes, and public random bits per query.*

Proof. Since $q \geq 9\Delta$, we can use Corollary 43 to obtain $\tau_{mix}(\epsilon) \leq 3(\ln n + \ln 1/\epsilon)$. Also, since $\alpha > 4.5$, we can invoke Lemma 47 to conclude that the number of calls to ACCEPTED is $\mathcal{O}(n^{3.06/\alpha}\epsilon^{-3.06/\alpha})$. Finally, we note that each call to ACCEPTED(v, t) potentially reads $\mathcal{O}(t\Delta)$ color proposals from the public random tape, while iterating through all $\leq \Delta$ neighbors of v in all t epochs. Since $t \leq 3\ln(n/\epsilon)$, this implies that the algorithm uses $\mathcal{O}((n/\epsilon)^{3.06/\alpha}\Delta \log(n/\epsilon))$ time and random bits, which is sub-linear for $\alpha > 3.06$. ◀

6 Open Problems

There are many interesting directions to pursue in this area. Below, we provide a few examples of random objects that may admit local access implementations.

Small Description Size

- Provide a local access implementation of degree queries for undirected random graphs, even for $G(n, p)$. How about i^{th} neighbor queries?
- For simple models such as $G(n, p)$, provide a local access implementation of a RANDOM-TRIANGLE(v) query, that returns a uniformly random triangle containing vertex v.
- Provide a memory-less local access implementation of basic queries for undirected random graphs.
- Given an ordered graph such as a lattice, provide an implementation to locally access a random perfect matching. Interesting special cases of this problem include random domino and lozenge tilings.

Huge Description Size

- Provide a faster local access implementation for sampling the color of a specified vertex in a random q-coloring of a bounded degree graph G.
- Improve the local access implementation for sampling the color of a specified vertex in a random q-coloring of G, by supporting smaller values of q (smaller than 9Δ). We remark that this problem in particular should be feasible, by simulating a faster mixing Markov chain. The important question is whether you can you get down to $q = 2\Delta$?

- Given query access to an input graph G and starting vertex v, provide a local access implementation for sampling the location of a random walk starting at v after t steps. This may be feasible in certain restricted classes of graphs.
- Given query access to an input DNF formula, provide an implementation to access the truth value of a single variable in a uniformly random satisfying assignment.

References

1 Emmanuel Abbe. Community detection and stochastic block models: recent developments. *The Journal of Machine Learning Research*, 18(1):6446–6531, 2017.

2 Emmanuel Abbe, Afonso S Bandeira, and Georgina Hall. Exact recovery in the stochastic block model. *IEEE Transactions on Information Theory*, 62(1):471–487, 2016.

3 Emmanuel Abbe and Colin Sandon. Community detection in general stochastic block models: Fundamental limits and efficient algorithms for recovery. In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*, pages 670–688. IEEE, 2015.

4 Maksudul Alam and Maleq Khan. Parallel algorithms for generating random networks with given degree sequences. *International Journal of Parallel Programming*, 45(1):109–127, 2017.

5 Noga Alon, Ronitt Rubinfeld, Shai Vardi, and Ning Xie. Space-efficient local computation algorithms. In *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms*, pages 1132–1139. Society for Industrial and Applied Mathematics, 2012.

6 Danielle Smith Bassett and ED Bullmore. Small-world brain networks. *The neuroscientist*, 12(6):512–523, 2006.

7 Vladimir Batagelj and Ulrik Brandes. Efficient generation of large random networks. *Physical Review E*, 71(3):036113, 2005.

8 Russ Bubley and Martin Dyer. Path coupling: A technique for proving rapid mixing in Markov chains. In *Proceedings 38th Annual Symposium on Foundations of Computer Science*, pages 223–231. IEEE, 1997.

9 Irineo Cabreros, Emmanuel Abbe, and Aristotelis Tsirigos. Detecting community structures in Hi-C genomic data. In *Information Science and Systems (CISS), 2016 Annual Conference on*, pages 584–589. IEEE, 2016.

10 Yi-Jun Chang, Manuela Fischer, Mohsen Ghaffari, Jara Uitto, and Yufan Zheng. The Complexity of $(\Delta+1)$ Coloring in Congested Clique, Massively Parallel Computation, and Centralized Local Computation. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, pages 471–480. ACM, 2019.

11 Jingchun Chen and Bo Yuan. Detecting functional modules in the yeast protein–protein interaction network. *Bioinformatics*, 22(18):2283–2290, 2006.

12 Peter Chin, Anup Rao, and Van Vu. Stochastic block model and community detection in sparse graphs: A spectral algorithm with optimal rate of recovery. In *Conference on Learning Theory*, pages 391–423, 2015.

13 Melissa S Cline, Michael Smoot, Ethan Cerami, Allan Kuchinsky, Nerius Landys, Chris Workman, Rowan Christmas, Iliana Avila-Campilo, Michael Creech, Benjamin Gross, et al. Integration of biological networks and gene expression data using Cytoscape. *Nature protocols*, 2(10):2366–2382, 2007.

14 Mark De Berg, Marc Van Kreveld, Mark Overmars, and Otfried Cheong Schwarzkopf. Computational geometry. In *Computational geometry*, pages 105–109. Springer, 2000.

15 Peter Sheridan Dodds, Roby Muhamad, and Duncan J Watts. An experimental study of search in global social networks. *science*, 301(5634):827–829, 2003.

16 David Easley and Jon Kleinberg. *Networks, crowds, and markets: Reasoning about a highly connected world*. Cambridge University Press, 2010.

17 Paul Erdos and Alfréd Rényi. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci*, 5(1):17–60, 1960.

18 Guy Even, Reut Levi, Moti Medina, and Adi Rosén. Sublinear Random Access Generators for Preferential Attachment Graphs. In *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, pages 6:1–6:15, 2017. doi:10.4230/LIPIcs.ICALP.2017.6.

19 Uriel Feige, Boaz Patt-Shamir, and Shai Vardi. On the probe complexity of local computation algorithms. *arXiv preprint*, 2017. arXiv:1703.07734.

20 Manuela Fischer and Mohsen Ghaffari. A Simple Parallel and Distributed Sampling Technique: Local Glauber Dynamics. In *32nd International Symposium on Distributed Computing (DISC 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

21 Santo Fortunato. Community detection in graphs. *Physics reports*, 486(3):75–174, 2010.

22 Alan Frieze and Eric Vigoda. A survey on the use of Markov chains to randomly sample colourings. *Oxford Lecture Series in Mathematics and its Applications*, 34:53, 2007.

23 Anna C Gilbert, Sudipto Guha, Piotr Indyk, Yannis Kotidis, Sivaramakrishnan Muthukrishnan, and Martin J Strauss. Fast, small-space algorithms for approximate histogram maintenance. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 389–398. ACM, 2002.

24 O Goldreich, S Goldwasser, and A Nussboim. On the implementation of huge random objects. In *Foundations of Computer Science, 2003. Proceedings. 44th Annual IEEE Symposium on*, pages 68–79. IEEE, 2003.

25 Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986. doi:10.1145/6490.6503.

26 Oded Goldreich, Shafi Goldwasser, and Asaf Nussboim. On the implementation of huge random objects. *SIAM Journal on Computing*, 39(7):2761–2822, 2010.

27 Oded Goldreich, Shari Goldwasser, and Dana Ron. Property testing and its connection to learning and approximation. *Journal of the ACM (JACM)*, 45(4):653–750, 1998.

28 Oded Goldreich and Dana Ron. Property testing in bounded degree graphs. In *Proceedings of the twenty-ninth annual ACM Symposium on Theory of Computing*, pages 406–415. ACM, 1997.

29 Paul W Holland, Kathryn Blackmond Laskey, and Samuel Leinhardt. Stochastic blockmodels: First steps. *Social networks*, 5(2):109–137, 1983.

30 Daxin Jiang, Chun Tang, and Aidong Zhang. Cluster analysis for gene expression data: a survey. *IEEE Transactions on knowledge and data engineering*, 16(11):1370–1386, 2004.

31 Jon Kleinberg. The small-world phenomenon: An algorithmic perspective. In *Proceedings of the thirty-second annual ACM Symposium on Theory of Computing*, pages 163–170. ACM, 2000.

32 Donald E Knuth. The art of computer programming, 3rd edn. seminumerical algorithms, vol. 2, 1997.

33 Greg Linden, Brent Smith, and Jeremy York. Amazon. com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing*, 7(1):76–80, 2003.

34 Michael Luby and Charles Rackoff. How to Construct Pseudorandom Permutations from Pseudorandom Functions. *SIAM J. Comput.*, 17(2):373–386, 1988. doi:10.1137/0217022.

35 Yishay Mansour, Aviad Rubinstein, Shai Vardi, and Ning Xie. Converting Online Algorithms to Local Computation Algorithms. In *Automata, Languages, and Programming - 39th International Colloquium, ICALP 2012, Warwick, UK, July 9-13, 2012, Proceedings, Part I*, pages 653–664, 2012. doi:10.1007/978-3-642-31594-7_55.

36 Edward M Marcotte, Matteo Pellegrini, Ho-Leung Ng, Danny W Rice, Todd O Yeates, and David Eisenberg. Detecting protein function and protein-protein interactions from genome sequences. *Science*, 285(5428):751–753, 1999.

37 Chip Martel and Van Nguyen. Analyzing Kleinberg's (and other) small-world models. In *Proceedings of the twenty-third annual ACM Symposium on Principles of Distributed Computing*, pages 179–188. ACM, 2004.

38 Joel Miller and Aric Hagberg. Efficient generation of networks with given expected degrees. *Algorithms and models for the web graph*, pages 115–126, 2011.

39 Ron Milo, Nadav Kashtan, Shalev Itzkovitz, Mark EJ Newman, and Uri Alon. On the uniform generation of random graphs with prescribed degree sequences. *arXiv preprint*, 2003. `arXiv:cond-mat/0312028`.

40 Elchanan Mossel, Joe Neeman, and Allan Sly. Reconstruction and estimation in the planted partition model. *Probability Theory and Related Fields*, 162(3-4):431–461, 2015.

41 Moni Naor and Asaf Nussboim. Implementing huge sparse random graphs. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 596–608. Springer, 2007.

42 Moni Naor and Omer Reingold. On the Construction of Pseudo-Random Permutations: Luby-Rackoff Revisited. *IACR Cryptology ePrint Archive*, 1996:11, 1996. URL: `http://eprint.iacr.org/1996/011`.

43 Mark EJ Newman. Models of the small world. *Journal of Statistical Physics*, 101(3):819–841, 2000.

44 Mark EJ Newman, Duncan J Watts, and Steven H Strogatz. Random graph models of social networks. *Proceedings of the National Academy of Sciences*, 99(suppl 1):2566–2572, 2002.

45 Sadegh Nobari, Xuesong Lu, Panagiotis Karras, and Stéphane Bressan. Fast random graph generation. In *Proceedings of the 14th International Conference on Extending Database Technology*, pages 331–342. ACM, 2011.

46 Michal Parnas and Dana Ron. Approximating the minimum vertex cover in sublinear time and a connection to distributed algorithms. *Theoretical Computer Science*, 381(1-3):183–196, 2007.

47 Shlomi Reuveni. CATALAN'S TRAPEZOIDS. *Probability in the Engineering and Informational Sciences*, 28(03):353–361, 2014.

48 Ronitt Rubinfeld, Gil Tamir, Shai Vardi, and Ning Xie. Fast local computation algorithms. *arXiv preprint*, 2011. `arXiv:1104.1377`.

49 Shaghayegh Sahebi and William W Cohen. Community-based recommendations: a solution to the cold start problem. In *Workshop on recommender systems and the social web, RSWEB*, page 60, 2011.

50 Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

51 Therese Sørlie, Charles M Perou, Robert Tibshirani, Turid Aas, Stephanie Geisler, Hilde Johnsen, Trevor Hastie, Michael B Eisen, Matt Van De Rijn, Stefanie S Jeffrey, et al. Gene expression patterns of breast carcinomas distinguish tumor subclasses with clinical implications. *Proceedings of the National Academy of Sciences*, 98(19):10869–10874, 2001.

52 Joel Spencer. *Asymptopia*, volume 71. American Mathematical Soc., 2014.

53 Richard P Stanley. *Catalan numbers*. Cambridge University Press, 2015.

54 Jeffrey Travers and Stanley Milgram. The small world problem. *Phychology Today*, 1:61–67, 1967.

55 Duncan J Watts and Steven H Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, 1998.

A Further Analysis and Extensions of Algorithm 1: Sampling `Next-Neighbor` without Blocks

A.1 Performance Guarantee

This section is devoted to showing the following lemma that bounds the required resources per query of Algorithm 1. We note that we only require efficient computation of $\prod_{u\in[a,b]}(1-p_{vu})$ (and not $\sum_{u\in[a,b]} p_{vu}$), and that for the $G(n,p)$ model, the resources required for such computation is asymptotically negligible.

▶ **Theorem 48.** *Each execution of Algorithm 1 (the* NEXT-NEIGHBOR *query), with high probability,*

- *terminates within* $\mathcal{O}(\log n)$ *iterations (of its* **repeat** *loop);*
- *computes* $\mathcal{O}(\log^2 n)$ *quantities of* $\prod_{u\in[a,b]}(1-p_{vu})$*;*
- *aside from the above computations, uses* $\mathcal{O}(\log^2 n)$ *time,* $\mathcal{O}(\log n)$ *random* N*-bit words, and* $\mathcal{O}(\log n)$ *additional space.*

Proof. We focus on the number of iterations as the remaining results follow trivially. This proof is rather involved and thus is divided into several steps.

Specifying random choices

The performance of the algorithm depends on not only the random variables X_{vu}'s, but also the unused coins C_{vu}'s. We characterize the two collections of Bernoulli variables $\{X_{vu}\}$ and $\{Y_{vu}\}$ that cover all random choices made by Algorithm 1 as follows.

- Each X_{vu} (same as X_{uv}) represents the result for the *first* coin-toss corresponding to cells $\mathbf{A}[v][u]$ and $\mathbf{A}[u][v]$, which is the coin-toss obtained when X_{vu} becomes decided: either C_{vu} during a NEXT-NEIGHBOR(v) call when $\mathbf{A}[v][u]=\phi$, or C_{vu} during a NEXT-NEIGHBOR(u) call when $\mathbf{A}[u][v]=\phi$, whichever occurs first. This description of X_{vu} respects our invariant that, if the generation process is executed to completion, we will have $\mathbf{A}[v][u]=X_{vu}$ in all entries.
- Each Y_{vu} represents the result for the *second* coin-toss corresponding to cell $\mathbf{A}[v][u]$, which is the coin-toss C_{vu} obtained during a NEXT-NEIGHBOR(v) call when X_{vu} is already decided. In other words, $\{Y_{vu}\}$'s are the coin-tosses that should have been skipped but still performed in Algorithm 1 (if they have indeed been generated). Unlike the previous case, Y_{vu} and Y_{uv} are two independent random variables: they may be generated during a NEXT-NEIGHBOR(v) call and a NEXT-NEIGHBOR(u) call, respectively.

As mentioned earlier, we allow any sequence of probabilities p_{vu} in our proof. The success probabilities of these indicators are therefore given by $\mathbb{P}[X_{vu}=1]=\mathbb{P}[Y_{vu}=1]=p_{vu}$.

Characterizing iterations

Suppose that we compute NEXT-NEIGHBOR(v) and obtain an answer u. Then $X_{v,\mathbf{last}[v]+1}=\cdots=X_{v,u-1}=0$ as none of $u'\in(\mathbf{last}[v],u)$ is a neighbor of v. The vertices considered in the loop of Algorithm 1 that do not result in the answer u, are $u'\in(\mathbf{last}[v],u)$ satisfying $\mathbf{A}[v][u']=0$ and $Y_{v,u'}=1$; we call the iteration corresponding to such a u' a *failed iteration*. Observe that if $X_{v,u'}=0$ but is undecided ($\mathbf{A}[v][u']=\phi$), then the iteration is not failed, even if $Y_{v,u'}=1$ (in which case, $X_{v,u'}$ takes the value of $C_{v,u'}$ while $Y_{v,u'}$ is never used). Thus we assume the worst-case scenario where all $X_{v,u'}$ are revealed: $\mathbf{A}[v][u']=X_{v,u'}=0$ for all $u'\in(\mathbf{last}[v],u)$. The number of failed iterations in this case stochastically dominates those in all other cases.[6]

[6]There exists an adversary who can enforce this worst case. Namely, an adversary that first makes NEXT-NEIGHBOR queries to learn all neighbors of every vertex except for v, thereby filling out the whole $\mathbf{A}$ in the process. The claimed worst case then occurs as this adversary now repeatedly makes NEXT-NEIGHBOR queries on v. In particular, a committee of n adversaries, each of which is tasked to perform this series of calls corresponding to each v, can always expose this worst case.

Then, the upper bound on the number of failed iterations of a call NEXT-NEIGHBOR(v) is given by the maximum number of cells $Y_{v,u'} = 1$ of $u' \in (\mathbf{last}[v], u)$, over any $u \in (\mathbf{last}[v], n]$ satisfying $X_{v,\mathbf{last}[v]+1} = \cdots = X_{vu} = 0$. Informally, we are asking "of all consecutive cells of 0's in a single row of $\{X_{vu}\}$-table, what is the largest number of cells of 1's in the corresponding cells of $\{Y_{vu}\}$-table?"

Bounding the number of iterations required for a fixed pair $(v, \mathrm{last}[v])$

We now proceed to bounding the number of iterations required over a sampled pair of $\{X_{vu}\}$ and $\{Y_{vu}\}$, from any probability distribution. For simplicity we renumber our indices and drop the index $(v, \mathbf{last}[v])$ as follows. Let $p_1, \ldots, p_L \in [0, 1]$ denote the probabilities corresponding to the cells $\mathbf{A}[v][\mathbf{last}[v] + 1 \ldots n]$ (where $L = n - \mathbf{last}[v]$), then let $X_1, \ldots, X_L$ and $Y_1, \ldots, Y_L$ be the random variables corresponding to the same cells on $\mathbf{A}$.

For $i = 1, \ldots, L$, define the random variable Z_i in terms of X_i and Y_i so that

- $Z_i = 2$ if $X_i = 0$ and $Y_i = 1$, which occurs with probability $p_i(1 - p_i)$.
  This represents the event where i is not a neighbor, and the iteration fails.
- $Z_i = 1$ if $X_i = Y_i = 0$, which occurs with probability $(1 - p_i)^2$.
  This represents the event where i is not a neighbor, and the iteration does not fail.
- $Z_i = 0$ if $X_i = 1$, which occurs with probability p_i.
  This represents the event where i is a neighbor.

For $\ell \in [L]$, define the random variable $M_\ell := \prod_{i=1}^{\ell} Z_i$, and $M_0 = 1$ for convenience. If $X_i = 1$ for some $i \in [1, \ell]$, then $Z_i = 0$ and $M_\ell = 0$. Otherwise, $\log M_\ell$ counts the number of indices $i \in [\ell]$ with $Y_i = 1$, the number of failed iterations. Therefore, $\log(\max_{\ell \in \{0,\ldots,L\}} M_\ell)$ gives the number of failed iterations this NEXT-NEIGHBOR(v) call.

To bound M_ℓ, observe that for any $\ell \in [L]$, $\mathbb{E}[Z_\ell] = 2p_\ell(1 - p_\ell) + (1 - p_\ell)^2 = 1 - p_\ell^2 \leq 1$ regardless of the probability $p_\ell \in [0, 1]$. Then, $\mathbb{E}[M_\ell] = \mathbb{E}[\prod_{i=1}^{\ell} Z_i] = \prod_{i=1}^{\ell} \mathbb{E}[Z_i] \leq 1$ because Z_ℓ's are all independent. By Markov's inequality, for any (integer) $r \geq 0$, $\Pr[\log M_\ell > r] = \Pr[M_\ell > 2^r] < 2^{-r}$. By the union bound, the probability that more than r failed iterations are encountered is $\Pr[\log(\max_{\ell \in \{0,\ldots,L\}} M_\ell) > r] < L \cdot 2^{-r} \leq n \cdot 2^{-r}$.

Establishing the overall performance guarantee

So far we have deduced that, for each pair of a vertex v and its $\mathbf{last}[v]$, the probability that the call NEXT-NEIGHBOR(v) encounters more than r failed iterations is less that $n \cdot 2^{-r}$, which is at most n^{-c-2} for any desired constant c by choosing a sufficiently large $r = \Theta(\log n)$. As Algorithm 1 may need to support up to $\Theta(n^2)$ NEXT-NEIGHBOR calls, one corresponding to each pair $(v, \mathbf{last}[v])$, the probability that it ever encounters more than $O(\log n)$ failed iterations to answer a single NEXT-NEIGHBOR query is at most n^{-c}. That is, with high probability, $O(\log n)$ iterations are required per NEXT-NEIGHBOR call, which concludes the proof of Theorem 48. ◀

A.2 Supporting Vertex-Pair Queries

We extend our implementation (Algorithm 1) to support the VERTEX-PAIR queries: given a pair of vertices (u, v), decide whether there exists an edge $\{u, v\}$ in the generated graph. To answer a VERTEX-PAIR query, we must first check whether the value X_{uv} for $\{u, v\}$ has already been assigned, in which case we answer accordingly. Otherwise, we must make a coin-flip with the corresponding bias p_{uv} to assign X_{uv}, deciding whether $\{u, v\}$ exists in the generated graph. If we maintained the full $\mathbf{A}$, we would have been able to simply set $\mathbf{A}[u][v]$ and $\mathbf{A}[v][u]$ to this new value. However, our more efficient Algorithm 1 that represents $\mathbf{A}$ compactly via $\mathbf{last}$ and P_v's cannot record arbitrary modifications to $\mathbf{A}$.

Observe that if we were to apply the trivial implementation of VERTEX-PAIR, then by Lemma 12, **last** and P_v's will only fail capture the state $\mathbf{A}[v][u] = 0$ when $u > \mathbf{last}[v]$ and $v > \mathbf{last}[u]$. Fortunately, unlike NEXT-NEIGHBOR queries, a VERTEX-PAIR query can only set one cell $\mathbf{A}[v][u]$ to 0 per query, and thus we may afford to store these changes explicitly.[7] To this end, we define the set $Q = \{\{u, v\} : X_{uv}$ is assigned to 0 during a VERTEX-PAIR query$\}$, maintained as a hash table. Updating Q during VERTEX-PAIR queries is trivial: we simply add $\{u, v\}$ to Q before we finish processing the query if we set $\mathbf{A}[u][v] = 0$. Conversely, we need to add u to P_v and add v to P_u if the VERTEX-PAIR query sets $\mathbf{A}[u][v] = 1$ as usual, yielding the following observation. It is straightforward to verify that each VERTEX-PAIR query requires $O(\log n)$ time, $O(1)$ random N-bit word, and $O(1)$ additional space per query.

▶ **Lemma 49.** *The data structures* **last**, P_v*'s and* Q *together provide a succinct representation of* $\mathbf{A}$ *when* NEXT-NEIGHBOR *queries (modified Algorithm 1) and* VERTEX-PAIR *queries are allowed. In particular,* $\mathbf{A}[v][u] = 1$ *if and only if* $u \in P_v$*. Otherwise,* $\mathbf{A}[v][u] = 0$ *if* $u < \mathbf{last}[v]$*,* $v < \mathbf{last}[u]$*, or* $\{v, u\} \in Q$*. In all remaining cases,* $\mathbf{A}[v][u] = \phi$*.*

We now explain other necessary changes to Algorithm 1. In the implementation of NEXT-NEIGHBOR, an iteration is not failed when the chosen X_{vu} is still undecided: $\mathbf{A}[v][u]$ must still be ϕ. Since X_{vu} may also be assigned to 0 via a VERTEX-PAIR(v, u) query, we must also consider an iteration where $\{v, u\} \in Q$ failed. That is, we now require one additional condition $\{v, u\} \notin Q$ for termination (which only takes $O(1)$ time to verify per iteration). As for the analysis, aside from handling the fact that X_{vu} may also become decided during a VERTEX-PAIR call, and allowing the states of the algorithm to support VERTEX-PAIR queries, all of the remaining analysis for correctness and performance guarantee still holds.

Therefore, we have established that our augmentation to Algorithm 1 still maintains all of its (asymptotic) performance guarantees for NEXT-NEIGHBOR queries, and supports VERTEX-PAIR queries with complexities as specified above, concluding the following corollary. We remark that, as we do not aim to support RANDOM-NEIGHBOR queries, this simple algorithm here provides significant improvement over the performance of RANDOM-NEIGHBOR queries (given in Corollary 19).

▶ **Corollary 50.** *Algorithm 1 can be modified to allow an implementation of* VERTEX-PAIR *query as explained above, such that the resource usages per query still asymptotically follow those of Theorem 48.*

B Omitted Details from Section 3: Undirected Random Graph Implementations

B.1 Removing the Perfect-Precision Arithmetic Assumption

In this section we remove the prefect-precision arithmetic assumption. Instead, we only assume that it is possible to compute $\prod_{u=a}^{b}(1 - p_{vu})$ and $\sum_{u=a}^{b} p_{vu}$ to N-bit precision, as well as drawing a random N-bit word, using polylogarithmic resources. Here we will focus on proving that the family of the random graph we generate via our procedures is statistically close to that of the desired distribution. The main technicality of this lemma arises from the fact that, not only the implementation is randomized, but the agent interacting with the implementation may choose their queries arbitrarily (or adversarially): our proof must handle any sequence of random choices the implementation makes, and any sequence of queries the agent may make.

[7] The disadvantage of this approach is that the implementation may allocate more than $\Theta(m)$ space over the entire graph generation process, if VERTEX-PAIR queries generate many of these 0's.

Observe that the distribution of the graphs constructed by our implementation is governed entirely by the samples u drawn from $\mathsf{F}(v,a,b)$ in Algorithm 3. By our assumption, the CDF of any $\mathsf{F}(v,a,b)$ can be efficiently computed from $\prod_{u=a}^{u'}(1-p_{vu})$, and thus sampling with $\frac{1}{\mathrm{poly}(n)}$ error in the L_1-distance requires a random N-bit word and a binary-search in $\mathcal{O}(\log(b-a+1)) = \mathcal{O}(\log n)$ iterations. Using this crucial fact, we prove our lemma that removes the perfect-precision arithmetic assumption.

▶ **Lemma 51.** *If Algorithm 3 (the* FILL *operation) is repeatedly invoked to construct a graph G by drawing the value u for at most S times in total, each of which comes from some distribution $\mathsf{F}'(v,a,b)$ that is ϵ-close in L_1-distance to the correct distribution $\mathsf{F}(v,a,b)$ that perfectly generates the desired distribution G over all graphs, then the distribution G' of the generated graph G is (ϵS)-close to G in the L_1-distance.*

Proof. For simplicity, assume that the algorithm generates the graph to completion according to a sequence of up to n^2 distinct blocks $\mathcal{B} = \langle B_{v_1}^{(u_1)}, B_{v_2}^{(u_2)}, \ldots\rangle$, where each $B_{v_i}^{(u_i)}$ specifies the unfilled block in which any query instigates a FILL function call. Define an *internal state* of our implementation as the triplet $s = (k, u, \mathbf{A})$, representing that the algorithm is currently processing the k^{th} FILL, in the iteration (the **repeat** loop of Algorithm 3) with value u, and have generated $\mathbf{A}$ so far. Let $t_{\mathbf{A}}$ denote the *terminal state* after processing all queries and having generated the graph $G_{\mathbf{A}}$ represented by $\mathbf{A}$. We note that $\mathbf{A}$ is used here in the analysis but not explicitly maintained; further, it reflects the changes in every iteration: as u is updated during each iteration of FILL, the cells $\mathbf{A}[v][u'] = \phi$ for $u' < u$ (within that block) that has been skipped are also updated to 0.

Let $\mathcal{S}$ denote the set of all (internal and terminal) states. For each state s, the implementation samples u from the corresponding $\mathsf{F}'(v,a,b)$ where $\|\mathsf{F}(v,a,b) - \mathsf{F}'(v,a,b)\|_1 \le \epsilon = \frac{1}{\mathrm{poly}(n)}$, then moves to a new state according to u. In other words, there is an induced pair of collection of distributions over the states: $(\mathcal{T}, \mathcal{T}')$ where $\mathcal{T} = \{\mathsf{T}_s\}_{s\in\mathcal{S}}, \mathcal{T}' = \{\mathsf{T}'_s\}_{s\in\mathcal{S}}$, such that $\mathsf{T}_s(s')$ and $\mathsf{T}'_s(s')$ denote the probability that the algorithm advances from s to s' by using a sample from the correct $\mathsf{F}(v,a,b)$ and from the approximated $\mathsf{F}'(v,a,b)$, respectively. Consequently, $\|\mathsf{T}_s - \mathsf{T}'_s\|_1 \le \epsilon$ for every $s \in \mathcal{S}$.

The implementation begins with the initial (internal) state $s_0 = (1, 0, \mathbf{A}_\phi)$ where all cells of $\mathbf{A}_\phi$ are ϕ's, goes through at most $S = O(n^3)$ other states (as there are up to n^2 values of k and $O(n)$ values of u), and reach some terminal state $t_{\mathbf{A}}$, generating the entire graph in the process. Let $\pi = \langle s_0^\pi = s_0, s_1^\pi, \ldots, s_{\ell(\pi)}^\pi = t_{\mathbf{A}}\rangle$ for some $\mathbf{A}$ denote a sequence ("path") of up to $S+1$ states the algorithm proceeds through, where $\ell(\pi)$ denote the number of transitions it undergoes. For simplicity, let $T_{t_{\mathbf{A}}}(t_{\mathbf{A}}) = 1$, and $T_{t_{\mathbf{A}}}(s) = 0$ for all state $s \neq t_{\mathbf{A}}$, so that the terminal state can be repeated and we may assume $\ell(\pi) = S$ for every π. Then, for the correct transition probabilities described as $\mathcal{T}$, each π occurs with probability $q(\pi) = \prod_{i=1}^{S} \mathsf{T}_{s_{i-1}}(s_i)$, and thus $\mathsf{G}(G_{\mathbf{A}}) = \sum_{\pi: s_S^\pi = t_{\mathbf{A}}} q(\pi)$.

Let $\mathcal{T}^{\min} = \{\mathsf{T}_s^{\min}\}_{s\in\mathcal{S}}$ where $\mathsf{T}_s^{\min}(s') = \min\{\mathsf{T}_s(s'), \mathsf{T}'_s(s')\}$, and note that each $\mathsf{T}_s^{\min}$ is not necessarily a probability distribution. Then, $\sum_{s'} \mathsf{T}_s^{\min}(s') = 1 - \|\mathsf{T}_s - \mathsf{T}'_s\|_1 \ge 1 - \epsilon$. Define $q', q^{\min}, \mathsf{G}'(G_{\mathbf{A}}), \mathsf{G}^{\min}(G_{\mathbf{A}})$ analogously, and observe that $q^{\min}(\pi) \le \min\{q(\pi), q'(\pi)\}$ for every π, so $\mathsf{G}^{\min}(G_{\mathbf{A}}) \le \min\{\mathsf{G}(G_{\mathbf{A}}), \mathsf{G}'(G_{\mathbf{A}})\}$ for every $G_{\mathbf{A}}$ as well. In other words, $q^{\min}(\pi)$ lower bounds the probability that the algorithm, drawing samples from the correct distributions or the approximated distributions, proceeds through states of π; consequently, $\mathsf{G}^{\min}(G_{\mathbf{A}})$ lower bounds the probability that the algorithm generates the graph $G_{\mathbf{A}}$.

Next, consider the probability that the algorithm proceeds through the prefix $\pi_i = \langle s_0^\pi, \ldots, s_i^\pi \rangle$ of π. Observe that for $i \geq 1$,

$$\sum_\pi q^{\min}(\pi_i) = \sum_\pi q^{\min}(\pi_{i-1}) \cdot \mathsf{T}^{\min}_{s^\pi_{i-1}}(s^\pi_i) = \sum_{s,s'} \sum_{\pi : s^\pi_{i-1}=s, s^\pi_i = s'} q^{\min}(\pi_{i-1}) \cdot \mathsf{T}^{\min}_s(s')$$
$$= \sum_{s'} \mathsf{T}^{\min}_s(s') \cdot \sum_s \sum_{\pi : s^\pi_{i-1}=s} q^{\min}(\pi_{i-1}) \geq (1-\epsilon) \sum_\pi q^{\min}(\pi_{i-1}).$$

Roughly speaking, at least a factor of $1-\epsilon$ of the "agreement" between the distributions over states according to $\mathcal{T}$ and $\mathcal{T}'$ is necessarily conserved after a single sampling process. As $\sum_\pi q^{\min}(\pi_0) = 1$ because the algorithm begins with $s_0 = (1, 0, \mathbf{A}_\phi)$, by an inductive argument we have $\sum_\pi q^{\min}(\pi) = \sum_\pi q^{\min}(\pi_S) \geq (1-\epsilon)^S \geq 1 - \epsilon S$. Hence, $\sum_{G_\mathbf{A}} \min\{\mathsf{G}(G_\mathbf{A}), \mathsf{G}'(G_\mathbf{A})\} \geq \sum_{G_\mathbf{A}} \mathsf{G}^{\min}(G_\mathbf{A}) \geq 1 - \epsilon S$, implying that $\|\mathsf{G} - \mathsf{G}'\|_1 \leq \epsilon S$, as desired. In particular, by substituting $\epsilon = \frac{1}{\mathrm{poly}(n)}$ and $S = O(n^3)$, we have shown that Algorithm 3 only creates a $\frac{1}{\mathrm{poly}(n)}$ error in the L_1-distance. ◀

We remark that RANDOM-NEIGHBOR queries also require that the returned edge is drawn from a distribution that is close to a uniform one, but this requirement applies only *per query* rather then over the entire execution of the generator. Hence, the error due to the selection of a random neighbor may be handled separately from the error for generating the random graph; its guarantee follows straightforwardly from a similar analysis.

B.2 Bounding Block Sizes

▶ **Lemma 13.** *With high probability, the number of neighbors in every block, $|\Gamma^{(i)}(v)|$, is at most $\mathcal{O}(\log n)$.*

Proof. Fix a block $B_v^{(i)}$, and consider the Bernoulli RVs $\{X_{vu}\}_{u \in B_v^{(i)}}$. The expected number of neighbors in this block is $\mathbb{E}\left[|\Gamma^{(i)}(v)|\right] = \mathbb{E}\left[\sum_{u \in B_v^{(i)}} X_{vu}\right] < L + 1$. Via the Chernoff bound,

$$\mathbb{P}\left[|\Gamma^{(i)}(v)| > (1 + 3c \log n) \cdot L\right] \leq e^{-\frac{3c \log n \cdot L}{3}} = n^{-\Theta(c)}$$

for any constant $c > 0$. ◀

▶ **Lemma 14.** *With high probability, for every v such that $|\mathbf{B}_v| = \Omega(\log n)$ (i.e., $\mathbb{E} = \Omega(\log n)$), at least a 1/3-fraction of the blocks $\{B_v^{(i)}\}_{i \in [|\mathbf{B}_v|]}$ are non-empty.*

Proof. For $i < |B_v|$, since $\mathbb{E}\left[|\Gamma^{(i)}(v)|\right] = \mathbb{E}\left[\sum_{u \in B_v^{(i)}} X_{vu}\right] > L - 1$, we bound the probability that $B_v^{(i)}$ is empty:

$$\mathbb{P}[B_v^{(i)} \text{ is empty}] = \prod_{u \in B_v^{(i)}} (1 - p_{vu}) \leq e^{-\sum_{u \in B_v^{(i)}} p_{vu}} \leq e^{1-L} = c$$

for any arbitrary small constant c given sufficienty large constant L. Let T_i be the indicator for the event that $B_v^{(i)}$ is *not* empty, so $\mathbb{E}1 - c$. By the Chernoff bound, the probability that less than $|B_v|/3$ blocks are non-empty is

$$\mathbb{P}\left[\sum_{i \in [|B_v|]} T_i < \tfrac{|B_v|}{3}\right] < \mathbb{P}\left[\sum_{i \in [|B_v|-1]} T_i < \tfrac{|B_v - 1|}{2}\right] \leq e^{-\Theta(|B_v|-1)} = n^{-\Omega(1)}$$

as $|B_v| = \Omega(\log n)$ by assumption. ◀

C Next-Neighbor Implementation with Deterministic Performance Guarantee

In this section, we construct data structures that allow us to sample for the next neighbor directly by considering only the cells $\mathbf{A}[v][u] = \phi$ in the Erdös-Rényi model and the Stochastic Block model. This provides poly$(\log n)$ *worst-case* performance guarantee for implementations supporting only the Next-Neighbor queries. We may again extend this data structure to support Vertex-Pair queries, however, at the cost of providing poly$(\log n)$ *amortized* performance guarantee instead.

In what follows, we first focus on the $G(n, p)$ model, starting with Next-Neighbor queries (Section C.1) then extend to Vertex-Pair queries (Section C.2. We then explain how this result may be generalized to support the Stochastic Block model with random community assignment in Section C.3.

C.1 Data structure for next-neighbor queries in the Erdös-Rényi model

Algorithm 6 Alternate implementation.

procedure Next-Neighbor(v)
  $w \leftarrow \min K_v$, or $n+1$ if $K_v = \emptyset$
  $t \leftarrow$ count(v)
  sample $F \sim$ ExactF(p, t)
  if $F \leq t$
    $u \leftarrow$ pick(v, F)
    $K_u \leftarrow K_u \cup \{v\}$
  else
    $u \leftarrow w$
    if $u \neq n+1$
      $K_v \leftarrow K_v \setminus \{u\}$
  update(v, u)
  last$[v] \leftarrow u$
  return u

Recall that Next-Neighbor(v) is given by $\min\{u > \mathbf{last}[v] : X_{vu} = 1\}$ (or $n+1$ if no satisfying u exists). To aid in computing this quantity, we define:

$$K_v = \{u \in (\mathbf{last}[v], n] : \mathbf{A}[v][u] = 1\},$$
$$w_v = \min K_v, \text{ or } n+1 \text{ if } K_v = \emptyset,$$
$$T_v = \{u \in (\mathbf{last}[v], w_v) : \mathbf{A}[v][u] = \phi\}.$$

The ordered set K_v is only defined for ease of presentation: it is equivalent to $(\mathbf{last}[v], n] \cap P_v$, recording the known neighbors of v after **last**$[v]$ (i.e., those that have not been returned as an answer by any Next-Neighbor(v) query yet). The quantity w_v remains unchanged but is simply restated in terms of K_v. T_v specifies the list of candidates u for Next-Neighbor(v) with $\mathbf{A}[v][u] = \phi$; in particular, all candidates u's, such that the corresponding RVs $X_{vu} = 0$ are decided, are explicitly excluded from T_v.

Unlike the approach of Algorithm 1 that simulates coin-flips even for decided X_{vu}'s, here we only flip undecided coins for the indices in T_v: we have $|T_v|$ Bernoulli trials to simulate. Let F be the random variable denoting the first index of a successful trial out of

$|T_v|$ coin-flips, or $|T_v|+1$ if all fail; denote the distribution of F by $\mathsf{ExactF}(p, |T|)$. The CDF of F is given by $\mathbb{P}[F = f] = 1-(1-p)^f$ for $f \le |T_v|$ (i.e., there is some success trial in the first f trials), and $\mathbb{P}[F = |T_v|+1] = 1$. Thus, we must design a data structure that can compute w_v, compute $|T_v|$, find the F^{th} minimum value in T_v, and update $\mathbf{A}[v][u]$ for the F lowest values $u \in T_v$ accordingly.

Let $k = \lceil \log n \rceil$. We create a range tree, where each node itself contains a balanced binary search tree (BBST), storing **last** values of its corresponding range. Formally, for $i \in [0, n/2^j)$ and $j \in [0, k]$, the i^{th} node of the j^{th} level of the range tree, stores $\mathbf{last}[v]$ for every $v \in (i \cdot 2^{k-j}, (i+1) \cdot 2^{k-j}]$. Denote the range tree by $\mathbf{R}$, and each BBST corresponding to the range $[a, b]$ by $\mathbf{B}_{[a,b]}$. We say that the range $[a, b]$ is *canonical* if it corresponds to a range of some $\mathbf{B}_{[a,b]}$ in $\mathbf{R}$.

Again, to allow fast initialization, we make the following adjustments from the given formalization above: (1) values $\mathbf{last}[v] = 0$ are never stored in any $\mathbf{B}_{[a,b]}$, and (2) each $\mathbf{B}_{[a,b]}$ is created on-the-fly during the first occasion it becomes non-empty. Further, we augment each $\mathbf{B}_{[a,b]}$ so that each of its node maintains the size of the subtree rooted at that node: this allows us to count, in $O(\log n)$ time, the number of entries in $\mathbf{B}_{[a,b]}$ that is no smaller than a given threshold.

Observe that each v is included in exactly one $\mathbf{B}_{[a,b]}$ per level in $\mathbf{R}$, so $k+1 = O(\log n)$ copies of $\mathbf{last}[v]$ are stored throughout $\mathbf{R}$. Moreover, by the property of range trees, any interval can be decomposed into a disjoint union of $O(\log n)$ canonical ranges. From these properties we implement the data structure $\mathbf{R}$ to support the following operations. (Note that $\mathbf{R}$ is initially an empty tree, so initialization is trivial.)

- `COUNT`(v): compute $|T_v|$.
  We break $(\mathbf{last}[v], w_v)$ into $O(\log n)$ disjoint canonical ranges $[a_i, b_i]$'s each corresponding to some $\mathbf{B}_{[a_i,b_i]}$, then compute $t_{[a_i,b_i]} = |\{u \in [a_i, b_i] : \mathbf{last}[u] < v\}|$, and return $\sum_i t_{[a_i,b_i]}$. The value $t_{[a_i,b_i]}$ is obtained by counting the entries of $\mathbf{B}_{[a_i,b_i]}$ that is at least v, then subtract it from $b_i - a_i + 1$; we cannot count entries less than v because $\mathbf{last}[u] = 0$ are not stored.
- `PICK`(v, F): find the F^{th} minimum value in T_v (assuming $F \le |T_v|$).
  We again break $(\mathbf{last}[v], w_v)$ into $O(\log n)$ canonical ranges $[a_i, b_i]$'s, compute $t_{[a_i,b_i]}$'s, and identify the canonical range $[a^*, b^*]$ containing the i^{th} smallest element (i.e., $[a_i, b_i]$ with the smallest b satisfying $\sum_{j \le i} t_{[a_j,b_j]} \ge F$ assuming ranges are sorted). Binary-search in $[a^*, b^*]$ to find exactly the i^{th} smallest element of T. This is accomplished by traversing $\mathbf{R}$ starting from the range $[a^*, b^*]$ down to a leaf, at each step computing the children's $T_{[a,b]}$'s and deciding which child's range contains the desired element.
- `UPDATE`(v, u): simulate coin-flips, assigning $X_{vu} \leftarrow 1$, and $X_{v,u'} \leftarrow 0$ for $u' \in (\mathbf{last}[v], u) \cap T_v$.
  This is done implicitly by handling the change $\mathbf{last}[v] \leftarrow u$: for each BBST $\mathbf{B}_{[a,b]}$ where $v \in [a, b]$, remove the old value of $\mathbf{last}[v]$ and insert u instead.

It is straightforward to verify that all operations require at most $O(\log^2 n)$ time and $O(\log n)$ additional space per call. The overall implementation is given in Algorithm 6, using the same asymptotic time and additional space. Recall also that sampling $F \sim \mathsf{ExactF}(p, t)$ requires $O(\log n)$ time and one N-bit random word for the $G(n, p)$ model.

C.2 Data structure for `Vertex-Pair` queries in the Erdös-Rényi model

Recall that we define Q in Algorithm 1 as the set of pairs (u, v) where X_{uv} is assigned to 0 during a `VERTEX-PAIR` query, allowing us to check for modifications of $\mathbf{A}$ not captured by $\mathbf{last}[v]$ and K_v. Here in Algorithm 6, rather than checking, we need to be able to count such entries. Thus, we instead create a BBST Q'_v for each v defined as:

$Q'_v = \{u : u > \textbf{last}[v], v > \textbf{last}[u],$ and X_{uv} is assigned to 0 during a VERTEX-PAIR query$\}$.

This definition differs from that of Q in Section A.2 in two aspects. First, we ensure that each $\mathbf{A}[v][u] = 0$ is recorded by either **last** (via Lemma 12) or Q'_v (explicitly), but *not both*. In particular, if u were to stay in Q'_v when $\textbf{last}[v]$ increases beyond u, we would have double-counted these entries 0 not only recorded by Q'_v but also implied by $\textbf{last}[v]$ and K_v. By having a BBST for each Q'_v, we can compute the number of 0's that must be excluded from T_v, which cannot be determined via $\textbf{last}[v]$ and K_v alone: we subtract these from any counting process done in the data structure $\mathbf{R}$.

Second, we maintain Q'_v separately for each v as an ordered set, so that we may identify non-neighbors of v within a specific range – this allows us to remove non-neighbors in specific range, ensuring that the first aspect holds. More specifically, when we increase $\textbf{last}[v]$, we must go through the data structure Q'_v and remove all $u < \textbf{last}[v]$, and for each such u, also remove v from Q'_u. There can be as many as linear number of such u, but the number of removals is trivially bounded by the number of insertions, yielding an amortized time performance guarantee in the following theorem. Aside from the deterministic guarantee, unsurprisingly, the required amount of random words for this algorithm is lower than that of the algorithm from Section A (given in Theorem 48 and Corollary 50).

▶ **Theorem 52.** *Consider the Erdös-Rényi $G(n, p)$ model. For* NEXT-NEIGHBOR *queries only, Algorithm 6 is an implementation that answers each query using $O(\log^2 n)$ time, $O(\log n)$ additional space, and one N-bit random word. For* NEXT-NEIGHBOR *and* VERTEX PAIR *queries,an extension of Algorithm 6 answers each query using $O(\log^2 n)$ amortized time, $O(\log n)$ additional space, and one N-bit random word.*

C.3 Data structure for the Stochastic Block model

We employ the data structure for generating and counting the number of vertices of each community in a specified range from Section 3.4.2. We create r different copies of the data structure $\mathbf{R}$ and Q'_v, one for each community, so that we may implement the required operations separately for each color, including using the COUNT subroutine to sample $F \sim$ ExactF via the corresponding CDF, and picking the next neighbor according to F. Recall that since we do not store $\textbf{last}[v] = 0$ in $\mathbf{R}$, and we only add an entry to K_v, P_v or Q'_v after drawing the corresponding X_{uv}, the communities of the endpoints, which cover all elements stored in these data structures, must have already been determined. Thus, we obtain the following corollary for the Stochastic Block model.

▶ **Corollary 53.** *Consider the Stochastic Block model with randomly-assigned communities. For* NEXT-NEIGHBOR *queries only, Algorithm 6 is an implementation that answers each query using $O(r \operatorname{poly}(\log n))$ time, random words, and additional space per query. For* NEXT-NEIGHBOR *and* VERTEX-PAIR *queries, Algorithm 6 answers each query using $O(r \operatorname{poly}(\log n))$ amortized time, $O(r \operatorname{poly}(\log n))$ random words, and $O(r \operatorname{poly}(\log n))$ additional space per query additional space, and one N-bit random word.*

D Sampling from the Multivariate Hypergeometric Distribution

Consider the following random experiment. Suppose that we have an urn containing $B \leq n$ marbles (representing vertices), each occupies one of the r possible colors (representing communities) represented by an integer from $[r]$. The number of marbles of each color in the

urn is known: there are C_k indistinguishable marbles of color $k \in [r]$, where $C_1 + \cdots + C_r = B$. Consider the process of drawing $\ell \leq B$ marbles from this urn *without replacement*. We would like to sample how many marbles of each color we draw.

More formally, let $\mathbf{C} = \langle c_1, \ldots, c_r \rangle$, then we would like to (approximately) sample a vector $\mathbf{S}_\ell^{\mathbf{C}}$ of r non-negative integers such that

$$\Pr[\mathbf{S}_\ell^{\mathbf{C}} = \langle s_1, \ldots, s_r \rangle] = \frac{\binom{C_1}{s_1} \cdot \binom{C_2}{s_2} \cdots \binom{C_r}{s_r}}{\binom{B}{C_1 + C_2 + \cdots + C_r}}$$

where the distribution is supported by all vectors satisfying $s_k \in \{0, \ldots, C_k\}$ for all $k \in [r]$ and $\sum_{k=1}^{r} s_k = \ell$. This distribution is referred to as the *multivariate hypergeometric distribution*.

The sample $\mathbf{S}_\ell^{\mathbf{C}}$ above may be generated easily by simulating the drawing process, but this may take $\Omega(\ell)$ iterations, which have linear dependency in n in the worst case: $\ell = \Theta(B) = \Theta(n)$. Instead, we aim to generate such a sample in $O(r \operatorname{poly}(\log n))$ time with high probability. We first make use of the following procedure from [26].

▶ **Lemma 54.** *Suppose that there are T marbles of color 1 and $B - T$ marbles of color 2 in an urn, where $B \leq n$ is even. There exists an algorithm that samples $\langle s_1, s_2 \rangle$, the number of marbles of each color appearing when drawing $B/2$ marbles from the urn without replacement, in $O(\operatorname{poly}(\log n))$ time and random words. Specifically, the probability of sampling a specific pair $\langle s_1, s_2 \rangle$ where $s_1 + s_2 = T$ is approximately $\binom{B/2}{s_1}\binom{B/2}{T-s_1}/\binom{B}{T}$ with error of at most n^{-c} for any constant $c > 0$.*

In other words, the claim here only applies to the two-color case, where we sample the number of marbles when drawing exactly half of the marbles from the entire urn ($r = 2$ and $\ell = B/2$). First we generalize this claim to handle any desired number of drawn marbles ℓ (while keeping $r = 2$).

▶ **Lemma 55.** *Given C_1 marbles of color 1 and $C_2 = B - C_1$ marbles of color 2, there exists an algorithm that samples $\langle s_1, s_2 \rangle$, the number of marbles of each color appearing when drawing ℓ marbles from the urn without replacement, in $O(\operatorname{poly}(\log B))$ time and random words.*

Proof. For the base case where $B = 1$, we trivially have $\mathbf{S}_1^{\mathbf{C}} = \mathbf{C}_1$ and $\mathbf{S}_0^{\mathbf{C}} = \mathbf{C}_2$. Otherwise, for even B, we apply the following procedure.

- If $\ell \leq B/2$, generate $\mathbf{C}' = \mathbf{S}_{B/2}^{\mathbf{C}}$ using Lemma 54.
  - If $\ell = B/2$ then we are done.
  - Else, for $\ell < B/2$ we recursively generate $\mathbf{S}_\ell^{\mathbf{C}'}$.
- Else, for $\ell > B/2$, we generate $\mathbf{S}_{B-\ell}^{\mathbf{C}'}$ as above, then output $\mathbf{C} - \mathbf{S}_{B-\ell}^{\mathbf{C}'}$.

On the other hand, for odd B, we simply simulate drawing a single random marble from the urn before applying the above procedure on the remaining $B - 1$ marbles in the urn. That is, this process halves the domain size B in each step, requiring $\log B$ iterations to sample $\mathbf{S}_\ell^{\mathbf{C}}$. ◀

Lastly we generalize to support larger r.

▶ **Theorem 21.** *Given B marbles of r different colors, such that there are C_i marbles of color i, there exists an algorithm that samples $\langle s_1, s_2, \cdots, s_r \rangle$, the number of marbles of each color appearing when drawing l marbles from the urn without replacement, in $O(r \cdot \operatorname{poly}(\log B))$ time and random words.*

Proof. Observe that we may reduce $r > 2$ to the two-color case by sampling the number of marbles of the first color, collapsing the rest of the colors together. Namely, define a pair $\mathbf{D} = \langle C_1, C_2 + \cdots + C_r \rangle$, then generate $\mathbf{S}_\ell^{\mathbf{D}} = \langle s_1, s_2 + \ldots + s_r \rangle$ via the above procedure. At this point we have obtained the first entry s_1 of the desired $\mathbf{S}_\ell^{\mathbf{C}}$. So it remains to generate the number of marbles of each color from the remaining $r - 1$ colors in $\ell - s_1$ remaining draws. In total, we may generate $\mathbf{S}_\ell^{\mathbf{C}}$ by performing r iterations of the two-colored case. The error in the L_1-distance may be established similarly to the proof of Lemma 51. ◀

▶ **Theorem 56.** *Given B marbles of r different colors in $[r]$, such that there are C_i marbles of color i and a parameter $k \le r$, there exists an algorithm that samples $s_1 + s_2 + \cdots + s_k$, the number of marbles among the first k colors appearing when drawing ℓ marbles from the urn without replacement, in $O(\mathrm{poly}(\log B))$ time and random words.*

Proof. Since we don't have to find the individual counts, we can be more efficient by grouping half the colors together at each step. Formally, we define a pair $\mathbf{D} = \langle D_1, D_2 \rangle$ where $D_1 = C_1 + C_2 + \cdots + C_{r/2}$ and $D_2 = C_{r/2+1} + \cdots + C_{r-1} + C_r$. We then generate $\langle D_1', D_2' \rangle = \mathbf{S}_\ell^{\mathbf{D}}$.

- If $k < r/2$, we recursively solve the problem with the first $r/2$ colors, $B \leftarrow D_1'$, and the original value of k.
- If $k > r/2$, we recurse on the last $r/2$ colors, B set to D_2', and k set to $k - r/2$. In this case, we add D_1' to the returned value.
- Otherwise, $k = r/2$ and we can return D_1'.

The number of recursive calls is $\mathcal{O}(\log r) = \mathcal{O}(\log B)$ (since $r \le B$). So, the overall runtime is $\mathcal{O}(\mathrm{poly}(\log B))$. ◀

E Local-Access Implementations for Random Directed Graphs

In this section, we consider Kleinberg's Small-World model [31, 37] where the probability that a *directed* edge (u, v) exists is $\min\{c/(\textsc{dist}(u, v))^2, 1\}$. Here, $\textsc{dist}(u, v)$ is the Manhattan distance between u and v on a $\sqrt{n} \times \sqrt{n}$ grid. We begin with the case where $c = 1$, then generalize to different values of $c = \log^{\pm\Theta(1)}(n)$. We aim to support ALL-NEIGHBORS queries using $\mathrm{poly}(\log n)$ resources. This returns the entire list of out-neighbors of v.

E.1 Implementation for $c = 1$

Observe that since the graphs we consider here are directed, the answers to the ALL-NEIGHBOR queries are all independent: each vertex may determine its out-neighbors independently. Given a vertex v, we consider a partition of all the other vertices of the graph into sets $\{\Gamma_1^v, \Gamma_2^v, \ldots\}$ by distance: $\Gamma_k^v = \{u : \textsc{dist}(v, u) = k\}$ contains all vertices at a distance k from vertex v. Observe that $|\Gamma_k^v| \le 4k = O(k)$. Then, the expected number of edges from v to vertices in Γ_k^v is therefore $|\Gamma_k^v| \cdot 1/k^2 = O(1/k)$. Hence, the expected degree of v is at most $\sum_{k=1}^{2(\sqrt{n}-1)} O(1/k) = O(\log n)$. It is straightforward to verify that this bound holds with high probability (use Hoeffding's inequality). Since the degree of v is small, in this model we can afford to perform ALL-NEIGHBORS queries instead of NEXT-NEIGHBOR queries using an additional $\mathrm{poly}(\log n)$ resources.

Nonetheless, internally in our implementation, we generate our neighbors one-by-one similarly to how we process NEXT-NEIGHBOR queries. We perform our sampling in two phases. In the first phase, we compute a distance d, such that the next neighbor closest to v is at distance d. We maintain $\mathbf{last}[v]$ to be the last computed distance. In the second phase, we

generate all neighbors of v at distance d, under the assumption that there must be at least one such neighbor. For simplicity, we generate these neighbors as if there are *full* $4d$ vertices at distance d from v: some generated neighbors may lie outside our $\sqrt{n} \times \sqrt{n}$ grid, which are simply discarded. As the running time of our implementation is proportional to the number of implementation neighbors, then by the bound on the number of neighbors, this assumption does not asymptotically worsen the performance of the implementation.

E.1.1 Phase 1: Generate the distance D

Let $a = \mathbf{last}[v] + 1$, and let $\mathsf{D}(a)$ to denote the probability distribution of the distance where the next closest neighbor of v is located, or $\perp$ if there is no neighbor at distance at most $2(\sqrt{n} - 1)$. That is, if $D \sim \mathsf{D}(a)$ is drawn, then we proceed to Phase 2 to generate all neighbors at distance D. We repeat the process by sampling the next distance from $\mathsf{D}(a + D)$ and so on until we obtain $\perp$, at which point we return our answers and terminate.

To generate the next distance, we perform a binary search: we must evaluate the CDF of $\mathsf{D}(a)$. The CDF is given by $\mathbb{P}[D \le d]$ where $D \sim \mathsf{D}(a)$, the probability that there is *some* neighbor at distance at most d. As usual, we compute the probability of the negation: there is *no* neighbor at distance at most d. Recall that each distance i has exactly $|\Gamma_i^v| = 4i$ vertices, and the probability of a vertex $u \in \Gamma_i^v$ is not a neighbor is exactly $1 - 1/i^2$. So, the probability that there is no neighbor at distance i is $(1 - 1/i^2)^{4i}$. Thus, for $D \sim \mathsf{D}(a)$ and $d \le 2(\sqrt{n} - 1)$,

$$\mathbb{P}[D \le d] = 1 - \prod_{i=a}^{d} \left(1 - \frac{1}{i^2}\right) = 1 - \prod_{i=a}^{d} \left(\frac{(i-1)(i+1)}{i^2}\right)^{4i} = 1 - \left(\frac{(a-1)^a}{a^{a-1}} \cdot \frac{(d+1)^d}{d^{d+1}}\right)^4$$

where the product enjoys telescoping as the denominator $(i^2)^{4i}$ cancels with $(i^2)^{4(i-1)}$ and $(i^2)^{4(i+1)}$ in the numerators of the previous and the next term, respectively. This gives us a closed form for the CDF, which we can compute with 2^{-N} additive error in constant time (by our computation model assumption). Thus, we may generate the distance $D \sim \mathsf{D}(a)$ using $O(\log n)$ time and one random N-bit word.

E.1.2 Phase 2: Sampling neighbors at distance D

After sampling a distance D, we now have to generate all the neighbors at distance D. We label the vertices in Γ_D^v with unique indices in $\{1, \ldots, 4D\}$. Note that now each of the $4D$ vertices in Γ_D^v is a neighbor with probability $1/D^2$. However, by Phase 1, this is conditioned on the fact that there is at least one neighbor among the vertices in Γ_D^v, which may be difficult to generate when $1/D^2$ is very small. We can emulate this naïvely by repeatedly sampling a "block", composing of the $4D$ vertices in Γ_D^v, by deciding whether each vertex is a neighbor of v with uniform probability $1/D^2$ (i.e., $4D$ identical independent Bernoulli trials), and then discarding the entire block if it contains no neighbor. We repeat this process until we finally generate one block that contains at least one neighbor, and use this block as our output.

For the purpose of making the sampling process more efficient, we view this process differently. Let us imagine that we are given an infinite sequence of independent Bernoulli variables, each with bias $1/D^2$. We then divide the sequence into contiguous blocks of length $4D$ each. Our task is to find the *first* occurrence of success (a neighbor), then report the whole block hosting this variable.

This first occurrence of a successful Bernoulli trial is given by sampling from the geometric distribution, $X \sim \mathsf{Geo}(1/D^2)$. Since the vertices in each block are labeled by $1, \ldots, 4D$, then this first occurrence has label $X' = X \bmod 4D$. By sampling $X \sim \mathsf{Geo}(1/D^2)$, the first X'

Bernoulli variables of this block is also implicitly determined. Namely, the vertices of labels $1, \ldots, X' - 1$ are non-neighbors, and that of label X' is a neighbor. The sampling for the remaining $4D - X'$ vertices can then be performed in the same fashion we generate next neighbors in the $G(n, p)$ case: repeatedly find the next neighbor by sampling from $\mathsf{Geo}(1/D^2)$, until the index of the next neighbor falls beyond this block.

Thus at this point, we have generated all neighbors in Γ_D^v. We can then update $\mathbf{last}[v] \leftarrow D$ and continue the process of larger distances. Sampling each neighbor takes $O(\log n)$ time and one random N-bit word; the resources spent sampling the distances is also bounded by that of the neighbors. As there are $O(\log n)$ neighbors with high probability, we obtain the following theorem.

▶ **Theorem 57.** *There exists an algorithm that generates a random graph from Kleinberg's Small World model, where probability of including each directed edge (u, v) in the graph is $1/(\textsc{dist}(u, v))^2$ where $\textsc{dist}$ denote the Manhattan distance, using $O(\log^2 n)$ time and random $\log n$-bit words per $\textsc{All-Neighbors}$ query with high probability.*

E.2 Implementation for $c \neq 1$

Observe that to support different values of c in the probability function $c/(\textsc{dist}(u, v))^2$, we do not have a closed-form formula for computing the CDF for Phase 1, whereas the process for Phase 2 remains unchanged. To handle the change in the probability distribution Phase 1, we consider the following, more general problem. Suppose that we have a process P that, one-by-one, provide occurrences of successes from the sequence of independent Bernoulli trials with success probabilities $\langle p_1, p_2, \ldots \rangle$. We show how to construct a process $\mathcal{P}^c$ that provide occurrences of successes from Bernoulli trials with success probabilities $\langle c \cdot p_1, c \cdot p_2, \ldots \rangle$ (truncated down to 1 as needed). For our application, we assume that c is given in N-bit precision, there are $O(n)$ Bernoulli trials, and we aim for an error of $\frac{1}{\mathrm{poly}(n)}$ in the L_1-distance.

E.2.1 Case $c < 1$

We use rejection sampling in order to construct a new Bernoulli process.

▶ **Lemma 58.** *Given a process $\mathcal{P}$ outputting the indices of successful Bernoulli trials with bias $\langle p_i \rangle$, there exists a process $\mathcal{P}^c$ outputting the indices of successful Bernoulli trials with bias $\langle c \cdot p_i \rangle$ where $c < 1$, using one additional N-bit word overhead for each answer of $\mathcal{P}$.*

Proof. Consider the following rejection sampling process to simulate the Bernoulli trials. In addition to each Bernoulli variable X_i with bias p_i, we generate another coin-flip C_i with bias c. Set $Y_i = X_i \cdot C_i$, then $\mathbb{P}[Y_i = 1] = \mathbb{P}[X_i = 1] \cdot \mathbb{P}[C_i] = c \cdot p_i$, as desired. That is, we keep a success of a Bernoulli trial with probability c, or reject it with probability $1 - c$.

Now, we are already given the process $\mathcal{P}$ that "handles" X_i's, generating a sequence of indices i with $X_i = 1$. The new process $\mathcal{P}^c$ then only needs to handle the C_i's. Namely, for each i reported as success by $\mathcal{P}$, $\mathcal{P}^c$ flips a coin C_i to see if it should also report i, or discard it. As a result, $\mathcal{P}^c$ can generate the indices of successful Bernoulli trials using only one random N-bit word overhead for each answer from $\mathcal{P}$. ◀

Applying this reduction to the distance sampling in Phase 1, we obtain the following corollary.

▶ **Corollary 59.** *There exists an algorithm that generates a random graph from Kleinberg's Small World model with edge probabilities $c/(\textsc{dist}(u, v))^2$ where $c < 1$, using $O(\log^2 n)$ time and random $\log n$-bit words per $\textsc{All-Neighbors}$ query with high probability.*

E.2.2 Case $c > 1$

Since we aim to sample with larger probabilities, we instead consider making $k \cdot c$ independent copies of each process $\mathcal{P}$, where $k > 1$ is a positive integer. Intuitively, we hope that the probability that one of these process returns an index i will be at least $c \cdot p_i$, so that we may perform rejection sampling to decide whether to keep i or not. Unfortunately such a process cannot handle the case where $c \cdot p_i$ is large, notably when $c \cdot p_i > 1$ is truncated down to 1, while there is always a possibility that none of the processes return i.

▶ **Lemma 60.** *Let $k > 1$ be a constant integer. Given a process $\mathcal{P}$ outputting the indices of successful Bernoulli trials with bias $\langle p_i \rangle$, there exists a process $\mathcal{P}^c$ outputting the indices of successful Bernoulli trials with bias $\langle \min\{c \cdot p_i, 1\} \rangle$ where $c > 1$ and $c \cdot p_i \leq 1 - \frac{1}{k}$ for every i, using one additional N-bit word overhead for each answer of $k \cdot c$ independent copies of $\mathcal{P}$.*

Proof. By applying the following form of Bernoulli's inequality, we have

$$(1 - p_i)^{k \cdot c} \leq 1 - \frac{k \cdot c \cdot p_i}{1 + (k \cdot c - 1) \cdot p_i} = 1 - \frac{k \cdot c \cdot p_i}{1 + k \cdot c \cdot p_i - p_i} \leq 1 - \frac{k \cdot c \cdot p_i}{1 + (k - 1)} = 1 - c \cdot p_i$$

That is, the probability that at least one of the implementations report an index i is $1 - (1 - p_i)^{k \cdot c} \geq c \cdot p_i$, as required. Then, the process $\mathcal{P}^c$ simply reports i with probability $(c \cdot p_i)/(1 - (1 - p_i)^{k \cdot c})$ or discard i otherwise. Again, we only require N-bit of precision for each computation, and thus one random N-bit word suffices. ◀

In Phase 1, we may apply this reduction only when the condition $c \cdot p_i \leq 1 - \frac{1}{k}$ is satisfied. For lower value of $p_i = 1/D^2$, namely for distance $D < \sqrt{c/(1 - 1/k)} = O(\sqrt{c})$, we may afford to generate the Bernoulli trials one-by-one as c is $\mathrm{poly}(\log n)$. We also note that the degree of each vertex is clearly bounded by $O(\log n)$ with high probability, as its expectation is scaled up by at most a factor of c. Thus, we obtain the following corollary.

▶ **Corollary 61.** *There exists an algorithm that generates a random graph from Kleinberg's Small World model with edge probabilities $c/(\texttt{DIST}(u, v))^2$ where $c = \mathrm{poly}(\log n)$, using $O(\log^2 n)$ time and random words per* ALL-NEIGHBORS *query with high probability.*

F Omitted Proofs for the Dyck Path Implementation

▶ **Theorem 62.** *There are $\frac{1}{n+1}\binom{2n}{n}$ Dyck paths for length $2n$ (construction from [53]).*

Proof from [53]. Consider all possible sequences containing $n+1$ up-steps and n down-steps with the restriction that the first step is an up-step. We say that two sequences belong to the same *class* if they are cyclic shifts of each other. Because of the restriction, the total number of sequences is $\binom{2n}{n}$ and each class is of size $n + 1$. Now, within each class, exactly one of the sequences is such that the prefix sums are *strictly greater* than zero. From such a sequence, we can obtain a Dyck sequence by deleting the first up-step. Similarly, we can start with a Dyck sequence, add an initial up-step and consider all $n + 1$ cyclic shifts to obtain a *class*. This bijection shows that the number of Dyck paths is $\frac{1}{n+1}\binom{2n}{n}$. ◀

F.1 Approximating Close-to-Central Binomial Coefficients

We start with Stirling's approximation which states that

$$m! = \sqrt{2\pi m}\left(\frac{m}{e}\right)^m \left(1 + \mathcal{O}\left(\frac{1}{m}\right)\right)$$

We will also use the logarithm approximation when a better approximation is required:

$$\log(m!) = m\log m - m + \frac{1}{2}\log(2\pi m) + \frac{1}{12m} - \frac{1}{360m^3} + \frac{1}{1260m^5} - \cdots \tag{12}$$

This immediately gives us an asymptotic formula for the central binomial coefficient as:

▶ **Lemma 63.** *The central binomial coefficient can be approximated as:*

$$\binom{n}{n/2} = \sqrt{\frac{2}{\pi n}}2^n\left(1+\mathcal{O}\left(\frac{1}{n}\right)\right)$$

Now, we consider a "off-center" Binomial coefficient $\binom{n}{k}$ where $k = \frac{n+c\sqrt{n}}{2}$.

▶ **Lemma 64.**

$$\binom{n}{k} = \binom{n}{n/2}e^{-c^2/2}exp\left(\mathcal{O}(c^3/\sqrt{n})\right)$$

Proof from [52]. We consider the ratio: $R = \binom{n}{k}/\binom{n}{n/2}$:

$$R = \frac{\binom{n}{k}}{\binom{n}{n/2}} = \frac{(n/2)!(n/2)!}{k!(n-k)!} = \prod_{i=1}^{c\sqrt{n}/2}\frac{n/2-i+1}{n/2+i} \tag{13}$$

$$\implies \log R = \sum_{i=1}^{c\sqrt{n}/2}\log\left(\frac{n/2-i+1}{n/2+i}\right) \tag{14}$$

$$= \sum_{i=1}^{c\sqrt{n}/2} -\frac{4i}{n} + \mathcal{O}\left(\frac{i^2}{n^2}\right) = -\frac{c^2 n}{2n} + \mathcal{O}\left(\frac{(c\sqrt{n})^3}{n^2}\right) = -\frac{c^2}{2} + \mathcal{O}\left(\frac{c^3}{\sqrt{n}}\right) \tag{15}$$

$$\implies \binom{n}{k} = \binom{n}{n/2}e^{-c^2/2}exp\left(\mathcal{O}(c^3/\sqrt{n})\right) \tag{16}$$

◀

F.2 Dyck Path Boundaries and Deviations

▶ **Lemma 65.** *Given a random walk of length $2n$ with exactly n up and down steps, consider a contiguous* sub-path *of length $2B$ that comprises of U up-steps and D down-steps i.e. $U + D = 2B$. Both $|B - U|$ and $|B - D|$ are $\mathcal{O}(\sqrt{B\log n})$ with probability at least $1 - 1/n^4$.*

Proof. We consider the random walk as a sequence of unbiased random variables $\{X_i\}_{i=1}^{2n} \in \{0,1\}^{2n}$ with the constraint $\sum_{i=1}^{2n} X_i = n$. Here, 1 corresponds to an up-step and 0 corresponds to a down step. Because of the constraint, X_i, X_j are negatively correlated for $i \neq j$ which allows us to apply Chernoff bounds. Now we consider a sub-path of length $2B$ and let U denote the sum of the X_is associated with this subpath. Using Chernoff bound with $\mathbb{E}[X] = B$, we get:

$$\mathbb{P}\left[|U-B| < 3\sqrt{B\log n}\right] = \mathbb{P}\left[|U-B| < 3\frac{\sqrt{\log n}}{\sqrt{B}}B\right] < e^{-\frac{9\log n}{3}} = \frac{1}{n^3}$$

Since U and D are symmetric, the same argument applies. ◀

▶ **Corollary 66.** *With high probability, every contiguous sub-path of length $2B$ (with U up and D down steps such that $U + D = 2B$) in the random walk satisfies the property that $|B - U|$ and $|B - D|$ are upper bounded by $c\sqrt{B\log n}$ w.h.p. $1 - 1/n^2$ (for some constant c).*

Proof. We can simply apply Lemma 65 and union bound over all n^2 possible contiguous sub-paths. ◀

▶ **Lemma 22.** *Consider a contiguous* sub-path *of a simple Dyck path of length $2n$ where the sub-path is of length $2B$ comprising of U up-steps and D down-steps (with $U + D = 2B$). Then there exists a constant c such that the quantities $|B - U|$, $|B - D|$, and $|U - D|$ are all $< c\sqrt{B \log n}$ with probability at least $1 - 1/n^2$ for every possible sub-path.*

Proof. As a consequence of Theorem 62, we can sample a Dyck path by first sampling a *balanced* random walk with n up steps and n down steps and adding an initial up step. We can then find the corresponding Dyck path by taking the unique cyclic shift that satisfies the Dyck constraint (after removing the initial up-step). Any interval in a cyclic shift is the union of at most two intervals in the original sequence. This affects the bound only by a constant factor. So, we can simply use Corollary 66 to finish the proof. Notice that since $|U - D| \leq |B - U| + |B - D|$, $|U - D| = \mathcal{O}(\sqrt{B \log n})$ comes for free. ◀

▶ **Lemma 23.** *Given a Dyck path sampling problem of length B with U up, D down steps, and a boundary at k, there exists a constant c such that if $k > c\sqrt{B \log n}$, then the distribution of paths sampled without a boundary $\mathsf{C}_\infty(U, D)$ is $\mathcal{O}(1/n^2)$-close in L_1 distance to the distribution of Dyck paths $\mathsf{C}_k(U + D)$.*

Proof. We use $\mathcal{D}$ and $\mathcal{R}$ to denote the set of all valid Dyck paths and all random sequences respectively. Clearly, $\mathcal{D} \subseteq \mathcal{R}$. Since the random walk/sequence distribution is uniform on $\mathcal{R}$, and by Corollary 66 we see that at least $1 - 1/n^2$ fraction of the elements of $\mathcal{R}$ do not violate the boundary constraint. Therefore, $|\mathcal{D}| \geq (1 - 1/n^2)|\mathcal{R}|$, and so the L_1 distance between the uniform distributions on $\mathcal{D}$ and $\mathcal{R}$ is $\mathcal{O}(1/n^2)$. ◀

F.3 Estimating the Sampling Probabilities

▶ **Lemma 67.** *Given a Dyck sub-path problem within a global Dyck path of size $2n$ and a probability expression of the form $p_d = \frac{S_{left} \cdot S_{right}}{S_{total}}$, there exists a* $\text{poly}(\log n)$ *time oracle that returns a $\left(1 \pm 1/n^2\right)$ multiplicative approximation to p_d if $p_d = \Omega(1/n^2)$ and returns* 0 *otherwise.*

Proof. We first compute a $1 + 1/n^3$ multiplicative approximation to $\ln p_d$. Using $\mathcal{O}(\log n)$ terms of the series in Equation 12, it is possible to estimate the logarithm of a factorial up to $1/n^c$ additive error. So, we can use the series expansion from Equation 12 up to $\mathcal{O}(\log n)$ terms. The additive error can also be cast as multiplicative since factorials are large positive integers.

The probability p_d can be written as an arithmetic expression involving sums and products of a constant number of factorial terms. Given a $1 \pm 1/n^c$ multiplicative approximation to $l_a = \ln a$ and $l_a = \ln b$, we wish to approximate $\ln(ab)$ and $\ln(a + b)$. The former is trivial since $\ln(ab) = l_a + l_b$. For the latter, we assume $a > b$ and use the identity $\ln(a + b) = \ln a + \ln(1 + b/a)$ to note that it suffices to approximate $\ln(1 + b/a)$. We define $\widetilde{l_a} = l_a \cdot (1 \pm \mathcal{O}(1/n^c))$ and $\widetilde{l_b} = l_b \cdot (1 \pm \mathcal{O}(1/n^c))$. In case $\widetilde{l_b} - \widetilde{l_a} < -c \ln n \implies b/a < 1/n^c$, we approximate $\ln(a + b)$ by $\ln a$ since $\ln(1 + b/a) = \mathcal{O}(1/n^c)$ in this case.

Otherwise, we notice that $\max(a, b) = \mathcal{O}\left((n!)^2\right) \implies \max(l_a, l_b) = o(n^3) \implies l_a - l_b = o(n^3)$. This is true because if we write out the expression for $p_d = S_{left} \cdot S_{right}/S_{total}$ in terms of sums and products of factorials, the largest possible value will be a product of at most two terms that are present in the numerator/denominator of a binomial term (see the expression

for generalized Catalan numbers in Equation 1). Hence, the claim $\max(a,b) = \mathcal{O}\left((n!)^2\right)$ follows from the fact that the numerators/denominators of the relevant binomial coefficient in Equation 1 are at most $n!$. Using this fact, we obtain the following:

$$1 + e^{\widetilde{l}_b - \widetilde{l}_a} = 1 + \frac{b}{a} \cdot e^{\mathcal{O}\left(\frac{l_b - l_a}{n^c}\right)} = 1 + \frac{b}{a} \cdot \left(1 \pm \mathcal{O}\left(\frac{1}{n^{c-3}}\right)\right) = \left(1 + \frac{b}{a}\right) \cdot \left(1 \pm \mathcal{O}\left(\frac{1}{n^{c-3}}\right)\right)$$

In other words, the value of c decreases every time we have a sum operation. Since there are only a constant number of such arithmetic operations in the expression for p_d, we can set c to be a high enough constant when approximating the factorials, cewhich allows us to obtain the desired $1 \pm 1/n^3$ multiplicative approximation to $\ln p_d$. If $\ln p_d < -3 \ln n$, we approximate $p_d = 0$. Otherwise, we can exponentiate the approximation to obtain $\widetilde{p}_d = p_d \cdot e^{-\mathcal{O}(\ln n / n^3)} = p_d \left(1 \pm \mathcal{O}(1/n^2)\right)$. ◀

F.4 Omitted Proofs from Section 4.3: Sampling the Height

▶ **Lemma 68.** *For $x < 1$ and $k \geq 1$, we claim that $1 - kx < (1-x)^k < 1 - kx + \frac{k(k-1)}{2}x^2$*

▶ **Lemma 25.** *$S_{left} \leq c_1 \frac{k \cdot \sqrt{\log n}}{\sqrt{B}} \cdot \binom{B}{D-d}$ for some constant c_1.*

Proof. This involves some simple manipulations.

$$S_{left} = \binom{B}{D-d} - \binom{B}{D-d-k} \tag{17}$$

$$= \binom{B}{D-d} \cdot \left[1 - \frac{(D-d)(D-d-1)\cdots(D-d-k+1)}{(B-D-d+k)(B-D-d+k-1)\cdots(B-D-d+1)}\right] \tag{18}$$

$$\leq \binom{B}{D-d} \cdot \left[1 - \left(\frac{D-d-k+1}{B-D+d+k}\right)^k\right] \tag{19}$$

$$\leq \binom{B}{D-d} \cdot \left[1 - \left(\frac{U+d+k-(U-D+d+k-1)}{U+d+k}\right)^k\right] \tag{20}$$

$$\leq \binom{B}{D-d} \cdot \left[1 - \left(\frac{U+d+k-\mathcal{O}(\sqrt{B\log n})}{U+d+k}\right)^k\right] \tag{21}$$

$$\leq \Theta\left(\frac{k\sqrt{\log n}}{\sqrt{B}}\right) \cdot \binom{B}{D-d} \tag{22}$$

◀

▶ **Lemma 26.** *$S_{right} \leq c_2 \frac{k' \cdot \sqrt{logn}}{\sqrt{B}} \cdot \binom{B}{U-d}$ for some constant c_2.*

Proof.

$$S_{right} = \binom{B}{U-d} - \binom{B}{U-d-k'} \tag{23}$$

$$= \binom{B}{U-d} \cdot \left[1 - \frac{(U-d)(U-d-1)\cdots(U-d-k'+1)}{(B-U-d+k')(B-U-d+k'-1)\cdots(B-U-d+1)}\right] \tag{24}$$

$$\dots \leq \binom{B}{U-d} \cdot \left[1 - \left(\frac{U-d-k'+1}{B-U+d+k'}\right)^{k'}\right] \tag{25}$$

$$\leq \binom{B}{U-d} \cdot \left[1 - \left(\frac{2D-U-d-k+1}{2U-D+k+d}\right)^{k'}\right] \tag{26}$$

$$\leq \binom{B}{U-d} \cdot \left[1 - \left(\frac{U+k+d-(2U-2D+2d+2k-1)}{U+k+d}\right)^{k'}\right] \tag{27}$$

$$\leq \binom{B}{U-d} \cdot \left[1 - \left(\frac{U+k+d-\mathcal{O}(\sqrt{B\log n})}{U+k+d}\right)^{k'}\right] \tag{28}$$

$$\leq \Theta\left(\frac{k'\sqrt{\log n}}{\sqrt{B}}\right) \cdot \binom{B}{U-d} \tag{29}$$

◀

▶ **Lemma 69.** $S_{tot} \geq \binom{2B}{2D} \cdot \left[1 - \left(1 - \frac{k'}{2U+1}\right)^{k}\right]$.

Proof.

$$S_{tot} = \binom{2B}{2D} - \binom{2B}{2D-k} \tag{30}$$

$$= \binom{2B}{2D} \cdot \left[1 - \frac{(2D)(2D-1)\cdots(2D-k+1)}{(2B-2D+k)(2B-2D+k-1)\cdots(2B-2D+1)}\right] \tag{31}$$

$$\geq \binom{2B}{2D} \cdot \left[1 - \left(\frac{2D-k+1}{2B-2D+1}\right)^{k}\right] \tag{32}$$

$$\geq \binom{2B}{2D} \cdot \left[1 - \left(\frac{2U-(2U-2D+k-1)}{2U+1}\right)^{k}\right] \tag{33}$$

$$\geq \binom{2B}{2D} \cdot \left[1 - \left(\frac{(2U+1)-k'}{2U+1}\right)^{k}\right] \tag{34}$$

$$\geq \binom{2B}{2D} \cdot \left[1 - \left(1 - \frac{k'}{2U+1}\right)^{k}\right] \tag{35}$$

$$\tag{36}$$

◀

▶ **Lemma 24.** *When $kk' > 2U+1$, $S_{total} > \frac{1}{2} \cdot \binom{2B}{2D}$.*

Proof. When $kk' > 2U+1 \implies k > \frac{2U+1}{k'}$, we will show that the above expression is greater than $\frac{1}{2}\binom{2B}{2D}$. Defining $\nu = \frac{2U+1}{k'} > 1$, we see that $(1-\frac{1}{\nu})^k \leq (1-\frac{1}{\nu})^{\nu}$. Since this is an increasing function of ν and since the limit of this function is $\frac{1}{e}$, we conclude that

$$1 - \left(1 - \frac{k'}{2U+1}\right)^{k} > \frac{1}{2}$$

◀

▶ **Lemma 27.** *When $kk' \leq 2U+1$, $S_{total} \geq c_3 \frac{k \cdot k'}{B} \cdot \binom{2B}{2D}$ for some constant c_3.*

Proof. Now we bound the term $1 - \left(1 - \frac{k'}{2U+1}\right)^k$, given that $kk' \leq 2U+1 \implies \frac{kk'}{2U+1} \leq 1$. Using Bernoulli's inequality, we know that $(1-x)^n \leq 1/(1+nx)$ for $x \in [0,1]$ and $n \in \mathbb{N}$. Since the term $k'/(2U+1)$ is positive and ≤ 1 (since $kk' \leq 2U+1$), we can apply this as follows:

$$1 - \left(1 - \frac{k'}{2U+1}\right)^k \tag{37}$$

$$\geq 1 - \frac{1}{1 + \frac{kk'}{2U+1}} = \frac{\frac{kk'}{2U+1}}{1 + \frac{kk'}{2U+1}} = \frac{kk'}{2U+1} \cdot \frac{1}{1 + \frac{kk'}{2U+1}} \tag{38}$$

$$\geq \frac{kk'}{2U+1} \cdot \frac{1}{2} \qquad \left(\text{Since } \frac{kk'}{2U+1} \leq 1\right) \tag{39}$$

$$\geq \frac{kk'}{\Theta(B)} \tag{40}$$

◀

F.5 Omitted Proofs from Section 4.4: `First-Return` Queries

▶ **Lemma 31.** *If $d > \log^4 n$, then $S_{left}(d) = \Theta\left(\frac{2^{2d+k}}{\sqrt{d}} e^{-r_{left}(d)} \cdot \frac{k-1}{d+k-1}\right)$ where $r_{left}(d) = \frac{(k-2)^2}{2(2d+k-2)}$. Furthermore, $r_{left}(d) = \mathcal{O}(\log^2 n)$.*

Proof. In what follows, we will drop constant factors: Refer to Figure 8 for the setup. The left section of the path reaches one unit above the boundary (the next step would make it touch the boundary). The number of up-steps on the left side is d and therefore the number of down steps must be $d+k-2$. This inclues d down steps to cancel out the upwards movement, and $k-2$ more to get to one unit above the boundary. The boundary for this section is $k' = k-1$. This gives us:

$$S_{left}(d) = \binom{2d+k-2}{d} - \binom{2d+k-2}{d-1} \tag{41}$$

$$= \binom{2d+k-2}{d}\left[1 - \frac{d}{d+k-1}\right] = \binom{2d+k-2}{d}\frac{k-1}{d+k-1} \tag{42}$$

Now, letting $z = 2d+k-2$, we can write $d = \frac{z-(k-2)}{2} = \frac{z - \frac{k-2}{\sqrt{z}}\sqrt{z}}{2}$. Using Lemma 22, we see that $\frac{k-2}{\sqrt{z}}$ should be $\mathcal{O}(\sqrt{\log n})$. If this is not the case, we can simply return 0 because the probability associated with this value of d is negligible. Since $z > \log^4 n$, we can apply Lemma 64 to get:

$$S_{left}(d) = \Theta\left(\binom{z}{z/2} e^{\frac{(k-2)^2}{2z}} \frac{k-1}{d+k-1}\right) = \Theta\left(\frac{2^{2d+k}}{\sqrt{d}} e^{\frac{(k-2)^2}{2(2d+k-2)}} \frac{k-1}{d+k-1}\right)$$ ◀

▶ **Lemma 32.** *If $U+D-2d-k > \log^4 n$, then $S_{right}(d) = \Theta\left(\frac{2^{U+D-2d-k}}{\sqrt{U+d-2d-k}} e^{-r_{right}(d)} \cdot \frac{U-D+k}{U-d+1}\right)$ where $r_{right}(d) = \frac{(U-D-k-1)^2}{4(U+D-2d-k+1)}$. Furthermore, $r_{right}(d) = \mathcal{O}(\log^2 n)$.*

Proof. The right section of the path starts from the original boundary. Consequently, the boundary for this section is at $k' = 1$. The number of up-steps on the right side is $U - d$ and the number of down steps is $D - d - k + 1$. This gives us:

$$S_{right}(d) = \binom{U + D - 2d - k + 1}{U - d} - \binom{U + D - 2d - k + 1}{U - d + 1} \tag{43}$$

$$= \binom{U + D - 2d - k + 1}{U - d} \left[1 - \frac{D - d - k - 1}{U - d + 1}\right] \tag{44}$$

$$= \binom{U + D - 2d - k + 1}{U - d} \frac{U - D + k}{U - d + 1} \tag{45}$$

Now, letting $z = U + D - 2d - k + 1$, we can write $U - d = \frac{z+(U-D+k-1)}{2} = \frac{z+\frac{U-D+k-1}{\sqrt{z}}\sqrt{z}}{2}$. Using Lemma 22, we see that $\frac{k-2}{\sqrt{z}}$ should be $\mathcal{O}(\sqrt{\log n})$. If this is not the case, we can simply return 0 because the probability associated with this value of d is negligible. Since $z > \log^4 n$, we can apply Lemma 64 to get:

$$S_{right}(d) = \Theta\left(\binom{z}{z/2} e^{\frac{(U-D+k-1)^2}{2z}} \frac{U - D + k}{U - d + 1}\right) \tag{46}$$

$$= \Theta\left(\frac{2^{U+D-2d-k}}{\sqrt{U + D - 2d - k}} e^{\frac{(U-D+k-1)^2}{2(U+D-2d-k+1)}} \frac{U - D + k}{U - d + 1}\right) \tag{47}$$

◀

G Additional related work

Random graph models

The Erdös-Rényi model, given in [17], is one of the most simple theoretical random graph model, yet more specialized models are required to capture properties of real-world data. The Stochastic Block model (or the planted partition model) was proposed in [29] originally for modeling social networks; nonetheless, it has proven to be an useful general statistical model in numerous fields, including recommender systems [33, 49], medicine [51], social networks [21, 44], molecular biology [11, 36], genetics [9, 30, 13], and image segmentation [50]. Canonical problems for this model are the community detection and community recovery problems: some recent works include [12, 40, 3, 2]; see e.g., [1] for survey of recent results. The study of Small-World networks is originated in [55] has frequently been observed, and proven to be important for the modeling of many real world graphs such as social networks [15, 54], brain neurons [6], among many others. Kleinberg's model on the simple lattice topology (as considered in this paper) imposes a geographical that allows navigations, yielding important results such as routing algorithms (decentralized search) [31, 37]. See also e.g., [43] and Chapter 20 of [16].

Generation of random graphs

The problem of local-access implementation of random graphs has been considered in the aforementioned work [24, 41, 18], as well as in [35] that locally generates out-going edges on bipartite graphs while minimizing the maximum in-degree. The problem of generating full graph instances for random graph models have been frequently considered in many models of computations, such as sequential algorithms [39, 7, 45, 38], and the parallel computation model [4].

Query models

In the study of sub-linear time graph algorithms where reading the entire input is infeasible, it is necessary to specify how the algorithm may access the input graph, normally by defining the type of queries that the algorithm may ask about the input graph; the allowed types of queries can greatly affect the performance of the algorithms. While Next-Neighbor query is only recently considered in [18], there are other query models providing a neighbor of a vertex, such as asking for an entry in the adjacency-list representation [28], or traversing to a random neighbor. On the other hand, the Vertex-Pair query is common in the study of dense graphs as accessing the adjacency matrix representation [27]. The All-Neighbors query has recently been explicitly considered in local algorithms [19].

Other constructions of huge pseudorandom functions that are permutations or random hash functions were given in [34, 42, 35].

Monotone Probability Distributions over the Boolean Cube Can Be Learned with Sublinear Samples

Ronitt Rubinfeld
CSAIL at MIT, Cambridge, MA, USA
Blavatnik School of Computer Science at Tel Aviv University, Israel
https://people.csail.mit.edu/ronitt/
ronitt@csail.mit.edu

Arsen Vasilyan
CSAIL at MIT, Cambridge, MA, USA
vasilyan@mit.edu

Abstract

A probability distribution over the Boolean cube is **monotone** if flipping the value of a coordinate from zero to one can only increase the probability of an element. Given samples of an unknown monotone distribution over the Boolean cube, we give (to our knowledge) the first algorithm that learns an approximation of the distribution in statistical distance using a number of samples that is sublinear in the domain.

To do this, we develop a structural lemma describing monotone probability distributions. The structural lemma has further implications to the sample complexity of basic testing tasks for analyzing monotone probability distributions over the Boolean cube: We use it to give nontrivial upper bounds on the tasks of estimating the distance of a monotone distribution to uniform and of estimating the support size of a monotone distribution. In the setting of monotone probability distributions over the Boolean cube, our algorithms are the first to have sample complexity lower than known lower bounds for the same testing tasks on arbitrary (not necessarily monotone) probability distributions.

One further consequence of our learning algorithm is an improved sample complexity for the task of testing whether a distribution on the Boolean cube is monotone.

2012 ACM Subject Classification Theory of computation → Streaming, sublinear and near linear time algorithms; Theory of computation

Keywords and phrases Learning distributions, monotone probability distributions, estimating support size

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.28

Funding *Ronitt Rubinfeld*: FinTech@CSAIL, MIT-IBM Watson AI Lab and Research Collaboration Agreement No. W1771646, and NSF Award Numbers: CCF-1650733, CCF-1740751, CCF-1733808, and IIS-1741137
Arsen Vasilyan: NSF grant IIS-1741137, EECS SuperUROP program

1 Introduction

1.1 Learning Monotone Distributions

Data generated from probability distributions is ubiquitous, and algorithms for understanding such data are of fundamental importance. In particular, a fundamental task is to *learn* an approximation to the probability distribution underlying the data. For probability distributions over huge discrete domains, the sample complexity and run-time bounds for the learning task can be prohibitive. In particular, learning an arbitrary probability distribution on a universe of N_{universe} elements up to sufficiently small constant total variation distance

requires $\Omega(N_{\text{universe}})$ samples. However, when the probability distribution is known to belong to a more structured class of distributions, much better results are possible (cf. [16, 19, 22, 21, 20, 24, 13, 12, 6, 10, 25, 23, 15, 28]) – for example, learning an unknown Poisson binomial distribution up to variation distance ϵ can be achieved with only $\tilde{O}(1/\epsilon^3)$ samples and $\tilde{O}(\log(N_{\text{universe}})/\epsilon^3)$ run-time [13].

A fundamental class of probability distributions is the class of multidimensional monotone probability distributions, which broadly satisfy the following properties:

- The elements of the probability distribution have n different features.
- For every element, an increase in the value of one of the features can only increase its probability.

This basic class of distributions is of great interest because many commonly studied distributions are either monotone or can be approximated by a combination of monotone distributions. Furthermore, often the tools developed for monotone distributions are useful for other classes of distributions: for example, in the one dimensional setting, [12] use tools developed for testing monotone distributions in order to learn k-modal distributions. In [8], tools developed for testing properties of monotone distributions by [4] are used to develop testers for many other classes of distributions.

For the case of only one feature, or equivalently for monotone probability distributions over the totally ordered set $[k]$, a sample-efficient algorithm is known for learning the unknown distribution up total variation distance ϵ with $O(\log(k)/\epsilon^3)$ samples [6, 12]. In [1] it was also shown that an unknown probability distribution over $[k]^n$ can be learned up to χ^2 distance ϵ^2 with $O((n \log k/\epsilon^2)^n/\epsilon^2)$ samples (note that for constant ϵ, this sample complexity is non-trivial only when k is sufficiently large). Overall, the cases considered in the literature specialize on the regime when all the dimensions have a wide range that grows with n. Here we focus on a contrasting case, where each feature has only two possible values, 0 and 1, thus specializing on the Boolean cube:

▶ **Definition 1.** *A probability distribution ρ over $\{0,1\}^n$ is **monotone** if whenever for $x, y \in \{0,1\}^n$ we have that $x \preceq y$ (which means that for all i $x_i \leq y_i$), then we have that $\rho(x) \leq \rho(y)$.*

When studying multi-dimensional objects, focusing on the specific case of the Boolean cube is a common research theme, because the ideas and techniques developed for the Boolean cube are often applicable in the general case. A lower bound of $\Omega(2^{0.15n})$ for learning monotone probability distributions over the Boolean cube (up to sufficiently small constant variation distance) can be inferred from an entropy testing lower bound in [29, page 39] and an argument in [32] (see Claim 17 in Preliminaries). Though the dramatic exponential improvement as in [6, 12] for the totally ordered set is thereby impossible, this still leaves open the possibility of a sublinear sample algorithm for the Boolean cube.

We give, to the best of our knowledge, the first sublinear sample algorithm for learning a monotone probability distribution over the Boolean cube:

▶ **Theorem 2.** *For every positive ϵ, such that $0 < \epsilon \leq 1$ and for all sufficiently large n, there exists an algorithm, which given $\frac{2^n}{2^{\Theta_\epsilon(n^{1/5})}}$ samples from an unknown monotone probability distribution ρ over $\{0,1\}^n$, can reliably return a description of an estimate probability distribution $\hat{\rho}$, such that $d_{TV}(\rho, \hat{\rho}) \leq \epsilon$. The algorithm runs in time $O\left(2^{n+O_\epsilon(n^{1/5}\log n)}\right)$.*

Our algorithm relies on a new structural lemma describing monotone probability distributions on the Boolean cube, as described in Section 1.3. These structural insights also allow us to get improved sample complexity for certain testing tasks on monotone distributions – namely, estimating the closeness of a distribution to uniformity and the support size of the distribution, as presented in Section 1.2.

Theorem 2, together with the L_1 distance tester in [31], can be applied to give the best known sample complexity for testing whether a distribution is monotone. Specifically, one can test whether an unknown distribution ρ over the Boolean cube is monotone or ϵ-far from monotone with $O(\frac{2^n}{n\epsilon^2})$ samples as shown in Claim 18 in Preliminaries. Note that this does not follow from [32] directly, because monotonicity is not a symmetric property. The best previously known algorithm for testing monotonicity over the Boolean cube was presented in [5], requiring $\tilde{O}\left(\frac{2^n}{(n/\log n)^{1/4}}\text{poly}(1/\epsilon)\right)$ samples. The best sample complexity lower bound for testing monotonicity over $\{0,1\}^n$ is $\Omega(2^{(1-\Theta(\sqrt{\epsilon})+o(1))\cdot n})$, as presented in [3]. For the domain $[k]^n$, a monotonicity testing algorithm that requires $O\left(k^{n/2}/\epsilon^2 + \left(\frac{n\log k}{\epsilon^2}\right)^n \cdot \frac{1}{\epsilon^2}\right)$ samples is given and shown to be optimal in [1] (note that this is inapplicable to the Boolean setting, because this sample bound is non-trivial only for sufficiently large k).

1.2 Testing properties of monotone distributions

In addition to learning a distribution, several other basic tasks aimed at understanding distributions have received attention. These include estimating the entropy of a distribution, the size of the support and whether the distribution has certain "shape" properties (monotonicity, convexity, monotone hazard rate, etc.). For arbitrary probability distributions over huge domains, the sample complexity and run-time bounds for the above tasks can be prohibitive, provably requiring $\Omega\left(\frac{N_{\text{universe}}}{\log(N_{\text{universe}})}\right)$ samples. This is true in particular for the properties of support size, entropy and the distance to the uniform distribution [27, 32, 30, 33, 34].

This state of affairs motivates going beyond worst-case analysis and considering common classes of structured probability distributions, a direction that has been considered by many and with a large variety of results (cf. [4, 9, 24, 29, 14, 18]). Some specific examples include: In [4] it is shown that testing whether a monotone distribution is uniform requires only $\Theta(\log^3(N_{\text{universe}})/\epsilon^3)$ samples, in contrast to the $\Theta(\sqrt{N_{\text{universe}}}/\epsilon^2)$ samples required for testing arbitrary distributions for uniformity [26, 11, 17]. The situation is analogous for the tasks of testing whether two distributions given by samples are either the same or far, and testing whether a constant dimensional distribution is independent, which require only polylogarithmic samples if the unknown distributions are promised to be monotone on a total order [4].

Algorithms for testing properties of monotone probability distributions over the Boolean cube were studied in [29, 2]. It was shown that, given samples from a probability distribution over $\{0,1\}^n$ that is promised to be monotone, distinguishing the uniform distribution over $\{0,1\}^n$ from one that is ϵ-far from uniform can be done using only $O\left(\frac{n}{\epsilon^2}\right)$ samples, which is nearly optimal. In contrast, a number of other testing problems cannot have such dramatic improvements when the distribution is known to be monotone: for example in [29] it was shown that for sufficiently small constant ϵ the estimation of entropy up to an additive error of ϵn requires $2^{\Omega(n)}$ samples. However, no nontrivial[1] upper bounds on the sample complexity of any other computational tasks for monotone probability distributions over the Boolean cube are known.

[1] i.e. using monotonicity in an essential way and going beyond the bounds known for arbitrary probability distributions.

1.2.1 Estimating support size

We consider the task of additively estimating the support size of an unknown monotone probability distribution over the Boolean cube. The following assumption is standard in support size estimation:

▶ **Definition 3.** *A probability distribution over a universe of size $N_{universe}$ is called **well-behaved** (in context of support size estimation) if for every x in the set, the probability of x is either zero or at least $1/N_{universe}$.*

The purpose of this definition is to rule out pathological cases in which there are items that are in the support, yet have probability very close to zero. We henceforth adapt this definition to probability distributions over $\{0,1\}^n$, where we have $N_{\text{universe}} = 2^n$. We prove the following theorem:

▶ **Theorem 4.** *For every positive ϵ, the following is true: for all sufficiently large n, there exists an algorithm, which given $\frac{2^n}{2^{\Theta_\epsilon(\sqrt{n})}}$ samples from an unknown well-behaved monotone probability distribution ρ over $\{0,1\}^n$, can reliably[2] approximate the support size of ρ with an additive error of up to ϵ. The algorithm runs in time $O_\epsilon\left(\frac{2^n}{2^{\Theta_\epsilon(\sqrt{n})}}\right)$*

We contrast this result to the results of [27, 30, 31, 32, 34] that show that one needs $\Omega(N_{\text{universe}}/\log(N_{\text{universe}}))$ samples to estimate the support size of an arbitrary distribution up to a sufficiently small constant, which equals to $\Omega(2^n/n)$ for a universe of size 2^n, such as the Boolean cube.

1.2.2 Estimating distance to uniformity

We now consider the task of additively estimating the distance from an unknown monotone probability distribution over the Boolean cube to the uniform distribution. We prove the following theorem:

▶ **Theorem 5.** *For every positive ϵ, the following is true: for all sufficiently large n, there exists an algorithm, which given $\frac{2^n}{2^{\Theta_\epsilon(\sqrt{n})}}$ samples from an unknown monotone probability distribution ρ over $\{0,1\}^n$, can reliably approximate the distance between ρ and the uniform distribution over $\{0,1\}^n$ with an additive error of up to ϵ. The algorithm runs in time $O\left(2^{n+O_\epsilon(\sqrt{n}\log n)}\right)$.*

We, again, contrast this result to the results of [30, 31, 32] that show that one needs $\Omega(N_{\text{universe}}/\log(N_{\text{universe}}))$ samples to estimate the distance of an arbitrary distribution to the uniform distribution, which equals to $\Omega(2^n/n)$ for a universe of size 2^n, such as the Boolean cube.

We also have the following sample complexity lower bound on this task, which we prove using the sub-cube decomposition technique of [29]:

▶ **Theorem 6.** *For infinitely many positive integers n, there exist two probability distributions Δ_{Close} and Δ_{Far} over monotone distributions over $\{0,1\}^n$, satisfying:*

1. *Every distribution in Δ_{Far} is $1/2$-far from the uniform distribution.*
2. *Any algorithm that takes only $o\left(2^{\frac{n^{0.5-0.01}}{2}}\right)$ samples from a probability distribution, fails to reliably distinguish between Δ_{Close} and Δ_{Far}.*
3. *Every distribution in Δ_{Close} is $o(1)$-close to the uniform distribution.*

[2] By **reliably** we henceforth mean that the probability of success is at least $2/3$.

▶ Remark 7. In our construction, the distribution Δ_{Close} consists of only one probability distribution. Additionally, the constant 0.01 can be made arbitrarily small.

Recall that in [29, 2] it was shown that, given samples from a probability distribution over the Boolean cube that is promised to be monotone, distinguishing the uniform distribution from one that is ϵ-far from uniform can be done using only $O\left(\frac{n}{\epsilon^2}\right)$ samples. Yet, as the theorem above shows, the **tolerant** version of this problem, which requires one to distinguish a distribution that is $o(1)$-close to the uniform from a distribution that is $1/2$-far from uniform, requires $\Omega\left(2^{\frac{n^{0.5-0.01}}{2}}\right)$ samples, which is dramatically greater.

1.3 Technical overview

1.3.1 Structural results

Our analysis applies and builds upon the main structural lemma in [7]. To state it, recall that a **DNF** is a Boolean function that is formed as an OR of ANDs, and it is monotone if there are no negations. Each AND is referred to as a **clause**, with the number of variables in the AND is referred to as the **width** of the clause. Their structural lemma shows that each monotone function can be approximated by a DNF with only a constant number of distinct clause widths. Specifically:

▶ **Lemma 8** (Main Lemma in [7], abridged and restated)**.** *For every positive ϵ, for all sufficiently large n, let f be a monotone Boolean function over the domain $\{0,1\}^n$. There is a function $g = g_1 \vee ... \vee g_t$ with the following properties: (i) $t \leq 2/\epsilon$ (ii) each g_i is a monotone DNF with terms of width exactly k_i (iii) g disagrees with f at no more than $\epsilon \cdot 2^n$ elements of $\{0,1\}^n$ (iv) $g(x) \leq f(x)$ for all x in $\{0,1\}^n$.*

For Theorem 4, we use the lemma above on the indicator function of the support of the probability distribution, which allows us to prove the correctness of our algorithm. For the problems of learning and estimating the distance to uniform, we go a step further and prove an analogous structural lemma for *monotone probability distributions*.

There are some crucial differences between monotone Boolean functions in the setting of Boolean function approximation and monotone probability distributions in our setting. First of all, the basic properties of the two objects are different: a Boolean function always has one of the two values (zero or one), which is usually not the case for a probability distribution, but a probability distribution, summed over $\{0,1\}^n$, has to equal one. Secondly, the relevant notions of a function f_2 being well-approximated by a function f_1 are different: for Boolean functions we bound the fraction of points on which f_1 and f_2 disagree, whereas for monotone probability distributions we would like to bound the L_1 distance between f_1 and f_2.

To overcome these differences, we generalize to the setting of non-Boolean functions the main concept used in the proof of Lemma 8: the concept of a **minterm** of a monotone Boolean function. In [7] the minterm of a monotone Boolean function f is defined as follows:

$$\text{minterm}_f(x) \stackrel{\text{def}}{=} \begin{cases} 1 & \text{if } f(x) = 1 \text{ and for all } y \prec x,\ f(y) = 0 \\ 0 & \text{otherwise} \end{cases}$$

Using this language, the function g in Lemma 8 can be characterized as a function, for which:

$$\sum_{h=0}^{n} \mathbf{1}_{\exists\, x \in \{0,1\}^n:\, ||x||=h \,\wedge\, \text{minterm}_g(x) \neq 0} \leq \frac{2}{\epsilon} \tag{1}$$

We introduce the notion of monotone **slack** that generalizes the notion of a minterm to non-Boolean functions:

$$\text{slack}_f(x) \stackrel{\text{def}}{=} f(x) - \max_{y \prec x} f(y) = f(x) - \max_{y \preceq x \text{ and } ||y||=||x||-1} f(y)$$

With such a definition at hand, one could hope to prove that every monotone probability distribution ρ is well-approximated in the L_1 norm by a monotone function f, for which $\sum_{h=0}^{n} \mathbf{1}_{\exists\, x \in \{0,1\}^n:\, ||x||=h \,\wedge\, \text{slack}_f(x) \neq 0}$ is bounded by a constant independent of n. We were not able to prove such a theorem, and instead we bound a related quantity that can be thought of as the weighted analogue of the expression in Equation 1: $\sum_{h=0}^{n} R_h \cdot \mathbf{1}_{\exists\, x \in \{0,1\}^n:\, ||x||=h \,\wedge\, \text{slack}_f(x) \neq 0}$, where the R_h are positive weights that can be chosen arbitrarily, as long as they satisfy a certain technical condition that ensures that not too many of these weights are too large. Precisely, our main lemma is:

▶ **Lemma 9** (Main Structural Lemma). *For all positive ζ, for all sufficiently large n, the following is true: Let ρ be a monotone probability distribution over $\{0,1\}^n$. Suppose, for each h between 0 and n we are given a positive value R_h, and it is the case that: $\sum_{h=0}^{n} R_h \cdot \frac{\binom{n}{h}}{\sum_{j=h}^{n} \binom{n}{j}} \leq \zeta$*

Then, there exists a positive monotone function f, mapping $\{0,1\}^n$ to positive real numbers, satisfying:

1. *For all x, it is the case that $\rho(x) \geq f(x)$.*
2. *It is the case that: $\sum_{x \in \{0,1\}^n} \rho(x) - f(x) \leq \zeta$*
3. *It is the case that: $\sum_{h=0}^{n} R_h \cdot \mathbf{1}_{\exists\, x \in \{0,1\}^n:\, ||x||=h \,\wedge\, slack_f(x) \neq 0} \leq 1$*

Now, as a corollary, we present a simple special case (proven to be so in Subsection 3.1) that not only illustrates the power of Lemma 9, but also is sufficient for our proof of Theorem 5:

▶ **Corollary 10.** *Let ρ be a monotone probability distribution over $\{0,1\}^n$ and let h_0 be an integer for which:*

$$\frac{\epsilon}{4} \leq \Pr_{x \sim \{0,1\}^n}[||x|| \geq h_0] \leq \frac{\epsilon}{2}$$

Then, there exists a positive monotone function $f : \{0,1\}^n \to R$ satisfying:

1. *For all x, it is the case that $\rho(x) \geq f(x)$.*
2. *It is the case that: $\sum_{x \in \{0,1\}^n} \rho(x) - f(x) \leq \frac{\epsilon}{4}$.*
3. *There exists a set of values $\{k_1, ..., k_t\}$ (ordered in an increasing order) with $t \leq \frac{16}{\epsilon^2}$, satisfying that if for some x in $\{0,1\}^n$ we have $||x|| < h_0$ and $slack_f(x) \neq 0$, then $||x|| = k_i$ for some i.*

For Theorem 2, however, we use the full power of Lemma 9.

1.3.2 Algorithmic ideas

Here we present an informal overview of the ideas involved in the design and analysis of our algorithms. Throughout we omit details and technicalities. As already mentioned, our algorithms for Theorems 4, 5 and 2 use respectively Lemma 8, Corollary 10 and Lemma 9 as their structural core. Here we present the algorithmic ideas in the order of increasing technical sophistication.

1.3.2.1 Support size estimation (Theorem 4)

The idea behind our support size estimation algorithm is as follows: if we received x as a sample, then not only x has to be in the support of ρ, but every y, satisfying $x \preceq y$ is in the support of ρ. For all such y, we say that y is **covered** by x. Our algorithms estimates the support size of ρ through estimating the number of all such y that are covered by at least one of the samples.

This algorithm can be made computationally efficient by standard methods in randomized algorithms, and the only non-trivial step is to show that $\frac{2^n}{2^{\Theta_\epsilon(\sqrt{n})}}$ samples suffice. To show this, we first apply Lemma 8 to the indicator function of the support of ρ (which we from now on call the support function of ρ). This gives us a Boolean function g that approximates well the support function of ρ and has zero slack everywhere, except for a small number of levels[3] of $\{0,1\}^n$. For simplicity, assume that the support function of ρ itself has this property, and there are only a small number of levels of the Boolean cube on which the support function of ρ can have non-zero slack, which we call the **slacky** levels.

Now, we divide the elements of $\{0,1\}^n$ (which we also call **points**) into **good**[4] points and **bad** points, with the former defined as all the points sufficiently close to a slacky level, and the latter defined as all the other points. Clearly, a given level of $\{0,1\}^n$ consists either fully from good points or fully from bad points, so we also refer to levels as good or bad.

We argue that if a point y in the support of ρ is a good point, then it is likely to be covered by one of the samples, because there is a large number of values x in the support of ρ, for which $y \preceq x$.

We conclude by bounding the number of elements in the support of ρ that are bad, by using the fact that there cannot be too many slacky levels.

1.3.2.2 Estimation of distance to uniform (Theorem 5)

To estimate the distance of a monotone distribution to uniform, we pick a value h_0 as in Corollary 10 and break down the value of the total variation distance from ρ to uniform into contributions from two disjoint components: (i) $\{x \in \{0,1\}^n \text{ s.t. } ||x|| \geq h_0\}$ and (ii) all the other points of $\{0,1\}^n$. In other words, we use h_0 as the cutoff value for the Hamming weight, to separate $\{0,1\}^n$ into components (i) and (ii). The first contribution is straightforward to estimate simply through estimating how likely a random sample x from ρ is to have $||x|| \geq h_0$, because it is straightforward to prove that if one redistributes the probability mass of ρ in $\{x \in \{0,1\}^n \text{ s.t. } ||x|| \geq h_0\}$, while keeping the total amount of probability mass in this set fixed, the total variation distance between ρ and the uniform distribution cannot change by more than $O_\epsilon(1)$.

For any element x of the component (ii), we prepare an estimate of $\rho(x)$, which we call $\hat{\phi}(x)$. Our approach here is somewhat similar to the one for our support size estimation algorithm. In the case of support size estimation, we only registered whether x was covered by a sample from ρ or not. In this case, we actually need an estimate on $\rho(x)$ (as opposed to $\mathbf{1}_{\rho(x)\neq 0}$) which we obtain by studying the pattern of all the samples covering x. More precisely, suppose we draw N_2 samples from the distribution, which form a multiset S_2. We extract the estimate $\hat{\phi}(x)$ from the pattern of samples as follows:

$$\hat{\phi}(x) := \frac{1}{2^L} \cdot \frac{\max_{y \text{ s.t. } y \preceq x \text{ and } ||x||-||y||=L} \left| \left\{ z \in S_2 : y \preceq z \preceq x \right\} \right|}{N_2}$$

[3] i.e. subsets of $\{0,1\}^n$ that have the same Hamming weight.

[4] We later re-define these notions in order to adapt them for the technical details we ignore in the introduction.

Here L is a parameter equal to $\Theta_\epsilon(\sqrt{n})$. We then estimate the contribution of set (ii) as $\sum_{x\in\{0,1\}^n:\ ||x||\geq h_0} \left|\hat{\phi}(x) - 1/2^n\right|$.

We show the correctness of our algorithm as follows. We use a tail bound to show that $\hat{\phi}(x)$ concentrates sufficiently closely to the value: .

$$\phi(x) \stackrel{\text{def}}{=} \frac{1}{2^L} \cdot \max_{y \text{ s.t. } y\preceq x \text{ and } ||x||-||y||=L} \Pr_{z\sim\rho}[y \preceq z \preceq x] =$$

$$\frac{1}{2^L} \cdot \max_{y \text{ s.t. } y\preceq x \text{ and } ||x||-||y||=L} \sum_{z \text{ s.t. } y\preceq z\preceq x} \rho(z)$$

Then, we apply Corollary 10, which implies that ρ is approximated well by a function f and a certain set of constraints on the slack of f holds.

Now for the sake of simplicity (analogously to the case of support size estimation), assume that ρ itself satisfies the condition that below the threshold h_0 there are at most $O_\epsilon(1)$ levels of $\{0,1\}^n$ on which there are points x with non-zero $\text{slack}_\rho(x)$ (in reality it is merely well-approximated by such a function). We now can (analogously to the case of support size estimation) introduce the concepts of **slacky** levels as levels on which ρ has non-zero slack, and **good** levels, which are below h_0 and farther than L from all **slacky** levels of ρ. Now, one can prove that for x on a good level the value of $\phi(x)$ equals precisely to $\rho(x)$, for the following reasons: First of all the inequality:

$$\max_{y \text{ s.t. } y\preceq x \text{ and } ||x||-||y||=L} \rho(y) \leq \phi(x) \leq \rho(x)$$

follows immediately from the monotonicity of ρ and the definition of ϕ. Secondly, if ρ has no slack on the levels between $||x||$ and $||x|| - L$ (inclusive), then from the definition of slack it follows immediately using induction on L that:

$$\max_{y \text{ s.t. } y\preceq x \text{ and } ||x||-||y||=L} \rho(y) = \rho(x)$$

Therefore, it has to be the case that $\rho(x) = \phi(x)$.

Finally, we bound the contribution to the L_1 distance between $\hat{\phi}$ and ρ of all the levels below h_0 that are not good (which we again call the **bad** levels). We do this by upper-bounding the number of bad levels, and then upper bounding the total probability mass on a single level below h_0.

1.3.2.3 Learning a monotone probability distribution (Theorem 2)

As we saw, our algorithm for the estimation of the distance to the uniform distribution contained a component that learned in L_1 distance the restriction of ρ on the levels below the cutoff h_0. The main challenge here is to extend these ideas to levels above h_0. To this, we make the following changes to our setup:

- Instead of having one fixed constant L defining whether a point is close to a slacky level, we make this value level-dependent. In other words, for every h we define L_h, and then after drawing N samples, which form a multiset S, we compute:

$$\hat{\phi}(x) := \frac{1}{2^{\lfloor L_{||x||}\rfloor}} \cdot \frac{\max_{y \text{ s.t. } y\preceq x \text{ and } ||y||-||x||=\lfloor L_{||x||}\rfloor} \left|\left\{z \in S : y \preceq z \preceq x\right\}\right|}{N}$$

- Instead of using Corollary 10, we use the the full power of Lemma 9. This, again gives us a function f that approximates ρ closely and has a restriction on its slacky levels.

Finally, we pick values of L_h in the algorithm and R_h in the analysis so we balance (i) The random error from the deviation of $\hat{\phi}(x)$ from its expectation and (ii) The systematic error introduced by the slacky levels of f and the levels close to them. As a result, we find that $\frac{2^n}{2^{\Theta(n^{1/5})}}$ samples suffice.

2 Preliminaries

We use the following basic definitions and notation:

▶ **Definition 11.** *For $x \in \{0,1\}^n$, its **Hamming weight** is denoted as $||x||$ and is equal to $\sum_i x_i$.*

▶ **Definition 12.** *For a function $f : \{0,1\}^n \to R$, we define the **average value on level** k (with $0 \le k \le n$) as: $\mu_f(k) = \frac{1}{\binom{n}{k}} \sum_{x \in \{0,1\}^n : ||x|| = k} f(x)$. We also refer to average value on level k for a probability distribution ρ, which we denote $\mu_\rho(k)$. By this we mean the average value on level k of the density function of ρ.*

▶ **Definition 13.** *For a monotone function $f : \{0,1\}^n \to R$, we define the **monotone slack** $slack_f(x)$ at point $x \in \{0,1\}^n$ as follows: $slack_f(x) \stackrel{\text{def}}{=} f(x) - \max_{y \prec x} f(y) = f(x) - \max_{y \preceq x \text{ and } ||y|| = ||x|| - 1} f(y)$. We also stipulate that $slack_f(0^n) = f(0^n)$.*

▶ **Definition 14.** *The **total variation distance** between two probability distributions ρ_1 and ρ_2 is defined as:*
$d_{TV}(\rho_1, \rho_2) \stackrel{\text{def}}{=} \frac{1}{2} \sum_x |\rho_1(x) - \rho_2(x)|$.

The following are well-known facts, which were also used in [7]:

▶ **Fact 15.** *For a monotone function $f : \{0,1\}^n \to R$, for all k_1, k_2 satisfying $0 \le k_1 \le k_2 \le n$, it is the case that $\mu_f(k_1) \le \mu_f(k_2)$.*

▶ **Fact 16.** *For all k, it is the case that $\binom{n}{k} \le \frac{2}{\sqrt{n}} \cdot 2^n$.*

Now, we justify two claims we made in the introduction:

▷ Claim 17. For sufficiently small ϵ_0, for all sufficiently large n, any algorithm that learns an unknown monotone probability distribution over $\{0,1\}^n$ requires at least $\Omega(2^{0.15n})$ samples from the distribution.

Proof. From the argument in [32, pages 1937-1938] it follows that if two probability distributions are ϵ-close in total variation distance, then their entropy values are within $2\log(N_{\text{universe}})\epsilon = 2\epsilon n$. Therefore, the task of estimating the entropy of an unknown monotone probability up to an additive error $2\epsilon n$ is not harder than learning it to withing total variation distance ϵ. But in [29, page 39] it is shown that at least $\sqrt{T}/10$ samples are required for the task of distinguishing whether the unknown monotone probability distribution has entropy at least $0.81n$ or at most $n/2 + \log T$. Picking $T = 2^{0.3n}$ gives us the desired learning lower bound. ◁

▷ Claim 18. Given Theorem 2, one can test whether an unknown distribution ρ over the Boolean cube is monotone or ϵ-far from monotone with $O(\frac{2^n}{n\epsilon^2})$ samples.

Proof. This can be done in the following way: (1) Use our learning algorithm with an error parameter $\epsilon/4$. This gives us a description of a distribution $\hat{\rho}$, which is $\epsilon/4$-close to ρ if ρ is monotone. (2) Estimate, using the estimator of [31], the total variation distance between

ρ and $\hat{\rho}$ up to $\epsilon/4$. If the result is closer to ϵ than to zero, output NO. (3) Compute the total variation distance between $\hat{\rho}$ and the closest monotone probability distribution. If this distance estimate is closer to ϵ than to zero, output NO, otherwise output YES. For constant ϵ, the sample complexity is dominated by step (3), which is $O(\frac{2^n}{\epsilon^2 n})$. It is easy to see that a monotone probability distribution will pass this test, whereas a distribution that is ϵ-far from monotone will fail either step (2) or step (3). ◁

3 Learning monotone probability distributions

Algorithm 1 Algorithm for learning a monotone probability distribution over the Boolean cube (given sample access from a distribution ρ, which is monotone over $\{0,1\}^n$).

1. Set $A := \frac{1}{2n} \cdot e^{\frac{1}{2000} \cdot n^{1/5}}$. For all $h \geq n/2$, set $L_h := \max\left(\log\left(2nA \cdot \frac{\binom{n}{h}}{2^n}\right), 0\right)$
   Similarly, for all h, satisfying $n/2 > h \geq 0$, set: $L_h := L_{n/2} = \log\left(2nA \cdot \frac{\binom{n}{n/2}}{2^n}\right)$.
2. Set $N := \frac{2^n}{A} \cdot \frac{192}{\epsilon^2} \cdot (n + 9\sqrt{n} + 4)$
   Draw N samples from the probability distribution ρ and denote the multiset of these samples as S.
3. For all x in $\{0,1\}^n$, if $||x|| < 9\sqrt{n}$, then set $\hat{\phi}(x) = 0$, otherwise compute:
$$\hat{\phi}(x) := \frac{1}{2^{\lfloor L_{||x||} \rfloor}} \cdot \frac{\max_{y \text{ s.t. } y \preceq x \text{ and } ||y|| - ||x|| = \lfloor L_{||x||} \rfloor} \left| \left\{ z \in S : y \preceq z \preceq x \right\} \right|}{N}$$
   Do this by first making a look-up table, which given arbitrary $z \in \{0,1\}^n$ returns the number of times z was encountered in S. Then, use this look-up table to compute the necessary values of $|\{z \in S : y \preceq z \preceq x\}|$ by querying all these values of z in the lookup table and summing the results up.
4. For all x in $\{0,1\}^n$, compute the following: $\hat{\rho}(x) = \hat{\phi}(x) + \frac{1}{2^n}\left(1 - \sum_{y \in \{0,1\}^n} \hat{\phi}(y)\right)$
5. Output the value table of $\hat{\rho}$.

In this section we prove our upper-bound on the sample complexity of learning an unknown monotone probability distribution over the Boolean cube. We restate the theorem:

▶ **Theorem 2.** *For every positive ϵ, such that $0 < \epsilon \leq 1$ and for all sufficiently large n, there exists an algorithm, which given $\frac{2^n}{2^{\Theta_\epsilon(n^{1/5})}}$ samples from an unknown monotone probability distribution ρ over $\{0,1\}^n$, can reliably return a description of an estimate probability distribution $\hat{\rho}$, such that $d_{TV}(\rho, \hat{\rho}) \leq \epsilon$. The algorithm runs in time $O\left(2^{n + O_\epsilon(n^{1/5} \log n)}\right)$.*

Proof. We present the algorithm as Algorithm 1. The number of samples drawn from ρ is $N = \frac{2^n}{2^{\Theta_\epsilon(n^{1/5})}}$. The run-time, in turn, is dominated by computing the values of $\hat{\phi}$ in step (3), in which the construction of the lookup table takes $O(n \cdot 2^n)$ time, and the time spent computing each $\hat{\phi}(x)$ can be upper bounded by the product of: (i) the number of pairs (y, z) that simultaneously satisfy $y \preceq z \preceq x$ and $||y|| - ||x|| = L_{||x||}$, which can be upper-bounded by $O(n^{L_{||x||}} \cdot 2^{L_{||x||}})$ and (ii) the time it takes to look up a given z in the lookup table, which can be upper-bounded by $O(n)$. Overall, this gives us a run-time upper bound of $O(2^{n + O_\epsilon(n^{1/5} \log n)})$.

Now, the only thing to prove is correctness. Here is our main claim:

▷ Claim 19. If the following conditions are the case:

a) As a function of h, L_h is non-increasing.

b) For all h, we have that $L_h \leq 9\sqrt{n}$.

c)

$$\frac{1}{2^n} \cdot \sum_{h=9\sqrt{n}}^{n} \binom{n}{h} \cdot \frac{A}{2^{L_h}} \leq \frac{1}{2}$$

d)

$$\sum_{h=9\sqrt{n}}^{n} L_h \cdot \left(\begin{cases} \frac{400}{n^{2.5}} & \text{if } h \leq n/2 - \sqrt{n\ln(n)} \\ \frac{40000}{n} & \text{if } n/2 - \sqrt{n\ln(n)} < h < n/2 + \sqrt{n} \\ 40000 \cdot \left(\frac{h-n/2}{n}\right)^2 & \text{if } h \geq n/2 + \sqrt{n} \end{cases} \right) \leq \frac{\epsilon^2}{20000}$$

Then, with probability at least $2/3$, it is the case that $\sum_{x\in\{0,1\}^n \text{ s.t. } 9\sqrt{n}\leq||x||} \left|\hat{\phi}(x) - \rho(x)\right| \leq \frac{\epsilon}{2}$.

We verify in Appendix A, subsection 7.1, that L_h indeed satisfy the conditions above. In fact, the values of L_h and A were chosen specifically to satisfy the constraints above. We prove Claim 19 in Section 3.2, after we develop our main structural lemma in Section 3.1.

We now bound the contribution to the L_1 distance between $\hat{\phi}$ to ρ that comes from points of Hamming weight less than $9\sqrt{n}$. Since $\sum_{x\in\{0,1\}^n} \rho(x) = 1$ and ρ is monotone, then whenever $||x|| \leq n/2$ we have $\rho(x) \leq 1/2^{n/2}$. Therefore, for sufficiently large n we have:

$$\sum_{x\in\{0,1\}^n \text{ s.t. } ||x||<9\sqrt{n}} \left|\hat{\phi}(x) - \rho(x)\right| = \sum_{x\in\{0,1\}^n \text{ s.t. } ||x||<9\sqrt{n}} \rho(x) \leq \frac{n^{9\sqrt{n}}}{2^{n/2}} \leq \frac{\epsilon}{2}$$

Combining this with the bound in Claim 19 we get:

$$\sum_{x\in\{0,1\}^n} \left|\hat{\phi}(x) - \rho(x)\right| \leq \epsilon$$

Overall, we have:

$$2 \cdot d_{\mathrm{TV}}(\rho, \hat{\rho}) = \sum_{x\in\{0,1\}^n} \left|\hat{\rho}(x) - \rho(x)\right| =$$

$$\sum_{x\in\{0,1\}^n} \left|\hat{\phi}(x) - \rho(x) + \frac{1}{2^n}\left(1 - \sum_{y\in\{0,1\}^n} \hat{\phi}(y)\right)\right| \leq$$

$$\sum_{x\in\{0,1\}^n} \left|\hat{\phi}(x) - \rho(x)\right| + \left|1 - \sum_{y\in\{0,1\}^n} \hat{\phi}(y)\right| =$$

$$\sum_{x\in\{0,1\}^n} \left|\hat{\phi}(x) - \rho(x)\right| + \left|\sum_{x\in\{0,1\}^n} \rho(x) - \hat{\phi}(x)\right| \leq 2 \cdot \sum_{x\in\{0,1\}^n} \left|\hat{\phi}(x) - \rho(x)\right| \leq 2 \cdot \epsilon$$

Thus, with probability at least $2/3$, we have $d_{\mathrm{TV}}(\rho, \hat{\rho}) \leq \epsilon$. ◀

3.1 Main lemma

Here we prove the following structural lemma. The lemma, as well as its proof are inspired by the main structural lemma of [7] (i.e. Lemma 8). Recall that the slack of a monotone function was given in Definition 13.

▶ **Lemma 9** (Main Structural Lemma). *For all positive ζ, for all sufficiently large n, the following is true: Let ρ be a monotone probability distribution over $\{0,1\}^n$. Suppose, for each h between 0 and n we are given a positive value R_h, and it is the case that:* $\sum_{h=0}^{n} R_h \cdot \frac{\binom{n}{h}}{\sum_{j=h}^{n}\binom{n}{j}} \leq \zeta$

Then, there exists a positive monotone function f, mapping $\{0,1\}^n$ to positive real numbers, satisfying:

1. *For all x, it is the case that $\rho(x) \geq f(x)$.*
2. *It is the case that: $\sum_{x\in\{0,1\}^n} \rho(x) - f(x) \leq \zeta$*
3. *It is the case that: $\sum_{h=0}^{n} R_h \cdot \mathbf{1}_{\exists\, x\in\{0,1\}^n:\, ||x||=h \,\wedge\, slack_f(x)\neq 0} \leq 1$*

Proof. We use the following process to obtain f:

a) Set $f^* = \rho$.

b) For $h = 0$ to n:

- If it is the case that:

$$\frac{1}{\binom{n}{h}} \cdot \sum_{x\in\{0,1\}^n \text{ s.t. } ||x||=h} \text{slack}_{f^*}(x) < R_h \cdot \frac{1}{\sum_{j=h}^{n}\binom{n}{h}} \tag{2}$$

  Then, for all x in $\{0,1\}^n$, satisfying $||x|| = h$ set: $f^*(x) := f^*(x) - \text{slack}_{f^*}(x)$.

c) Set $f = f^*$ and output f.

By inspection, f^* remains monotone and positive at every iteration of the process. Therefore, f is also monotone and positive.

Property (1) in the Lemma is true, because at every step of the process, values of f^* only decrease.

To see why Property (2) is the case, note that the value $\sum_{x\in\{0,1\}^n} \rho(x) - f(x)$ is zero in the beginning of the process, and at a step h it either stays the same or decreases by at most $R_h \cdot \frac{\binom{n}{h}}{\sum_{j=h}^{n}\binom{n}{j}}$. Therefore we can upper-bound:

$$\sum_{x\in\{0,1\}^n} \rho(x) - f(x) \leq \sum_{h=0}^{n} R_h \cdot \frac{\binom{n}{h}}{\sum_{j=h}^{n}\binom{n}{j}} \leq \zeta$$

Now, the only thing left to prove is that property (3) holds.

From the definition of monotone slack, it follows that modifying the value of a function on points of Hamming weight j does not affect the slack on any point with Hamming weight lower than j. Therefore, the value $\frac{1}{\binom{n}{j}} \cdot \sum_{x\in\{0,1\}^n \text{ s.t. } ||x||=j} \text{slack}_{f^*}(x)$ will not change as f^* changes after the jth iteration. Therefore, this value will be equal to $\frac{1}{\binom{n}{j}} \cdot \sum_{x\in\{0,1\}^n \text{ s.t. } ||x||=j} \text{slack}_{f}(x)$. Thus, the value of $\frac{1}{\binom{n}{j}} \cdot \sum_{x\in\{0,1\}^n \text{ s.t. } ||x||=j} \text{slack}_{f}(x)$ is either zero or at least $R_h \cdot \frac{1}{\sum_{j=h}^{n}\binom{n}{h}}$.

Now, we need the following generalization of Fact 15:

▶ **Observation 20.** *Let f be an arbitrary monotone function $\{0,1\}^n \to R$. Then, for any k in $[0, n-1]$ it is the case that:*

$$\mu_f(k+1) \geq \mu_f(k) + \frac{1}{\binom{n}{k+1}} \cdot \sum_{x\in\{0,1\}^n \text{ s.t. } ||x||=k+1} slack_f(x)$$

Proof. For all x with $||x|| = k+1$ we have that:

$$f(x) = \text{slack}_f(x) + \max_{y \in \{0,1\}^n \text{ s.t. } ||y||=k \text{ and } y \preceq x} f(y)$$

We have that:

$$\max_{y \in \{0,1\}^n \text{ s.t. } ||y||=k \text{ and } y \preceq x} f(y) \geq \mathbb{E}_{y \sim \{0,1\}^n \text{ conditioned on } ||y||=k \text{ and } y \preceq x}[f(y)]$$

Therefore:

$$f(x) \geq \text{slack}_f(x) + \mathbb{E}_{y \sim \{0,1\}^n \text{ conditioned on } ||y||=k \text{ and } y \preceq x}[f(y)]$$

Averaging the both sides, we get:

$$\mu_f(k+1) \geq \frac{1}{\binom{n}{k+1}} \cdot \sum_{x \in \{0,1\}^n \text{ s.t. } ||x||=k+1} \text{slack}_f(x) +$$

$$\mathbb{E}_{x \sim \{0,1\}^n \text{ conditioned on } ||x||=k+1} \mathbb{E}_{y \sim \{0,1\}^n \text{ conditioned on } ||y||=k \text{ and } y \preceq x}[f(y)] =$$

$$\frac{1}{\binom{n}{k+1}} \cdot \sum_{x \in \{0,1\}^n \text{ s.t. } ||x||=k+1} \text{slack}_f(x) + \mathbb{E}_{y \sim \{0,1\}^n \text{ conditioned on } ||y||=k}[f(y)] =$$

$$\frac{1}{\binom{n}{k+1}} \cdot \sum_{x \in \{0,1\}^n \text{ s.t. } ||x||=k+1} \text{slack}_f(x) + \mu_f(k) \quad (3)$$

Above, the penultimate equality followed from a simple probabilistic fact: if one picks a random n-bit string of Hamming weight $k+1$ and then sets to zero a random bit that equals to one, this is equivalent to picking a random n-bit string of weight k. ◀

Using the Observation 20 repeatedly and recalling that in Definition 13 we defined $\text{slack}_f(0^n) = f(0^n)$, we get that for all h:

$$\mu_f(h) \geq \mu_f(0) + \sum_{k=1}^{h} \frac{1}{\binom{n}{k}} \cdot \sum_{x \in \{0,1\}^n \text{ s.t. } ||x||=k} \text{slack}_f(x) =$$

$$\sum_{k=0}^{h} \frac{1}{\binom{n}{k}} \cdot \sum_{x \in \{0,1\}^n \text{ s.t. } ||x||=k} \text{slack}_f(x) \geq \sum_{k=0}^{h} R_k \cdot \frac{1}{\sum_{j=k}^{n} \binom{n}{j}} \cdot \mathbf{1}_{\exists\, x \in \{0,1\}^n :\, ||x||=k \,\wedge\, \text{slack}_f(x) \neq 0}$$

Summing this up over all h and changing the order of summations, we get:

$$1 = \sum_{x \in \{0,1\}^n} \rho(x) \geq \sum_{x \in \{0,1\}^n} f(x) = \sum_{h=0}^{n} \binom{n}{h} \mu_f(h) \geq$$

$$\sum_{h=0}^{n} \binom{n}{h} \sum_{k=0}^{h} R_k \cdot \frac{1}{\sum_{j=k}^{n} \binom{n}{j}} \cdot \mathbf{1}_{\exists\, x \in \{0,1\}^n :\, ||x||=k \,\wedge\, \text{slack}_f(x) \neq 0} =$$

$$\sum_{k=0}^{n} \sum_{h=k}^{n} \binom{n}{h} R_k \cdot \frac{1}{\sum_{j=k}^{n} \binom{n}{j}} \cdot \mathbf{1}_{\exists\, x \in \{0,1\}^n :\, ||x||=k \,\wedge\, \text{slack}_f(x) \neq 0} =$$

$$\sum_{k=0}^{n} R_k \cdot \mathbf{1}_{\exists\, x \in \{0,1\}^n :\, ||x||=k \,\wedge\, \text{slack}_f(x) \neq 0}$$

This finishes the proof of the lemma. ◀

Now, we prove the following corollary:

▶ **Corollary 10.** *Let ρ be a monotone probability distribution over $\{0,1\}^n$ and let h_0 be an integer for which:*

$$\frac{\epsilon}{4} \leq \Pr_{x \sim \{0,1\}^n}[||x|| \geq h_0] \leq \frac{\epsilon}{2}$$

Then, there exists a positive monotone function $f : \{0,1\}^n \to R$ satisfying:

1. *For all x, it is the case that $\rho(x) \geq f(x)$.*
2. *It is the case that: $\sum_{x \in \{0,1\}^n} \rho(x) - f(x) \leq \frac{\epsilon}{4}$.*
3. *There exists a set of values $\{k_1, ..., k_t\}$ (ordered in an increasing order) with $t \leq \frac{16}{\epsilon^2}$, satisfying that if for some x in $\{0,1\}^n$ we have $||x|| < h_0$ and $slack_f(x) \neq 0$, then $||x|| = k_i$ for some i.*

Proof. We use Lemma 9, setting $\zeta = \epsilon/4$ and

$$R_h = \begin{cases} \frac{\epsilon^2}{16} & \text{if } h \leq h_0 \\ 0 & \text{otherwise} \end{cases}$$

We verify the precondition to Lemma 9, by using that $\sum_{x \in \{0,1\}^n} \rho(x) - f(x) \leq \frac{\epsilon}{4}$:

$$\sum_{h=0}^{n} R_h \cdot \frac{\binom{n}{h}}{\sum_{j=h}^{n} \binom{n}{j}} = \sum_{h=0}^{h_0} \frac{\epsilon^2}{16} \cdot \frac{\binom{n}{h}}{\sum_{j=h}^{n} \binom{n}{j}} \leq \sum_{h=0}^{h_0} \frac{\epsilon^2}{16} \cdot \frac{\binom{n}{h}}{\sum_{j=h_0}^{n} \binom{n}{j}} \leq$$

$$\sum_{h=0}^{h_0} \frac{\epsilon^2}{16} \cdot \frac{\binom{n}{h}}{2^n \cdot \epsilon/4} = \frac{\epsilon}{4} \cdot \sum_{h=0}^{h_0} \frac{\binom{n}{h}}{2^n} \leq \frac{\epsilon}{4}$$

Now, we simply check that properties (1), (2) and (3) of the Lemma directly imply the properties (1), (2) and (3) of the Corollary respectively. This completes the proof. ◀

To use Lemma 9, we need an upper bound on the value of $\frac{\binom{n}{h}}{\sum_{j \geq h}^{n} \binom{n}{j}}$. The following claim provides such an upper bound:

▷ Claim 21. For all sufficiently large n, for all h, satisfying $0 \leq h \leq n$, it is the case that:

$$\frac{\binom{n}{h}}{\sum_{j \geq h}^{n} \binom{n}{j}} \leq \left(\begin{cases} \frac{2}{n^2} & \text{if } h \leq n/2 - \sqrt{n \ln(n)} \\ \frac{200}{\sqrt{n}} & \text{if } n/2 - \sqrt{n \ln(n)} < h < n/2 + \sqrt{n} \\ 200 \cdot \frac{h - n/2}{n} & \text{if } h \geq n/2 + \sqrt{n} \end{cases} \right)$$

Proof. See Appendix A, Subsection 7.2 ◁

3.2 Proof of Claim 19

For all x in $\{0,1\}^n$, satisfying $9\sqrt{n} \leq ||x||$, we define the following quantity:

$$\phi(x) \stackrel{\text{def}}{=} \frac{1}{2^{\lfloor L_{||x||} \rfloor}} \cdot \max_{y \text{ s.t. } y \preceq x \text{ and } ||x|| - ||y|| = \lfloor L_{||x||} \rfloor} \Pr_{z \sim \rho}[y \preceq z \preceq x] =$$

$$\frac{1}{2^{\lfloor L_{||x||} \rfloor}} \cdot \max_{y \text{ s.t. } y \preceq x \text{ and } ||x|| - ||y|| = \lfloor L_{||x||} \rfloor} \sum_{z \text{ s.t. } y \preceq z \preceq x} \rho(z) \quad (4)$$

Observe that since for every such x and y there are $2^{\lfloor L_{||x||} \rfloor}$ values of z satisfying $y \preceq z \preceq x$, and ρ is a monotone probability distribution, it has to be the case that $\phi(x) \leq \rho(x)$ for all x on which $\phi(x)$ is defined.

More interestingly, we will be claiming that ϕ is (in terms of L_1 distance) a good approximation to ρ, but first we will show that $\hat{\phi}$ is a good approximation to ϕ, assuming that the values L_h are not too small:

▷ Claim 22. If it is the case that $\frac{1}{2^n} \cdot \sum_{h=9\sqrt{n}}^{n} \binom{n}{h} \cdot \frac{A}{2^{L_h}} \leq \frac{1}{2}$, then, with probability at least $7/8$, it is the case that:

$$\sum_{x \in \{0,1\}^n \text{ s.t. } 9\sqrt{n} \leq ||x||} \left| \hat{\phi}(x) - \phi(x) \right| \leq \frac{\epsilon}{4} \tag{5}$$

Proof. See Appendix A, Subsection 7.3, for the proof, which follows using tail bounds. ◁

Now, we apply Lemma 9 to ρ, with value $\zeta := \epsilon/100$. For now, we postpone setting the values of R_h, which we will do later in our derivation (of course, we will then check that the required constraint is indeed satisfied by these values).

This gives a positive monotone function f that satisfies the three conditions of Lemma 9. We separate all the values of x in $\{0,1\}^n$ for which $9\sqrt{n} \leq ||x||$ into two kinds: **good** and **bad**. We say that x is **bad** if there is some y for which $0 \leq ||x|| - ||y|| < \lfloor L_{||x||} \rfloor$ and $\text{slack}_f(y)$ is non-zero. Otherwise, x if **good**. Clearly, for a given Hamming weight value, wither every point with this Hamming weight is good, or every such point is bad.

We can write:

$$\sum_{x \in \{0,1\}^n \text{ s.t. } 9\sqrt{n} \leq ||x||} |\phi(x) - f(x)| = \sum_{\text{good } x} |\phi(x) - f(x)| + \sum_{\text{bad } x} |\phi(x) - f(x)| \tag{6}$$

Now, we bound the two terms above separately. If x is good, then it is the case that for all y satisfying $||x|| - \lfloor L_{||x||} \rfloor < ||y|| \leq ||x||$ we have $\text{slack}_f(x) = 0$, and therefore $f(y) = \max_{y' \in \{0,1\}^n \text{ s. t. } y' \preceq y \text{ and } ||y|| - ||y'|| = 1} f(y')$. Using this relation recursively, we obtain that:

$$f(x) = \max_{y \in \{0,1\}^n \text{ s. t. } y \preceq x \text{ and } ||x|| - ||y|| = \lfloor L_{||x||} \rfloor} f(y)$$

Therefore, since f is monotone, we obtain that:

$$f(x) = \frac{1}{2^{\lfloor L_{||x||} \rfloor}} \cdot \max_{y \text{ s.t. } y \preceq x \text{ and } ||x|| - ||y|| = \lfloor L_{||x||} \rfloor} \sum_{z \text{ s.t. } y \preceq z \preceq x} f(z)$$

By Lemma 9, it is the case $\rho(x) \geq f(x)$. This, together with the equation above and Equation 4 implies:

$$\phi(x) \geq f(x)$$

But we also know that $\rho(x) \geq \phi(x)$. Therefore:

$$\sum_{\text{good } x} |\phi(x) - f(x)| \leq \sum_{\text{good } x} |\rho(x) - f(x)| \leq \frac{\epsilon}{4} \tag{7}$$

Where the last inequality follows from Lemma 9.

Now, we bound the contribution of bad points. Since $\phi(x) \leq \rho(x)$, $f(x) \leq \rho(x)$ and recalling the definition of a bad point, we get:

$$\sum_{\text{bad } x} |\phi(x) - f(x)| \leq \sum_{\text{bad } x} \max(\phi(x), f(x)) \leq \sum_{\text{bad } x} \rho(x) \leq \sum_{h_2=9\sqrt{n}}^{n} \mu_\rho(h_2) \cdot \binom{n}{h_2} \cdot \mathbf{1}_{\exists\, x\in\{0,1\}^n:\, \left(h_2 - \lfloor L_{h_2} \rfloor < ||x|| \leq h_2\right) \wedge \text{slack}_f(x) \neq 0} \quad (8)$$

Since Lemma 9 gives us a bound on a weighed sum of indicator variables of the form $\mathbf{1}_{\exists\, x\in\{0,1\}^n:\, ||x||=h \wedge \text{slack}_f(x)\neq 0}$, we would like to upper-bound the expression above by such a weighted sum. To do this, to every Hamming weight value h that has a point x with non-zero $\text{slack}_f(x)$ (we call such Hamming weight value h **slacky**) we "charge" every value h_2, for which points of Hamming weight h_2 are rendered bad because h is slacky. This will happen only if $h_2 \geq h$ and $h_2 - \lfloor L_{h_2} \rfloor < h$. But since $\lfloor L_{h_2} \rfloor$ can only decrease as h_2 increases, the latter can happen only if $h_2 - \lfloor L_h \rfloor < h$. Therefore:

$$\sum_{\text{bad } x} |\phi(x) - f(x)| \leq \sum_{h=0}^{n} \left(\sum_{h_2=h}^{h+\lfloor L_h \rfloor - 1} \mu_\rho(h_2) \cdot \binom{n}{h_2} \right) \cdot \mathbf{1}_{\exists\, x\in\{0,1\}^n:\, ||x||=h \wedge \text{slack}_f(x)\neq 0} \quad (9)$$

Now, to upper-bound $\mu_\rho(h_2)$, we need the following claim:

▷ Claim 23. For any monotone probability distribution ρ it is the case that for all h:

$$\mu_\rho(h) \leq \frac{1}{\sum_{j=h}^{n} \binom{n}{j}}$$

Proof. This follows immediately from Fact 15 and that $\sum_{x\in\{0,1\}^n} \rho(x) = 1$. ◁

Claim 23, Equation 8 and Claim 21 together imply:

$$\sum_{\text{bad } x} |\phi(x) - f(x)| \leq \sum_{h=0}^{n} \left(\sum_{h_2=h}^{h+\lfloor L_h \rfloor - 1} \frac{\binom{n}{h_2}}{\sum_{j=h_2}^{n} \binom{n}{j}} \right) \mathbf{1}_{\left(\substack{\exists\, x\in\{0,1\}^n: \\ ||x||=h \wedge \text{slack}_f(x)\neq 0}\right)} \leq$$

$$\sum_{h=0}^{n} \left(\sum_{h_2=h}^{h+\lfloor L_h \rfloor - 1} \left(\begin{cases} \frac{200}{\sqrt{n}} & \text{if } h_2 < n/2 + \sqrt{n} \\ 200 \cdot \frac{h_2 - n/2}{n} & \text{if } h_2 \geq n/2 + \sqrt{n} \end{cases} \right) \right) \mathbf{1}_{\left(\substack{\exists\, x\in\{0,1\}^n: \\ ||x||=h \wedge \text{slack}_f(x)\neq 0}\right)} \leq$$

$$\sum_{h=0}^{n} L_h \cdot \left(\begin{cases} \frac{200}{\sqrt{n}} & \text{if } h + L_h < n/2 + \sqrt{n} \\ 200 \cdot \frac{h + L_h - n/2}{n} & \text{if } h + L_h \geq n/2 + \sqrt{n} \end{cases} \right) \mathbf{1}_{\left(\substack{\exists\, x\in\{0,1\}^n: \\ ||x||=h \wedge \text{slack}_f(x)\neq 0}\right)} \quad (10)$$

Now, we claim that:

$$\left(\begin{cases} \frac{200}{\sqrt{n}} & \text{if } h + L_h < n/2 + \sqrt{n} \\ 200 \cdot \frac{h + L_h - n/2}{n} & \text{if } h + L_h \geq n/2 + \sqrt{n} \end{cases} \right) \leq 10 \cdot \left(\begin{cases} \frac{200}{\sqrt{n}} & \text{if } h < n/2 + \sqrt{n} \\ 200 \cdot \frac{h - n/2}{n} & \text{if } h \geq n/2 + \sqrt{n} \end{cases} \right) \quad (11)$$

This follows by considering three cases (i) $h + L_h < n/2 + \sqrt{n}$, in which case this is equivalent to $\frac{200}{\sqrt{n}} \leq \frac{2000}{\sqrt{n}}$, which is trivially true. (ii) $h \geq n/2 + \sqrt{n}$, in which case since $L_h \leq 9\sqrt{n}$, we have that $\frac{h + L_h - n/2}{n} \leq 10 \cdot \frac{h - n/2}{n}$ (iii) $h + L_h \geq n/2 + \sqrt{n}$, but $h < n/2 + \sqrt{n}$, in which case since $L_h \leq 9\sqrt{n}$, we have that $\frac{h + L_h - n/2}{n} \leq \frac{\sqrt{n} + L_h}{n} \leq 10\sqrt{n}$.

Combining Equations 10 and 11, we get:

$$\sum_{\text{bad } x} |\phi(x) - f(x)| \leq$$
$$\sum_{h=0}^{n} L_h \cdot 10 \cdot \left(\begin{cases} \frac{200}{\sqrt{n}} & \text{if } h < n/2 + \sqrt{n} \\ 200 \cdot \frac{h-n/2}{n} & \text{otherwise} \end{cases} \right) \cdot \mathbf{1}_{\exists\, x \in \{0,1\}^n \colon ||x|| = h \,\wedge\, \text{slack}_f(x) \neq 0} \quad (12)$$

Recall that we postponed setting the values of R_h. The equation above motivates us to set:

$$R_h := \frac{200}{\epsilon} \cdot L_h \cdot \left(\begin{cases} \frac{200}{\sqrt{n}} & \text{if } h < n/2 + \sqrt{n} \\ 200 \cdot \frac{h-n/2}{n} & \text{otherwise} \end{cases} \right)$$

Now, we check the constraint on R_h in Lemma 9. Using Claim 21 and the premise of Claim 19:

$$\sum_{h=0}^{n} R_h \cdot \frac{\binom{n}{h}}{\sum_{j \geq h}^{n} \binom{n}{j}} \leq \sum_{h=0}^{n} R_h \cdot \left(\begin{cases} \frac{2}{n^2} & \text{if } h \leq n/2 - \sqrt{n \ln(n)} \\ \frac{200}{\sqrt{n}} & \text{if } n/2 - \sqrt{n \ln(n)} < h < n/2 + \sqrt{n} \\ 200 \cdot \frac{h-n/2}{n} & \text{if } h \geq n/2 + \sqrt{n} \end{cases} \right) =$$
$$\frac{200}{\epsilon} \cdot \sum_{h=0}^{n} L_h \cdot \left(\begin{cases} \frac{400}{n^{2.5}} & \text{if } h \leq n/2 - \sqrt{n \ln(n)} \\ \frac{40000}{n} & \text{if } n/2 - \sqrt{n \ln(n)} < h < n/2 + \sqrt{n} \\ 40000 \cdot \left(\frac{h-n/2}{n} \right)^2 & \text{if } h \geq n/2 + \sqrt{n} \end{cases} \right) \leq \frac{\epsilon}{100} = \zeta$$

Therefore, Lemma 9, together with Equation 12 implies that:

$$\sum_{\text{bad } x} |\phi(x) - f(x)| \leq \sum_{h=0}^{n} \frac{\epsilon}{20} \cdot R_h \cdot \mathbf{1}_{\exists\, x \in \{0,1\}^n \colon ||x|| = h \,\wedge\, \text{slack}_f(x) \neq 0} \leq \frac{\epsilon}{20}$$

Now, using triangle inequality and then combining the inequality above with Equations 28, 6 and 7 we get:

$$\sum_{x \in \{0,1\}^n \text{ s.t. } 9\sqrt{n} \leq ||x||} \left| \hat{\phi}(x) - \rho(x) \right| \leq$$
$$\sum_{x \in \{0,1\}^n \text{ s.t. } 9\sqrt{n} \leq ||x||} \left| \hat{\phi}(x) - \phi(x) \right| + \sum_{x \in \{0,1\}^n \text{ s.t. } 9\sqrt{n} \leq ||x||} \left| \phi(x) - \rho(x) \right| \leq$$
$$\frac{\epsilon}{4} + \frac{\epsilon}{100} + \frac{\epsilon}{20} \leq \frac{\epsilon}{2} \quad (13)$$

4 Estimating the distance to uniform

In this section we prove our upper-bound on the sample complexity of estimating the distance from uniform of an unknown monotone probability distribution over the Boolean cube. We restate the theorem:

▶ **Theorem 5.** *For every positive ϵ, the following is true: for all sufficiently large n, there exists an algorithm, which given $\frac{2^n}{2^{\Theta_\epsilon(\sqrt{n})}}$ samples from an unknown monotone probability distribution ρ over $\{0,1\}^n$, can reliably approximate the distance between ρ and the uniform distribution over $\{0,1\}^n$ with an additive error of up to ϵ. The algorithm runs in time $O\left(2^{n+O_\epsilon(\sqrt{n}\log n)}\right)$.*

Algorithm 2 Algorithm for the estimation of distance to uniform efficiently (given sample access from a distribution ρ, which is monotone over $\{0,1\}^n$).

1. Pick set h_0 to be an integer for which it is the case that:

$$\frac{\epsilon}{4} \leq \Pr_{x \sim \{0,1\}^n}[||x|| \geq h_0] \leq \frac{\epsilon}{2} \tag{14}$$

   Do this by going through every integer candidate $h_{\text{candidate}}$ in the interval and computing the fraction of points x in $\{0,1\}$ for which $||x|| \geq h_{\text{candidate}}$. Finally, pick h_0 to be one of $h_{\text{candidate}}$ for which the relation above holds.
2. Set $N_1 := \frac{32 \ln 2}{\epsilon^2}$. Draw N_1 samples from the probability distribution ρ and denote the multiset of these samples as S_1.
3. Set:

$$\hat{d}_1 := \frac{1}{2} \cdot \frac{\left|\left\{z \in S_1 : ||z|| \geq h_0\right\}\right|}{N_1}$$

4. Set $L := \left\lfloor \frac{\sqrt{n}\epsilon^4}{512} \right\rfloor$.
5. Set

$$N_2 := \frac{2^n}{2^L} \cdot \frac{192}{\epsilon^2} \cdot \left(n \ln 2 + L \ln n + 4 \ln 2\right)$$

   Draw N_2 samples from the probability distribution ρ and denote the multiset of these samples as S_2.
6. For all x, satisfying $L \leq ||x|| < h_0$, compute:

$$\hat{\phi}(x) := \frac{1}{2^L} \cdot \frac{\max_{y \text{ s.t. } y \preceq x \text{ and } ||x||-||y||=L} \left|\left\{z \in S_2 : y \preceq z \preceq x\right\}\right|}{N_2}$$

   Do this by first making a look-up table, which given arbitrary $z \in \{0,1\}^n$ returns the number of times z was encountered in S_2. Then, use this look-up table to compute the necessary values of $|\{z \in S_2 : y \preceq z \preceq x\}|$ by querying all these values of z in the lookup table and summing the results up.
7. Compute the following:

$$\hat{d}_2 := \frac{1}{2} \cdot \sum_{x \text{ s.t. } L \leq ||x|| < h_0} \left|\hat{\phi}(x) - \frac{1}{2^n}\right|$$

8. Output $\hat{d}_1 + \hat{d}_2$.

Proof. We present the algorithm as Algorithm 2. The number of samples drawn from ρ is $N_1 + N_2 = \frac{2^n}{2^{\Theta_\epsilon(\sqrt{n})}}$. The run-time, in turn, is dominated[5] by computing the values of $\hat{\phi}$ in step (6), in which the construction of the lookup table takes $O(n \cdot 2^n)$ time and the time spent computing each $\hat{\phi}(x)$ can be upper bounded by the product of: (i) the number of pairs (y, z) that simultaneously satisfy $y \preceq z \preceq x$ and $||x|| - ||y|| = L$, which can be upper-bounded by $O(n^L \cdot 2^L)$ and (ii) the time it takes to look up a given z in the lookup table, which can be upper-bounded by $O(n)$. Overall, this gives us a run-time upper bound of $O(2^{n+O_\epsilon(\sqrt{n}\log n)})$.

Now, the only thing left to prove is correctness. First of all, it is not a priori clear that there exists a value of h_0 satisfying Equation 14 (in Algorithm 2). This is true for the following reason: imagine changing $h_{\text{candidate}}$ from n to 0 by decrementing it in steps of one. Then $\Pr_{x\in\{0,1\}^n}[||x|| \geq h_{\text{candidate}}]$ will increase from $\frac{1}{2^n}$ to 1 and by Fact 16 it will not increase by more than $\frac{2}{\sqrt{n}}$ at any given step. For sufficiently large n we have $\frac{2}{\sqrt{n}} < \frac{\epsilon}{4}$. Then it is impossible to skip over the interval between $\frac{\epsilon}{4}$ and $\frac{\epsilon}{2}$ in just one step of length at most $\frac{2}{\sqrt{n}}$, and therefore Equation 14 (in Algorithm 2) will be the case for some value of $h_{\text{candidate}}$.

We decompose the total variation distance between ρ and the uniform distribution into three terms:

$$\frac{1}{2}\cdot\sum_{x\in\{0,1\}^n}\left|\rho(x)-\frac{1}{2^n}\right| = \frac{1}{2}\cdot\sum_{\substack{x\in\{0,1\}^n \text{ s.t.}\\ ||x||\geq h_0}}\left|\rho(x)-\frac{1}{2^n}\right| + \frac{1}{2}\cdot\sum_{\substack{x\in\{0,1\}^n \text{ s.t.}\\ L\leq||x||<h_0}}\left|\rho(x)-\frac{1}{2^n}\right| + \frac{1}{2}\cdot\sum_{\substack{x\in\{0,1\}^n \text{ s.t.}\\ ||x||<L}}\left|\rho(x)-\frac{1}{2^n}\right| \tag{15}$$

We argue that the first term is well approximated by $\hat{d}_1$, the second term is well approximated by $\hat{d}_2$, and the third term is negligible. As the reader will see, out of these three terms, the middle term is the least trivial to prove guarantees for.

We will first handle the first term: From the triangle inequality, Hoeffding's bound and Equation 14 (in Algorithm 2) it follows immediately that with probability at least 7/8 it is the case that:

$$\left|\hat{d}_1 - \frac{1}{2}\cdot\sum_{x\in\{0,1\}^n \text{ s.t. } ||x||\geq h_0}\left|\rho(x)-\frac{1}{2^n}\right|\right| \leq \left|\hat{d}_1 - \frac{1}{2}\cdot\sum_{x\in\{0,1\}^n \text{ s.t. } ||x||\geq h_0}\rho(x)\right| + \frac{1}{2}\cdot\sum_{x\in\{0,1\}^n \text{ s.t. } ||x||\geq h_0}\frac{1}{2^n} \leq \frac{\epsilon}{8}+\frac{\epsilon}{4} = \frac{3\epsilon}{8} \tag{16}$$

Now, we use the two following facts: (i) Since $\sum_x \rho(x) = 1$ and ρ is monotone, for every x with $||x|| \leq L$ it should be the case that $\rho(x) \leq \frac{1}{2^{n-L}}$. (ii) The number of different values of x in $\{0,1\}^n$ for which $||x|| \leq L$ can be upper bounded by n^L. We get for sufficiently large n:

$$\frac{1}{2}\cdot\sum_{x\in\{0,1\}^n \text{ s.t. } ||x||<L}\left|\rho(x)-\frac{1}{2^n}\right| \leq \frac{1}{2}\cdot\sum_{x\in\{0,1\}^n \text{ s.t. } ||x||<L}\left(\frac{1}{2^n}+\rho(x)\right) \leq \frac{1}{2}\cdot n^L\cdot\left(\frac{1}{2^n}+\frac{1}{2^{n-L}}\right) = o(1) \leq \frac{\epsilon}{8} \tag{17}$$

[5] Step 1 requires only $2^n\text{poly}(n)$ time, which is less than what step (6) requires. By inspection, other steps require even less run-time. Incidentally, the task in step 1 can be done much faster by randomized sampling, but since this is not the run-time bottleneck, we use this direct approach for the sake of simplicity.

The rest of this section will be dedicated to proving the following claim:

▷ Claim 24. With probability at least 7/8 it is the case that:

$$\left|\hat{d}_2 - \frac{1}{2} \cdot \sum_{x \in \{0,1\}^n \text{ s.t. } L \leq ||x|| < h_0} \left|\rho(x) - \frac{1}{2^n}\right|\right| \leq \frac{\epsilon}{2} \tag{18}$$

Once this is proven, it follows by a union bound that with probability at least 3/4 both Equations 16 and 18 will be the case. This, together with Equation 17, when substituted into Equation 15 will imply that:

$$\left| \sum_{x \in \{0,1\}^n} \left|\rho(x) - \frac{1}{2^n}\right| - (\hat{d}_1 + \hat{d}_2)\right| \leq \epsilon$$

This will imply the correctness of our algorithm. ◀

4.1 Proof of Claim 24

For all x in $\{0,1\}^n$, satisfying $L \leq ||x|| < h_0$, we define the following quantity:

$$\phi(x) \stackrel{\text{def}}{=} \frac{1}{2^L} \cdot \max_{y \text{ s.t. } y \preceq x \text{ and } ||x||-||y||=L} \Pr_{z \sim \rho}[y \preceq z \preceq x] = \frac{1}{2^L} \cdot \max_{y \text{ s.t. } y \preceq x \text{ and } ||x||-||y||=L} \sum_{z \text{ s.t. } y \preceq z \preceq x} \rho(z) \tag{19}$$

Observe that since for every such x and y there are 2^L values of z satisfying $y \preceq z \preceq x$, and ρ is a monotone probability distribution, it has to be the case that $\phi(x) \leq \rho(x)$ for all x on which $\phi(x)$ is defined.

We will be claiming that $\phi(x)$ is (in terms of L_1 distance) a good approximation to $\rho(x)$, but first we will show that $\hat{\phi}(x)$ is a good approximation to $\phi(x)$:

▷ Claim 25. With probability at least 7/8, it is the case that:

$$\sum_{x \in \{0,1\}^n \text{ s.t. } L \leq ||x|| < h_0} \left|\hat{\phi}(x) - \phi(x)\right| \leq \frac{\epsilon}{4} \tag{20}$$

Proof. We claim that for any pair (x, y), such that ϕ is defined on x and $||x|| - ||y|| = L$, with probability at least $1 - \frac{1}{8 \cdot 2^n \cdot n^L}$ the following holds:

$$\frac{1}{2^L}\left|\Pr_{z \sim \rho}[y \preceq z \preceq x] - \frac{|\{z \in S : y \preceq z \preceq x\}|}{N_2}\right| \leq \frac{\epsilon}{8} \cdot \max\left(\frac{1}{2^n}, \frac{1}{2^L}\Pr_{z \sim \rho}[y \preceq z \preceq x]\right) \tag{21}$$

We use Chernoff's bound to prove this as follows. Denote by q the value $\Pr_{z \sim \rho}[y \preceq z \preceq x]$. If $q \geq \frac{2^L}{2^n}$ then by Chernoff's bound we have:

$$\Pr\left[\left||\{z \in S : y \preceq z \preceq x\}| - qN_2\right| \geq \frac{\epsilon}{8}qN_2\right] \leq 2\exp\left(-\frac{1}{3}\left(\frac{\epsilon}{8}\right)^2 qN_2\right) \leq 2\exp\left(-\frac{1}{3}\left(\frac{\epsilon}{8}\right)^2 \frac{2^L}{2^n} \cdot N_2\right) = \frac{1}{8 \cdot 2^n \cdot n^L}$$

Otherwise, if we have $q < \frac{2^L}{2^n}$, then by Chernoff's bound:

$$\Pr\left[\left||\{z \in S : y \preceq z \preceq x\}| - qN_2\right| \geq \frac{\epsilon}{8} \cdot \frac{2^L}{2^n} \cdot N_2\right] \leq 2\exp\left(-\frac{1}{3}\left(\frac{\epsilon}{8} \cdot \frac{2^L}{2^n} \cdot \frac{1}{q}\right)^2 qN_2\right) \leq$$
$$2\exp\left(-\frac{1}{3}\left(\frac{\epsilon}{8}\right)^2 \frac{2^L}{2^n} \cdot N_2\right) = \frac{1}{8 \cdot 2^n \cdot n^L}$$

Now, by taking a union bound, it follows that with probability 7/8 for all such pairs (x, y) Equation 21 will be the case. For all x on which ϕ is defined it then will be the case that:

$$\left|\hat{\phi}(x) - \phi(x)\right| \leq \frac{\epsilon}{8} \cdot \max\left(\frac{1}{2^n}, \phi(x)\right)$$

Summing this for all x in the domain of ϕ we get:

$$\sum_{x \in \{0,1\}^n \text{ s.t. } L \leq ||x|| < h_0} \left|\hat{\phi}(x) - \phi(x)\right| \leq \frac{\epsilon}{8} \cdot \left(2^n \cdot \frac{1}{2^n} + \sum_{x \in \{0,1\}^n \text{ s.t. } L \leq ||x|| < h_0} \phi(x)\right) \leq$$
$$\frac{\epsilon}{8} \cdot \left(1 + \sum_{x \in \{0,1\}^n \text{ s.t. } L \leq ||x|| < h_0} \rho(x)\right) \leq \frac{\epsilon}{4}$$

◁

Now, we apply Corollary 10 to ρ. This gives a positive monotone function f that satisfies the three conditions of Corollary 10. We separate all the values of x in $\{0,1\}^n$ for which $L \leq ||x|| < h_0$ into two kinds: **good** and **bad**. Recall that by Corollary 10 an element x of $\{0,1\}^n$ for which $L \leq ||x|| < h_0$ can have $\text{slack}_f(x) \neq 0$ only if $||x|| = k_i$ for some i between 1 and t. We say that x is **bad** if there is some k_i for which $0 \leq ||x|| - k_i \leq L$. Otherwise, x if **good**.

We can write:

$$\sum_{x \in \{0,1\}^n \text{ s.t. } L \leq ||x|| < h_0} |\phi(x) - f(x)| = \sum_{\text{good } x} |\phi(x) - f(x)| + \sum_{\text{bad } x} |\phi(x) - f(x)| \tag{22}$$

Now, we bound the two terms above separately. If x is good, then it is the case that for all z satisfying $||x|| - L \leq ||z|| \leq ||x||$ we have $\text{slack}_f(x) = 0$, and therefore $f(z) = \max_{z' \in \{0,1\}^n \text{ s. t. } z' \preceq z \text{ and } ||z|| - ||z'|| = 1} f(z')$. Using this relation recursively, we obtain that:

$$f(x) = \max_{z \in \{0,1\}^n \text{ s. t. } z \preceq x \text{ and } ||x|| - ||z|| = L} f(z)$$

Therefore, since f is monotone, we obtain that:

$$f(x) = \frac{1}{2^L} \cdot \max_{y \text{ s.t. } y \preceq x \text{ and } ||x|| - ||y|| = L} \sum_{z \text{ s.t. } y \preceq z \preceq x} f(z)$$

By Corollary 10, it is the case $\rho(x) \geq f(x)$. This, together with the equation above and Equation 19 implies:

$$\phi(x) \geq f(x)$$

But we also know that $\rho(x) \geq \phi(x)$. Therefore:

$$\sum_{\text{good } x} |\phi(x) - f(x)| \leq \sum_{\text{good } x} |\rho(x) - f(x)| \leq \frac{\epsilon}{4} \tag{23}$$

Where the last inequality follows from Corollary 10.

Now, we bound the contribution of bad points.

$$\sum_{\text{bad } x} |\phi(x) - f(x)| \leq \sum_{\text{bad } x} \max(\phi(x), f(x)) \leq \sum_{\text{bad } x} \rho(x) = \sum_{k \in [L, h_0] \text{ s.t. for some } k_i\text{: } |k - k_i| \leq L} \mu_\rho(k) \cdot \binom{n}{k}$$

Now, by Claim 23 we have $\mu_\rho(k) \leq \frac{1}{2^n} \cdot \frac{4}{\epsilon}$ and by Fact 16 we have that $\binom{n}{k} \leq \frac{2}{\sqrt{n}} \cdot 2^n$. Combining these two facts with the inequality above we get:

$$\sum_{\text{bad } x} |\phi(x) - f(x)| \leq \left(\frac{1}{2^n} \cdot \frac{4}{\epsilon}\right) \cdot \left(\frac{2}{\sqrt{n}} \cdot 2^n\right) \cdot (L \cdot t) = \frac{4}{\epsilon} \cdot \frac{2}{\sqrt{n}} \cdot L \cdot t$$

Substituting the value of L and the upper bound on t from Corollary 10 we get:

$$\sum_{\text{bad } x} |\phi(x) - f(x)| \leq \frac{4}{\epsilon} \cdot \frac{2}{\sqrt{n}} \cdot \frac{\epsilon^4 \sqrt{n}}{512} \cdot \frac{16}{\epsilon^2} = \frac{\epsilon}{4}$$

Combining this with Equations 22 and 23 we get:

$$\sum_{x \in \{0,1\}^n \text{ s.t. } L \leq ||x|| < h_0} |\phi(x) - f(x)| \leq \frac{\epsilon}{2} \tag{24}$$

Overall, we have:

$$\left| \sum_{x \in \{0,1\}^n \text{ s.t. } L \leq ||x|| < h_0} |\rho(x) - 1/2^n| - \sum_{x \in \{0,1\}^n \text{ s.t. } L \leq ||x|| < h_0} |\hat{\phi}(x) - 1/2^n| \right| \leq$$
$$\sum_{x \in \{0,1\}^n \text{ s.t. } L \leq ||x|| < h_0} |\hat{\phi}(x) - \rho(x)| \leq \sum_{x \in \{0,1\}^n \text{ s.t. } L \leq ||x|| < h_0} |\hat{\phi}(x) - \phi(x)| +$$
$$\sum_{x \in \{0,1\}^n \text{ s.t. } L \leq ||x|| < h_0} |\phi(x) - f(x)| + \sum_{x \in \{0,1\}^n \text{ s.t. } L \leq ||x|| < h_0} |f(x) - \rho(x)|$$

This three terms can be bound using respectively Equation 20, Corollary 10 and Equation 24. This gives us:

$$\left| 2 \cdot \hat{d}_2 - \sum_{x \in \{0,1\}^n \text{ s.t. } L \leq ||x|| < h_0} \left| \rho(x) - \frac{1}{2^n} \right| \right| =$$
$$\left| \sum_{x \in \{0,1\}^n \text{ s.t. } L \leq ||x|| < h_0} |\rho(x) - 1/2^n| - \sum_{x \in \{0,1\}^n \text{ s.t. } L \leq ||x|| < h_0} |\hat{\phi}(x) - 1/2^n| \right| \leq \epsilon$$

Therefore, with probability at least 7/8 Equation 18 holds, which proves Claim 24 and completes the proof of correctness.

5 Estimating the support size

In this section we prove our upper-bound on the sample complexity of estimating the support size of an unknown monotone probability distribution over the Boolean cube. Recall that a probability distribution ρ is **well-behaved** if for every x either $\rho(x) = 0$ or $\rho(x) \geq 1/2^n$. We restate the theorem:

▶ **Theorem 4.** *For every positive ϵ, the following is true: for all sufficiently large n, there exists an algorithm, which given $\frac{2^n}{2^{\Theta_\epsilon(\sqrt{n})}}$ samples from an unknown well-behaved monotone probability distribution ρ over $\{0,1\}^n$, can reliably[6] approximate the support size of ρ with an additive error of up to ϵ. The algorithm runs in time $O_\epsilon\left(\frac{2^n}{2^{\Theta_\epsilon(\sqrt{n})}}\right)$*

Proof. The algorithm we use is listed as Algorithm 3.

Algorithm 3 Algorithm for the estimation of support size (given sample access from the distribution).

1. Set
$$M_1 = \frac{2^n}{2^{\frac{\epsilon^2}{64}\sqrt{n}}}\left(\ln\frac{32}{\epsilon}+1\right)$$
2. Take M_1 samples from the probability distributions. Call the set of these samples S_1.
3. Set
$$M_2 = \frac{32\ln 2}{\epsilon^2}$$
4. Pick M_2 elements of $\{0,1\}^n$ uniformly at random. Call these samples S_2.
5. We say that a point y is **covered** if in S_1 there exists at least one z, so that $z \preceq y$. One can check if a point y is covered by going through all the M_1 elements in S_1. Using this checking procedure, compute the fraction $\hat{\eta}$ of the elements in S_2 that are covered.
6. Output $\hat{\eta}$.

Clearly, the sample complexity is:

$$O\left(\frac{2^n}{2^{\frac{\epsilon^2}{64}\sqrt{n}}}\left(\ln\frac{32}{\epsilon}+1\right)\right) = \frac{2^n}{2^{\Theta_\epsilon(\sqrt{n})}}$$

In turn, the run-time is:

$$O\left(\frac{2^n}{2^{\frac{\epsilon^2}{64}\sqrt{n}}}\left(\ln\frac{32}{\epsilon}+1\right)\cdot\frac{32\ln 2}{\epsilon^2}\right) = \frac{2^n}{2^{\Theta_\epsilon(\sqrt{n})}}$$

Now, all is left to prove is correctness.

Let η denote the fraction of elements in $\{0,1\}^n$ that are covered by our samples in S_1. Then, a random element of $\{0,1\}^n$ is covered with probability η. Therefore, by the Hoeffding bound it follows that:

$$\Pr_{S_2}\left[|\hat{\eta}-\eta| > \frac{\epsilon}{4}\right] \leq 2\exp\left(-2\left(\frac{\epsilon}{4}\right)^2 M_2\right) = \frac{1}{8} \tag{25}$$

The last equality follows by substituting the value of M_2.

[6] By **reliably** we henceforth mean that the probability of success is at least 2/3.

Since ρ is monotone, it has to be the case that every point that is covered is in the support of ρ. Hence, the support size of ρ is at least $\eta \cdot 2^n$.

Now, all we need to show is that $\eta \cdot 2^n$ is not likely to be much smaller than the support size of ρ. We call a point x in the support of ρ **good** if there are at least $2^{\frac{\epsilon^2}{64}\sqrt{n}}$ points y each of which satisfying: (i) y belongs to the support of ρ. (ii) $x \preceq y$. If a point in the support of ρ is not good, then it is **bad**. We will show that the bad points are few, while a lot of the good points are likely to be covered.

Let f_{support} be defined as follows:

$$f_{\text{support}} \stackrel{\text{def}}{=} \begin{cases} 1 \text{ if } \rho(x) \neq 0 \\ 0 \text{ otherwise} \end{cases}$$

In other words, f_{support} is the indicator function of the support of ρ. Since ρ is a monotone probability distribution, f_{support} is a monotone function. Therefore, applying Lemma 8, there exists a function $g = g_1 \vee ... \vee g_t$ that $\epsilon/4$-approximates f_{support}, where $t \leq 8/\epsilon$ and each g_i is a monotone DNF with terms of width exactly k_i. Additionally, $g(x) \leq f(x)$ for all x in $\{0,1\}^n$

▷ Claim 26. For all i, g_i contains at most $\frac{\epsilon^2}{32} \cdot 2^n$ bad points.

Proof. Recall that g_i is k_i-regular, and therefore every point x on which $g_i(x) = 1$ needs to have Hamming weight $||x|| \geq k_i$.

▷ Claim 27. If x satisfies $g_i(x) = 1$ and $||x|| \geq k_i + \frac{\epsilon^2}{64}\sqrt{n}$, then x has to be good

Proof. Since g_i is a DNF and $g_i(x) = 1$ then x satisfies at least one of the terms of g_i. If there are more than one, arbitrarily pick one of them. Let this term be

$$t(y) = \bigwedge_{j \in H_1} y_j$$

The width of this AND has to be k_i, therefore $|H| = k_i$. Since $||x|| \geq k_i + \frac{\epsilon^2}{64}\sqrt{n}$, there must be $\frac{\epsilon^2}{64}\sqrt{n}$ values of j for which $x_j = 1$ but j is not in H_1. Denote the set of these values of j as H_2.

Now, consider an element $y \in \{0,1\}^n$ satisfying the criteria:

- For all j in H_1, $y_j = 1$.
- For all j neither in H_1 nor in H_2, $y_j = 0$.

Clearly, $t(y) = 1$, which implies $g_i(y) = 1$, $g(y) = 1$, and $f_{\text{support}}(y) = 1$. Also, for all j, we have $y_j \leq x_j$, and therefore $y \preceq x$. Finally, for all j in H_2 the value of y_j can be set arbitrarily to zero or one, and therefore there are $2^{|H_2|}$ such points, which is at least $2^{\frac{\epsilon^2}{64}\sqrt{n}}$. Therefore, x is a good point. ◁

Thus, we can upper-bound the number of bad points x on which $g_i(x) = 1$ by:

$$\left|\left\{x \in \{0,1\}^n : g(x) = 1 \text{ and } k_i + \frac{\epsilon^2}{64}\sqrt{n} > \sum_j x_j \geq k_i\right\}\right| \leq$$

$$\left|\left\{x \in \{0,1\}^n : k_i + \frac{\epsilon^2}{64}\sqrt{n} > \sum_j x_j \geq k_i\right\}\right| = \sum_{j=k_i}^{k_i + \frac{\epsilon^2}{64}\sqrt{n} - 1} \binom{n}{j} \leq \frac{\epsilon^2}{64}\sqrt{n}\binom{n}{n/2}$$

By Fact 16, for sufficiently large n, it is the case that $\binom{n}{n/2} \leq 2 \cdot \frac{2^n}{\sqrt{n}}$. This implies that the expression above is upper-bounded by $\frac{\epsilon^2}{32} \cdot 2^n$, which completes the proof of this claim. ◁

Since $g = g_1 \vee ... \vee g_t(x)$, our claim implies the following:

$$\left|\left\{x : g(x) = 1 \text{ and } x \text{ is bad}\right\}\right| \leq \sum_{i=1}^{t} \left|\left\{x : g_i(x) = 1 \text{ and } x \text{ is bad}\right\}\right| \leq$$
$$t \cdot \frac{\epsilon^2}{32} \cdot 2^n \leq \frac{8}{\epsilon} \cdot \frac{\epsilon^2}{32} \cdot 2^n = \frac{\epsilon}{4} \cdot 2^n$$

In addition, there could be at most $\frac{\epsilon}{4} \cdot 2^n$ bad points among the points on which f_{support} and g disagree. Thus, in total, there are at most $\frac{\epsilon}{2} \cdot 2^n$ bad points.

Finally, we need to argue that it is likely that many of the good points get covered:

▷ Claim 28. Suppose there are G good points. Then, with probability at least $7/8$ it will be the case that at least $1 - \epsilon/4$ fraction of these good points are covered.

Proof. For every good point x there exist least $2^{\frac{\epsilon^2}{64}\sqrt{n}}$ values of y for which i) $x \preceq y$ and ii) y is in the support of ρ. Since $x \preceq y$, if y is ever picked from the distribution, then x will be covered. Since y is in the support of ρ, and ρ is well-behaved, we have $\rho(y) \geq \frac{1}{2^n}$. Together, these imply that the probability that a random sample from ρ covers x is at least $\frac{2^{\frac{\epsilon^2}{64}\sqrt{n}}}{2^n}$. Hence, the probability that any of the M_1 i.i.d. samples taken from ρ does not cover x is at most:

$$\left(1 - \frac{2^{\frac{\epsilon^2}{64}\sqrt{n}}}{2^n}\right)^{M_1} = \left(1 - \frac{2^{\frac{\epsilon^2}{64}\sqrt{n}}}{2^n}\right)^{\frac{2^n}{2^{\frac{\epsilon^2}{64}\sqrt{n}}}\left(\ln\frac{32}{\epsilon}+1\right)} \leq \frac{1}{e^{\ln\frac{32}{\epsilon}}} = \frac{\epsilon}{32}$$

Let C denote a random variable, whose value equals to the number of the good points (out of total G) covered after taking M_1 i.i.d. samples from ρ.

The value of C has to satisfy these two constraints: (i) It has to be between 0 and G (ii) By linearity of expectation, $E[C] \geq (1 - \frac{\epsilon}{32})G$. Thus, to finish the proof of the Lemma, it is sufficient to show the following claim:

▷ Claim 29. If, for some fixed G, a random variable C is supported on $[0, G]$ and $E[C] \geq (1 - \frac{\epsilon}{32})G$, then $\Pr[C \geq (1 - \epsilon/4)G] \geq 7/8$.

Proof. This is immediate from Markov's inequality for the random variable $G - C$. ◁

◁

Now, we put it all together. Suppose that the bad events we previously identified do not happen. In particular, we know that with probability at least $7/8$ we have:

$$\left|\hat{\eta} - \eta\right| \leq \frac{\epsilon}{4}$$

Additionally, we also know that with probability at least $7/8$ it is the case that:

$$\left|\frac{\left|\left\{x : \substack{f_{\text{support}}(x)=1 \text{ and} \\ x \text{ is good and covered}}\right\}\right|}{2^n} - \frac{\left|\left\{x : \substack{f_{\text{support}}(x)=1 \text{ and} \\ x \text{ is good}}\right\}\right|}{2^n}\right| \leq$$
$$\frac{\epsilon}{4} \cdot \frac{|\{x : f_{\text{support}}(x) = 1 \text{ and } x \text{ is good}\}|}{2^n}$$

By union bound, the probability that none of this bad events happens is at least 3/4, which we will henceforth assume. Using the inequalities above together with the fact that the fraction of bad points is at most $\epsilon/2$ we get:

$$\left|\hat{\eta} - \frac{|\{x : f_{\text{support}}(x) = 1\}|}{2^n}\right| \leq \left|\hat{\eta} - \eta\right| + \left|\eta - \frac{|\{x : f_{\text{support}}(x) = 1\}|}{2^n}\right| =$$
$$\left|\hat{\eta} - \eta\right| + \left|\frac{|\{x : f_{\text{support}}(x) = 1 \text{ and } x \text{ is covered}\}|}{2^n} - \frac{|\{x : f_{\text{support}}(x) = 1\}|}{2^n}\right| \leq \left|\hat{\eta} - \eta\right| +$$
$$\left|\frac{\left|\left\{x : \begin{smallmatrix} f_{\text{support}}(x)=1 \text{ and} \\ x \text{ is good and covered} \end{smallmatrix}\right\}\right|}{2^n} - \frac{\left|\left\{x : \begin{smallmatrix} f_{\text{support}}(x)=1 \text{ and} \\ x \text{ is good} \end{smallmatrix}\right\}\right|}{2^n}\right| +$$
$$\frac{|\{x : f_{\text{support}}(x) = 1 \text{ and } x \text{ is bad}\}|}{2^n} \leq \frac{\epsilon}{4} + \frac{\epsilon}{4} \cdot \frac{|\{x : f_{\text{support}}(x) = 1 \text{ and } x \text{ is good}\}|}{2^n} + \frac{\epsilon}{2} \leq$$
$$\frac{\epsilon}{4} + \frac{\epsilon}{4} + \frac{\epsilon}{2} = \epsilon$$

This completes the proof of correctness. ◀

6 A lower bound on tolerant testing of uniformity

In this section we prove a sample complexity lower bound on the problem of tolerantly testing the uniformity of an unknown monotone probability distribution over $\{0,1\}^n$: the task of distinguishing a distribution that is $o(1)$-close to uniform from a distribution that is sufficiently far from uniform. Recall the theorem:

▶ **Theorem 6.** *For infinitely many positive integers n, there exist two probability distributions Δ_{Close} and Δ_{Far} over monotone distributions over $\{0,1\}^n$, satisfying:*

1. *Every distribution in Δ_{Far} is $1/2$-far from the uniform distribution.*
2. *Any algorithm that takes only $o\left(2^{\frac{n^{0.5-0.01}}{2}}\right)$ samples from a probability distribution, fails to reliably distinguish between Δ_{Close} and Δ_{Far}.*
3. *Every distribution in Δ_{Close} is $o(1)$-close to the uniform distribution.*

Proof. A basic building block of our construction is the following:

▶ **Definition 30.** *For a member of the Boolean cube x, the **subcube distribution** S_x is the probability distribution that picks y uniformly, subject to $y \succeq x$.*

All our distributions will be mixtures of such subcube distributions. For all the mixtures we will use, each subcube in the mixture is given the same weight. This method involving subcube distributions was used in [29] to prove property testing lower bounds for monotone probability distributions.

We construct Δ_{Close} to have only one member, which is equal to the uniform mixture of S_x for all $\binom{n}{n^{0.5-0.01}}$ values of x with Hamming weight $n^{0.5-0.01}$.

We define a random member of Δ_{Far} to be the uniform mixture of $\frac{1}{2}2^{n^{0.5-0.01}}$ subcube distributions S_{x_j}, where each of the x_j is picked randomly among all the members of the Boolean cube with Hamming weight $n^{0.5-0.01}$.

We show that any member of Δ_{Far} is sufficiently far from uniform by upper-bounding the size of its support (i.e. the number of elements that have non-zero probability). Each of the subcube distributions has a support size of $2^{n-n^{0.5-0.01}}$. The support size of a mixture of distributions is at most the sum of the supports sizes of the respective distributions.

Therefore, the support size of a member of Δ_{Far} is at most:

$$2^{n-n^{0.5-0.01}} \cdot \frac{1}{2} 2^{n^{0.5-0.01}} = \frac{1}{2} 2^n$$

This is sufficient to conclude that any member of Δ_{Far} is 1/2-far from uniform.

A random member D_1 of Δ_{Far} and the sole member D_2 of Δ_{Close} cannot be reliably distinguished using only $o\left(2^{\frac{n^{0.5-0.01}}{2}}\right)$ samples. This follows by the argument used in [29]: Because of the number of samples, with probability at least 0.99, the samples drawn from a random distribution from D_1 will all be from different subcube distributions. Also with probability at least 0.99, this will also be true for the sole distribution of D_2. If both of these things happen (which is the case with probability at least 0.98), the samples will be statistically indistinguishable. Thus, no tester can distinguish between D_1 and D_2 with an advantage greater than 0.02.

Finally, we need to prove that D_2 is $o(1)$-close to the uniform distribution. Here, the proof goes as follows. Both D_2 and the uniform distribution are symmetric with respect to a change of indices. This implies that the distance between these probability distributions equals to the distance between random variables R_2 and R_1, where R_1 is distributed as the Hamming weight of a random sample from D_2, whereas R_2 is distributed as the Hamming weight of uniformly random element of the Boolean cube. It is not hard to see that R_1 and R_2 are distributed according to binomial distributions with slightly different parameters. Now, the problem is equivalent to proving that the two following probability distributions are $o(1)$-close in total variation distance:

- A sum of n i.i.d. uniform random variables from $\{0,1\}$.
- A sum of $n - n^{0.5-0.01}$ i.i.d. uniform random variables from $\{0,1\}$.

It is convenient to first bound the variation distance between 1) the sum of k i.i.d. uniform random variables from $\{0,1\}$ and 2) $k+1$ i.i.d. uniform random variables from $\{0,1\}$, where k. We write the total variation distance as:

$$\frac{1}{2^k} + \sum_{i=1}^{k-1} \left| \frac{1}{2^k}\binom{k}{i} - \frac{1}{2^{k-1}}\binom{k-1}{i} \right| = \frac{1}{2^k}\left(1 + \sum_{i=1}^{k-1} \left| \binom{k-1}{i} - \binom{k-1}{i-1} \right| = \right) =$$

$$\frac{1}{2^k}\left(1 + \sum_{i=1}^{(k-1)/2} \left(\binom{k-1}{i} - \binom{k-1}{i-1} \right) + \sum_{i=(k-1)/2}^{k-1} \left(\binom{k-1}{i-1} - \binom{k-1}{i} \right)\right) =$$

$$\frac{1}{2^k}\left(1 + \binom{k-1}{(k-1)/2} - 1 + \binom{k-1}{(k-1)/2} - 1\right) = O\left(\frac{1}{\sqrt{k}}\right)$$

We telescoped the sums, and used the inequality that for all k, we have that $\binom{k}{k/2} \leq O\left(\frac{2^k}{\sqrt{k}}\right)$. For simplicity, we assumed above that $k-1$ is even, the odd case can be handled analogously. Thus, we have an upper bound of $O(1/\sqrt{k})$ on the total variation distance.

Using this, together with the triangle inequality for total variation distance, we bound the variation distance between 1) the sum of n i.i.d. uniform random variables from $\{0,1\}$ and 2) the sum of $n - n^{0.5-0.01}$ i.i.d. uniform random variables from $\{0,1\}$ by

$$O\left(\frac{n^{0.5-0.01}}{n^{0.5}}\right) = o(1).$$

This finishes the proof. ◀

References

1 Jayadev Acharya, Constantinos Daskalakis, and Gautam Kamath. Optimal Testing for Properties of Distributions. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 3591–3599, 2015. URL: http://papers.nips.cc/paper/5839-optimal-testing-for-properties-of-distributions.

2 Michal Adamaszek, Artur Czumaj, and Christian Sohler. Testing Monotone Continuous Distributions on High-dimensional Real Cubes. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, pages 56–65, 2010. doi:10.1137/1.9781611973075.6.

3 Maryam Aliakbarpour, Themis Gouleakis, John Peebles, Ronitt Rubinfeld, and Anak Yodpinyanee. Towards Testing Monotonicity of Distributions Over General Posets. In *Conference on Learning Theory, COLT 2019, 25-28 June 2019, Phoenix, AZ, USA*, pages 34–82, 2019. URL: http://proceedings.mlr.press/v99/aliakbarpour19a.html.

4 Tugkan Batu, Ravi Kumar, and Ronitt Rubinfeld. Sublinear algorithms for testing monotone and unimodal distributions. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 381–390. ACM, 2004.

5 Arnab Bhattacharyya, Eldar Fischer, Ronitt Rubinfeld, and Paul Valiant. Testing monotonicity of distributions over general partial orders. In *ICS*, pages 239–252, 2011.

6 Lucien Birgé et al. Estimating a density under order restrictions: Nonasymptotic minimax risk. *The Annals of Statistics*, 15(3):995–1012, 1987.

7 Eric Blais, Johan Håstad, Rocco A Servedio, and Li-Yang Tan. On DNF approximators for monotone boolean functions. In *International Colloquium on Automata, Languages, and Programming*, pages 235–246. Springer, 2014.

8 Clément L Canonne, Ilias Diakonikolas, Themis Gouleakis, and Ronitt Rubinfeld. Testing shape restrictions of discrete distributions. *Theory of Computing Systems*, 62(1):4–62, 2018.

9 Clément L. Canonne, Ilias Diakonikolas, Daniel M. Kane, and Alistair Stewart. Testing Bayesian Networks. In *Proceedings of the 30th Conference on Learning Theory, COLT 2017, Amsterdam, The Netherlands, 7-10 July 2017*, pages 370–448, 2017. URL: http://proceedings.mlr.press/v65/canonne17a.html.

10 Siu-On Chan, Ilias Diakonikolas, Rocco A Servedio, and Xiaorui Sun. Learning mixtures of structured distributions over discrete domains. In *Proceedings of the twenty-fourth annual ACM-SIAM symposium on Discrete algorithms*, pages 1380–1394. Society for Industrial and Applied Mathematics, 2013.

11 Siu-on Chan, Ilias Diakonikolas, Paul Valiant, and Gregory Valiant. Optimal Algorithms for Testing Closeness of Discrete Distributions. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 1193–1203, 2014. doi:10.1137/1.9781611973402.88.

12 Constantinos Daskalakis, Ilias Diakonikolas, and Rocco A. Servedio. Learning k-Modal Distributions via Testing. *Theory of Computing*, 10:535–570, 2014. doi:10.4086/toc.2014.v010a020.

13 Constantinos Daskalakis, Ilias Diakonikolas, and Rocco A. Servedio. Learning Poisson Binomial Distributions. *Algorithmica*, 72(1):316–357, 2015. doi:10.1007/s00453-015-9971-3.

14 Constantinos Daskalakis, Ilias Diakonikolas, Rocco A. Servedio, Gregory Valiant, and Paul Valiant. Testing k-Modal Distributions: Optimal Algorithms via Reductions. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6-8, 2013*, pages 1833–1852, 2013. doi:10.1137/1.9781611973105.131.

15 Constantinos Daskalakis and Gautam Kamath. Faster and Sample Near-Optimal Algorithms for Proper Learning Mixtures of Gaussians. In *Proceedings of The 27th Conference on Learning Theory, COLT 2014, Barcelona, Spain, June 13-15, 2014*, pages 1183–1213, 2014.

16 Constantinos Daskalakis, Gautam Kamath, and Christos Tzamos. On the Structure, Covering, and Learning of Poisson Multinomial Distributions. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 1203–1217, 2015. doi:10.1109/FOCS.2015.77.

17 Ilias Diakonikolas, Themis Gouleakis, John Peebles, and Eric Price. Collision-Based Testers are Optimal for Uniformity and Closeness. *Chicago J. Theor. Comput. Sci.*, 2019, 2019. URL: http://cjtcs.cs.uchicago.edu/articles/2019/1/contents.html.

18 Ilias Diakonikolas, Daniel M Kane, and Vladimir Nikishkin. Testing identity of structured distributions. In *Proceedings of the twenty-sixth annual ACM-SIAM symposium on Discrete algorithms*, pages 1841–1854. Society for Industrial and Applied Mathematics, 2015.

19 Ilias Diakonikolas, Daniel M Kane, and Alistair Stewart. Efficient robust proper learning of log-concave distributions. *arXiv preprint*, 2016. arXiv:1606.03077.

20 Ilias Diakonikolas, Daniel M Kane, and Alistair Stewart. Optimal learning via the fourier transform for sums of independent integer random variables. In *Conference on Learning Theory*, pages 831–849, 2016.

21 Ilias Diakonikolas, Daniel M Kane, and Alistair Stewart. Properly learning poisson binomial distributions in almost polynomial time. In *Conference on Learning Theory*, pages 850–878, 2016.

22 Ilias Diakonikolas, Jerry Li, and Ludwig Schmidt. Fast and sample near-optimal algorithms for learning multidimensional histograms. *arXiv preprint*, 2018. arXiv:1802.08513.

23 Rong Ge, Qingqing Huang, and Sham M. Kakade. Learning Mixtures of Gaussians in High Dimensions. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 761–770, 2015. doi:10.1145/2746539.2746616.

24 Piotr Indyk, Reut Levi, and Ronitt Rubinfeld. Approximating and testing k-histogram distributions in sub-linear time. In *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGAI symposium on Principles of Database Systems*, pages 15–22. ACM, 2012.

25 Adam Tauman Kalai, Ankur Moitra, and Gregory Valiant. Efficiently learning mixtures of two Gaussians. In *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 553–562, 2010. doi:10.1145/1806689.1806765.

26 Liam Paninski. A coincidence-based test for uniformity given very sparsely sampled discrete data. *IEEE Transactions on Information Theory*, 54(10):4750–4755, 2008.

27 Sofya Raskhodnikova, Dana Ron, Amir Shpilka, and Adam Smith. Strong lower bounds for approximating distribution support size and the distinct elements problem. *SIAM Journal on Computing*, 39(3):813–842, 2009.

28 Oded Regev and Aravindan Vijayaraghavan. On Learning Mixtures of Well-Separated Gaussians. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 85–96, 2017. doi:10.1109/FOCS.2017.17.

29 Ronitt Rubinfeld and Rocco A Servedio. Testing monotone high-dimensional distributions. *Random Structures & Algorithms*, 34(1):24–44, 2009.

30 Gregory Valiant and Paul Valiant. Estimating the unseen: an n/log (n)-sample estimator for entropy and support size, shown optimal via new CLTs. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 685–694. ACM, 2011.

31 Gregory Valiant and Paul Valiant. The power of linear estimators. In *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, pages 403–412. IEEE, 2011.

32 Paul Valiant. Testing symmetric properties of distributions. *SIAM Journal on Computing*, 40(6):1927–1968, 2011.

33 Yihong Wu and Pengkun Yang. Minimax rates of entropy estimation on large alphabets via best polynomial approximation. *IEEE Transactions on Information Theory*, 62(6):3702–3720, 2016.

34 Yihong Wu, Pengkun Yang, et al. Chebyshev polynomials, moment matching, and optimal estimation of the unseen. *The Annals of Statistics*, 47(2):857–883, 2019.

7 Appendix A

7.1 Verifying the conditions on L_h

Recall that we defined A and L_h as follows:

- $A := \frac{1}{2n} \cdot e^{\frac{1}{2000} \cdot n^{1/5}}$
- For all $h \geq n/2$, we set $L_h := \max\left(\log\left(2nA \cdot \frac{\binom{n}{h}}{2^n}\right), 0\right)$
- For all h, satisfying $n/2 > h \geq 9\sqrt{n}$, we set: $L_h := L_{n/2} = \log\left(2nA \cdot \frac{\binom{n}{n/2}}{2^n}\right)$.

Here we prove that these values of A and L_h satisfy the following four conditions:

a) As a function of h, L_h is non-increasing.

b) For all h, we have that $L_h \leq 9\sqrt{n}$.

c)

$$\frac{1}{2^n} \cdot \sum_{h=9\sqrt{n}}^{n} \binom{n}{h} \cdot \frac{A}{2^{L_h}} \leq \frac{1}{2}$$

d)

$$\sum_{h=9\sqrt{n}}^{n} L_h \cdot \left(\begin{cases} \frac{400}{n^{2.5}} & \text{if } h \leq n/2 - \sqrt{n\ln(n)} \\ \frac{40000}{n} & \text{if } n/2 - \sqrt{n\ln(n)} < h < n/2 + \sqrt{n} \\ 40000 \cdot \left(\frac{h-n/2}{n}\right)^2 & \text{if } h \geq n/2 + \sqrt{n} \end{cases} \right) \leq \frac{\epsilon^2}{20000}$$

We will need the following standard fact can be proven, for example, by comparing $\sum_{i=0}^{N} i^k$ and $\int_{i=0}^{N} i^k \, di$:

▶ **Fact 31.** *For any positive constant k and for sufficiently large n, it is the case that:* $\sum_{i=0}^{n} i^k = (1 + o(1)) \, \frac{n^{k+1}}{k+1}$.

The truth of conditions (a) and (b) follows immediately by inspection. In fact a statement stronger than (b) is the case: for sufficiently large n we have $L_h \leq \log(n \cdot A) \leq 2 \cdot n^{1/5}$. Regarding condition (c), we have:

$$\frac{1}{2^n} \cdot \sum_{h=9\sqrt{n}}^{n} \binom{n}{h} \cdot \frac{A}{2^{L_h}} =$$

$$\frac{A}{2^n} \left(\sum_{h=9\sqrt{n}}^{n/2} \binom{n}{h} \frac{1}{2nA} \cdot \frac{2^n}{\binom{n}{n/2}} + \sum_{h=n/2}^{n} \binom{n}{h} \cdot \min\left(\frac{1}{2nA} \cdot \frac{2^n}{\binom{n}{h}}, 1\right) \right) \leq \sum_{h=9\sqrt{n}}^{n} \frac{1}{2n} \leq \frac{1}{2}$$

Finally, recall that for all h, we have $L_h \leq 2 \cdot n^{1/5}$. For sufficiently large n, we have:

$$\sum_{h=9\sqrt{n}}^{n} L_h \cdot \left(\begin{cases} \frac{400}{n^{2.5}} & \text{if } h \leq n/2 - \sqrt{n\ln(n)} \\ \frac{40000}{n} & \text{if } n/2 - \sqrt{n\ln(n)} < h < n/2 + \sqrt{n} \\ 40000 \cdot \left(\frac{h-n/2}{n}\right)^2 & \text{if } h \geq n/2 + \sqrt{n} \end{cases} \right) \leq$$

$$\sum_{h=9\sqrt{n}}^{n/2-\sqrt{n\ln n}} 2 \cdot n^{1/5} \cdot \frac{400}{n^{2.5}} + \sum_{h=n/2-\sqrt{n\ln n}}^{n+\sqrt{n}} 2 \cdot n^{1/5} \cdot \frac{40000}{n} + \sum_{h=n/2+\sqrt{n}}^{n} 40000 \cdot \left(\frac{h-n/2}{n}\right)^2 \cdot L_h \leq$$

$$\frac{\epsilon^2}{40000} + \sum_{h=n/2+\sqrt{n}}^{n} 40000 \cdot \left(\frac{h-n/2}{n}\right)^2 \cdot L_h \quad (26)$$

We now bound the last term using Hoeffding's inequality to bound the value of $\binom{n}{h}$, making a change of variables with $i := \frac{n-h}{2}$ and then using Fact 31 to bound the resulting summation. Precisely, we have the following chain of inequalities (some of which are only true for sufficiently large n):

$$\sum_{h=n/2+\sqrt{n}}^{n} 40000 \cdot \left(\frac{h-n/2}{n}\right)^2 \cdot L_h \leq$$

$$\sum_{h=n/2}^{n} 40000 \cdot \left(\frac{h-n/2}{n}\right)^2 \cdot \max\left(\log\left(2nA \cdot \frac{\binom{n}{h}}{2^n}\right), 0\right) \leq$$

$$\sum_{h=n/2}^{n} 40000 \cdot \left(\frac{h-n/2}{n}\right)^2 \cdot \max\left(\log\left(2nA \cdot \exp\left(-2\frac{(h-n/2)^2}{n}\right)\right), 0\right) =$$

$$\sum_{i=0}^{\sqrt{\frac{n}{2}\ln(2nA)}} \frac{40000}{\ln 2} \cdot \left(\frac{i}{n}\right)^2 \left(\ln(2nA) - 2 \cdot \frac{i^2}{n}\right) =$$

$$\frac{40000}{\ln 2} \cdot \left((1+o(1)) \frac{\left(\sqrt{\frac{n}{2}\ln(2nA)}\right)^3}{3n^2} \ln(2nA) - (1+o(1))\, 2 \cdot \frac{\left(\sqrt{\frac{n}{2}\ln(2nA)}\right)^5}{5n^3}\right)$$

Finally, simplifying and substituting the value of A we get:

$$\sum_{h=n/2+\sqrt{n}}^{n} 40000 \cdot \left(\frac{h-n/2}{n}\right)^2 \cdot L_h \leq (1+o(1)) \cdot \frac{40000}{\ln 2} \cdot \frac{\sqrt{2}}{30\sqrt{n}} \left(\ln(2nA)\right)^{5/2} \leq \frac{\epsilon^2}{40000}$$

Condition (d) is verified by combining Equation 26 with the equation above.

7.2 Proof of Claim 21

Here we prove that for all sufficiently large n, for all h, satisfying $0 \leq h \leq n$, it is the case that:

$$\frac{\binom{n}{h}}{\sum_{j\geq h}^{n}\binom{n}{j}} \leq \left(\begin{cases} \frac{2}{n^2} & \text{if } h \leq n/2 - \sqrt{n\ln(n)} \\ \frac{200}{\sqrt{n}} & \text{if } n/2 - \sqrt{n\ln(n)} < h < n/2 + \sqrt{n} \\ 200 \cdot \frac{h-n/2}{n} & \text{if } h \geq n/2 + \sqrt{n} \end{cases}\right)$$

We first handle the case when $h \geq n/2 + \sqrt{n}$. If, furthermore, $h > 11n/20$, then it is sufficient to prove that $\frac{\binom{n}{h}}{\sum_{j\geq h}^{n}\binom{n}{j}} \leq 10$, which is trivially true. Thus, we now assume that $h \leq 11n/20$

It is the case that:

$$\frac{\binom{n}{k}}{\binom{n}{k-1}} = \frac{1 - \frac{k-n/2-1}{n/2}}{1 + \frac{k-n/2}{n/2}} \tag{27}$$

Therefore, we can write:

$$\frac{\sum_{j\geq h}^{n}\binom{n}{j}}{\binom{n}{h}} = \sum_{j=h}^{n} \prod_{k=h+1}^{j} \frac{1 - \frac{k-n/2-1}{n/2}}{1 + \frac{k-n/2}{n/2}}$$

Since $n/2+\sqrt{n} \leq h \leq 11n/20$, for sufficiently large n we have that $h+\frac{1}{4}\cdot\frac{n}{h-n/2} \leq n$. Using this, we can truncate the sum above, and then lower-bound the result by the product of the smallest summand with the total number of summands, getting:

$$\frac{\sum_{j\geq h}^{n}\binom{n}{j}}{\binom{n}{h}} \geq \sum_{j=h}^{h+\frac{1}{4}\cdot\frac{n}{h-n/2}} \prod_{k=h+1}^{j} \frac{1-\frac{k-n/2-1}{n/2}}{1+\frac{k-n/2}{n/2}} \geq \frac{1}{4}\cdot\frac{n}{h-n/2}\cdot \prod_{k=h+1}^{h+\frac{1}{4}\cdot\frac{n}{h-n/2}} \frac{1-\frac{k-n/2-1}{n/2}}{1+\frac{k-n/2}{n/2}}$$

Now, we analogously lower-bound the product by lower-bounding each of the factors, and then use the fact that since $h \geq n/2+\sqrt{n}$, it is the case that $\frac{n}{h-n/2} \leq h-n/2$. We get:

$$\frac{\sum_{j\geq h}^{n}\binom{n}{j}}{\binom{n}{h}} \geq \frac{1}{4}\cdot\frac{n}{h-n/2}\cdot\left(\frac{1-\frac{h+\frac{1}{4}\cdot\frac{n}{h-n/2}-n/2-1}{n/2}}{1+\frac{h+\frac{1}{4}\cdot\frac{n}{h-n/2}-n/2}{n/2}}\right)^{\frac{1}{4}\cdot\frac{n}{h-n/2}} \geq$$

$$\frac{1}{4}\cdot\frac{n}{h-n/2}\cdot\left(\frac{1-1.25\frac{h-n/2}{n/2}}{1+1.25\frac{h-n/2}{n/2}}\right)^{\frac{1}{4}\cdot\frac{n}{h-n/2}}$$

Finally, we use the fact that for all w between zero and one we have that $\frac{1}{1+w} = 1-w+w^2-... \geq 1-w$. We get:

$$\frac{\sum_{j\geq h}^{n}\binom{n}{j}}{\binom{n}{h}} \geq \frac{1}{4}\cdot\frac{n}{h-n/2}\cdot\left(1-1.25\frac{h-n/2}{n/2}\right)^{\frac{1}{2}\cdot\frac{2n}{h-n/2}}$$

Now, recall that for any value w between zero and one, we have that $\ln(1-w) = -\sum_{i=1}^{\infty}\frac{w^i}{i} \geq -\sum_{i=1}^{\infty} w^i = -\frac{w}{1-w}$. Using this, and recalling that $h \leq 11n/20$, we get that:

$$\ln\left(1-1.25\frac{h-n/2}{n/2}\right) \geq -\frac{1.25\frac{h-n/2}{n/2}}{1-1.25\frac{h-n/2}{n/2}} \geq -\frac{1.25\frac{h-n/2}{n/2}}{1-1.25\frac{11n/20-n/2}{n/2}} = -\frac{20}{7}\frac{h-n/2}{n}$$

Combining the two previous equations together we get:

$$\frac{\sum_{j\geq h}^{n}\binom{n}{j}}{\binom{n}{h}} \geq \frac{1}{4}\cdot\frac{n}{h-n/2}\cdot\exp\left(-\frac{20}{7}\frac{h-n/2}{n}\cdot\frac{1}{2}\cdot\frac{n}{h-n/2}\right) \geq \frac{1}{200}\cdot\frac{n}{h-n/2}$$

This completes the proof in the case $h \geq n/2+\sqrt{n}$.

Given our bound in the range $h \geq n/2+\sqrt{n}$, to show the desired bound in the range $n/2-\sqrt{n\ln(n)} < h < n/2+\sqrt{n}$ it is sufficient to show that $\frac{\binom{n}{h}}{\sum_{j\geq h}^{n}\binom{n}{j}}$ is non-decreasing, as a function of h. If $h < n/2$, this follows immediately, because, as a function of h, the numerator is non-decreasing, whereas the denominator is decreasing. If $h \geq n/2$, then using Equation 27, we get:

$$\frac{\sum_{j\geq h+1}^{n}\binom{n}{j}}{\binom{n}{h+1}} = \frac{1+\frac{h+1-n/2}{n/2}}{1-\frac{h-n/2}{n/2}}\cdot\frac{1}{\binom{n}{h}}\cdot\sum_{j\geq h+1}^{n}\frac{1-\frac{j-n/2-1}{n/2}}{1+\frac{j-n/2}{n/2}}\binom{n}{j-1} \leq$$

$$\frac{1}{\binom{n}{h}}\cdot\sum_{j\geq h+1}^{n}\binom{n}{j-1} \leq \frac{1}{\binom{n}{h}}\cdot\sum_{j\geq h+1}^{n+1}\binom{n}{j-1} = \frac{\sum_{j\geq h}^{n}\binom{n}{j}}{\binom{n}{h}}$$

Which implies that $\frac{\binom{n}{h+1}}{\sum_{j\geq h+1}^{n}\binom{n}{j}} \geq \frac{\binom{n}{h}}{\sum_{j\geq h}^{n}\binom{n}{j}}$

Finally, for the range $h \leq n/2-\sqrt{n\ln(n)}$ we can use Hoeffding's bound:

$$\frac{\binom{n}{h}}{\sum_{j\geq h}^{n}\binom{n}{j}} \leq \frac{\binom{n}{h}}{\frac{1}{2}\cdot 2^n} \leq 2\cdot\Pr_{x\sim\{0,1\}^n}\left[x \leq n/2-\sqrt{n\ln(n)}\right] \leq 2\cdot\exp\left(-2\ln n\right) = \frac{2}{n^2}$$

7.3 Proof fo Claim 22

Recall that:

$$N \stackrel{\text{def}}{=} \frac{2^n}{A} \cdot \frac{192}{\epsilon^2} \cdot (n + 9\sqrt{n} + 4)$$

Our algorithm for learning a monotone probability distribution drew N samples from the probability distribution ρ and the resulting multiset of samples was denoted as S. For all x in $\{0,1\}^n$, if $||x|| < 9\sqrt{n}$, we set $\hat{\phi}(x) = 0$, otherwise we set:

$$\hat{\phi}(x) := \frac{1}{2^{\lfloor L_{||x||} \rfloor}} \cdot \frac{\max_{y \text{ s.t. } y \preceq x \text{ and } ||y||-||x|| = \lfloor L_{||x||} \rfloor} \left| \left\{ z \in S : y \preceq z \preceq x \right\} \right|}{N}$$

Where L_h is a specific value associated to each value of h. We also defined for all x with $x \geq 9\sqrt{n}$ the value:

$$\phi(x) \stackrel{\text{def}}{=} \frac{1}{2^{\lfloor L_{||x||} \rfloor}} \cdot \max_{y \text{ s.t. } y \preceq x \text{ and } ||x||-||y|| = \lfloor L_{||x||} \rfloor} \Pr_{z \sim \rho}[y \preceq z \preceq x] =$$
$$\frac{1}{2^{\lfloor L_{||x||} \rfloor}} \cdot \max_{y \text{ s.t. } y \preceq x \text{ and } ||x||-||y|| = \lfloor L_{||x||} \rfloor} \sum_{z \text{ s.t. } y \preceq z \preceq x} \rho(z)$$

Here we prove that if it is the case that $\frac{1}{2^n} \cdot \sum_{h=9\sqrt{n}}^{n} \binom{n}{h} \cdot \frac{A}{2^{L_h}} \leq \frac{1}{2}$, then, with probability at least $7/8$, it is the case that:

$$\sum_{x \in \{0,1\}^n \text{ s.t. } 9\sqrt{n} \leq ||x||} \left| \hat{\phi}(x) - \phi(x) \right| \leq \frac{\epsilon}{4}$$

We claim that for any pair (x, y), such that ϕ is defined on x and $||x|| - ||y|| = \lfloor L_{||x||} \rfloor$, with probability at least $1 - \frac{1}{8 \cdot 2^n \cdot n^{9\sqrt{n}}}$ the following holds:

$$\left| \Pr_{z \sim \rho}[y \preceq z \preceq x] - \frac{|\{z \in S : y \preceq z \preceq x\}|}{N} \right| \leq \frac{\epsilon}{8} \cdot \max\left(\frac{A}{2^n}, \Pr_{z \sim \rho}[y \preceq z \preceq x] \right) \tag{28}$$

We use Chernoff's bound to prove this as follows. Denote by q the value $\Pr_{z \sim \rho}[y \preceq z \preceq x]$. If $q \geq \frac{A}{2^n}$ then by Chernoff's bound we have:

$$\Pr\left[\left| |\{z \in S : y \preceq z \preceq x\}| - qN \right| \geq \frac{\epsilon}{8} qN \right] \leq$$
$$2 \exp\left(-\frac{1}{3} \left(\frac{\epsilon}{8} \right)^2 qN \right) \leq 2 \exp\left(-\frac{1}{3} \left(\frac{\epsilon}{8} \right)^2 \frac{A}{2^n} \cdot N \right) = \frac{1}{8 \cdot 2^n \cdot n^{9\sqrt{n}}}$$

Otherwise, if we have $q < \frac{A}{2^n}$, then by Chernoff's bound:

$$\Pr\left[\left| |\{z \in S : y \preceq z \preceq x\}| - qN \right| \geq \frac{\epsilon}{8} \cdot \frac{A}{2^n} \cdot N \right] \leq 2 \exp\left(-\frac{1}{3} \left(\frac{\epsilon}{8} \cdot \frac{A}{2^n} \cdot \frac{1}{q} \right)^2 qN \right) \leq$$
$$2 \exp\left(-\frac{1}{3} \left(\frac{\epsilon}{8} \right)^2 \frac{A}{2^n} \cdot N \right) = \frac{1}{8 \cdot 2^n \cdot n^{9\sqrt{n}}}$$

Now, by taking a union bound, it follows that with probability $7/8$ for all such pairs (x, y) Equation 28 will be the case. Recalling the definition of ϕ, for all x on which ϕ is defined it then will be the case that:

$$\left| \hat{\phi}(x) - \phi(x) \right| \leq \frac{\epsilon}{8} \cdot \max\left(\frac{1}{2^{\lfloor L_{||x||} \rfloor}} \frac{A}{2^n}, \phi(x) \right)$$

Now, we sum this for all x in the domain of ϕ and use the fact that $\lfloor L_h \rfloor \geq L_h + 1$, and then use that $\frac{1}{2^n} \cdot \sum_{h=9\sqrt{n}}^{n} \binom{n}{h} \cdot \frac{A}{2^{L_h}} \leq \frac{1}{2}$. We get:

$$\sum_{x \in \{0,1\}^n \text{ s.t. } 9\sqrt{n} \leq ||x||} \left| \hat{\phi}(x) - \phi(x) \right| \leq$$

$$\frac{\epsilon}{8} \cdot \left(\frac{1}{2^n} \cdot \sum_{x \in \{0,1\}^n \text{ s.t. } 9\sqrt{n} \leq ||x||} \frac{A}{2^{\lfloor L_{||x||} \rfloor}} + \sum_{x \in \{0,1\}^n \text{ s.t. } 9\sqrt{n} \leq ||x||} \phi(x) \right) =$$

$$\frac{\epsilon}{8} \cdot \left(\frac{1}{2^n} \cdot \sum_{h=9\sqrt{n}}^{n} \binom{n}{h} \cdot \frac{A}{2^{\lfloor L_h \rfloor}} + \sum_{x \in \{0,1\}^n \text{ s.t. } 9\sqrt{n} \leq ||x||} \phi(x) \right) \leq$$

$$\frac{\epsilon}{8} \cdot \left(2 \cdot \frac{1}{2^n} \cdot \sum_{h=9\sqrt{n}}^{n} \binom{n}{h} \cdot \frac{A}{2^{L_h}} + \sum_{x \in \{0,1\}^n \text{ s.t. } 9\sqrt{n} \leq ||x||} \phi(x) \right) \leq$$

$$\frac{\epsilon}{8} \cdot \left(1 + \sum_{x \in \{0,1\}^n \text{ s.t. } 9\sqrt{n} \leq ||x||} \rho(x) \right) \leq \frac{\epsilon}{4}$$

Sample Complexity Bounds for Influence Maximization

Gal Sadeh
Tel Aviv University, Israel
galsdh@gmail.com

Edith Cohen
Google Research, Mountain View, CA, USA
Tel Aviv University, Israel
edith@cohenwang.com

Haim Kaplan
Google Research, Tel Aviv, Israel
Tel Aviv University, Israel
haimk@tau.ac.il

Abstract

Influence maximization (IM) is the problem of finding for a given $s \geq 1$ a set S of $|S| = s$ nodes in a network with maximum influence. With stochastic diffusion models, the influence of a set S of seed nodes is defined as the expectation of its *reachability* over *simulations*, where each simulation specifies a deterministic reachability function. Two well-studied special cases are the *Independent Cascade (IC)* and the *Linear Threshold* (LT) models of Kempe, Kleinberg, and Tardos [31]. The influence function in stochastic diffusion is unbiasedly estimated by averaging reachability values over i.i.d. simulations. We study the IM *sample complexity*: the number of simulations needed to determine a $(1-\epsilon)$-approximate maximizer with confidence $1-\delta$. Our main result is a surprising upper bound of $O(s\tau\epsilon^{-2}\ln\frac{n}{\delta})$ for a broad class of models that includes IC and LT models and their mixtures, where n is the number of nodes and τ is the number of diffusion steps. Generally $\tau \ll n$, so this significantly improves over the generic upper bound of $O(sn\epsilon^{-2}\ln\frac{n}{\delta})$. Our sample complexity bounds are derived from novel upper bounds on the variance of the reachability that allow for small relative error for influential sets and additive error when influence is small. Moreover, we provide a data-adaptive method that can detect and utilize fewer simulations on models where it suffices. Finally, we provide an efficient greedy design that computes an $(1-1/e-\epsilon)$-approximate maximizer from simulations and applies to any submodular stochastic diffusion model that satisfies the variance bounds.

2012 ACM Subject Classification Theory of computation → Machine learning theory

Keywords and phrases Sample complexity, Influence maximization, Submodular maximization

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.29

Funding This research is partially supported by the Israel Science Foundation (Grant No. 1841/14 and Grant No. 1595/19).

1 Introduction

Models for the spread of information among networked entities were studied for decades in sociology and economics [26, 20, 29]. A diffusion process is initiated from a seed set of nodes (entities) and progresses in steps: Initially, only the seed nodes are activated. In each step additional nodes may become active based on the current set of active nodes. The progression can be deterministic or stochastic. The t-stepped influence of a seed set S of nodes is then defined as its expected reachability (total number of active nodes) in t steps.

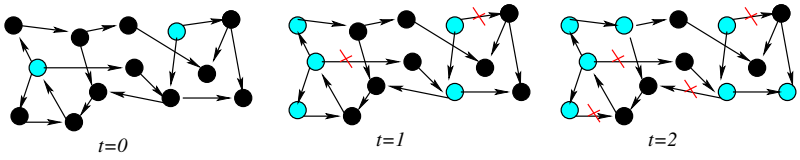


Figure 1 A 2-step diffusion from two seed nodes in a live-edge model.

Influence maximization (IM) is the problem of finding a set S of nodes of specified cardinality $|S| = s$ and maximum *influence*. The IM problem was formulated nearly two decades ago by Richardson and Domingos [17, 40] and inspired by the application of viral marketing. In a seminal paper, Kempe, Klienberg, and Tardos [31] studied stochastic diffusion models and introduced two elegant special cases, the *Independent Cascade (IC)* and *Generalized Threshold (GT)* diffusion models. Their work sparked extensive followup research and large scale implementations [35, 7, 30, 39]. Currently IM is applied in multiple domains with linked entities for tasks as varied as diversity-maximization (the most representative subset of the population) and sensor placement that maximize coverage [32, 4, 34].

We consider *stochastic diffusion models (SDM)* $\mathcal{G}(V)$ over $|V| = n$ nodes that are specified by a distribution $\boldsymbol{\phi} \sim \mathcal{G}$ over sets $\boldsymbol{\phi} := \{\phi_v\}_{v \in V}$ of monotone non-decreasing boolean *activation functions*

$$\phi_v : 2^{V \setminus \{v\}} \to \{0, 1\}.$$

A diffusion process starts with a seed set $S \subset V$ of nodes and $\boldsymbol{\phi} \sim \mathcal{G}$. At step 0 we activate the seed nodes $\texttt{Reach}^0(\boldsymbol{\phi}, S) := S$. The diffusion then proceeds deterministically: At step $t > 0$ all active nodes remain active and we activate any inactive node v where $\phi_v(\texttt{Reach}^{t-1}(\boldsymbol{\phi}, S)) = 1$:

$$\texttt{Reach}^{t+1}(\boldsymbol{\phi}, S) := \{v \in V \mid \phi_v(\texttt{Reach}^t(\boldsymbol{\phi}, S)) = 1\}.$$

The *τ-steps reachability set* of a seed set S is the random variable $\texttt{Reach}^\tau(\boldsymbol{\phi}, S)$ for $\boldsymbol{\phi} \sim \mathcal{G}$ and respectively the τ-steps *reachability*, $\texttt{R}^\tau(S)$, is the random variable that is the number of active nodes $|\texttt{Reach}^\tau(\boldsymbol{\phi}, S)|$ for $\boldsymbol{\phi} \sim \mathcal{G}$. Finally, the influence value of S is defined to be the expectation

$$\texttt{I}^\tau(S) := \mathrm{E}[\texttt{R}^\tau(S)] = \mathrm{E}_{\boldsymbol{\phi} \sim \mathcal{G}}[|\texttt{Reach}^\tau(\boldsymbol{\phi}, S)|].$$

We refer to the case where the diffusion is allowed to progress until there is no growth as *unrestricted* diffusion and this corresponds to $\tau = n - 1$. The influence $\texttt{I}^\tau(S)$ is a monotone set function. We say that an SDM is *submodular* when the influence function is submodular and that it is *independent* if the activation functions ϕ_v of different nodes are independent random variables. The IM problem for seed set size s and τ steps is to find

$$\arg \max_{S : |S| \le k} \texttt{I}^\tau(S).$$

The reader might be more familiar with well-studied special cases of this general formulation. *Live-edge* diffusion models $\mathcal{G}(V, \mathcal{E})$ are specified by a graph $(V, \mathcal{E})$ with $|V| = n$ nodes and $|\mathcal{E}| = m$ directed edges and a distribution $E \sim \mathcal{G}$ over subsets $E \subset \mathcal{E}$ of "live" edges. When expressed as an SDM, the activation functions that correspond to E have $\phi_v(T) = 1$ if and only if there is an edge from a node in T to v in the graph (V, E). Live-edge models are always submodular: This because $|\texttt{Reach}^\tau(E, S)|$, which is the number of nodes reachable

from S in (V, E) by paths of length at most τ, is a coverage function and hence monotone and submodular. Therefore, so is the influence function $\mathtt{I}^{\tau}(S)$, which is an expectation of a distribution over coverage functions. A live-edge model is independent if we only have dependencies between incoming edges to the same node. The Independent Cascade (IC) model is the special case of an independent live-edge model where all edges $e \in \mathcal{E}$ are independent Bernoulli random variables selected with probabilities p_e ($e \in \mathcal{E}$).

Another well-studied class are *generalized threshold* (GT) models [31, 35]. A GT model $\mathcal{G}(V, \boldsymbol{f})$ is specified by a set $\boldsymbol{f} := \{f_v\}_{v \in V}$ of monotone functions $f_v : 2^V \rightarrow [0, 1]$. The randomization is specified by a set of threshold values $\boldsymbol{\theta} \sim \mathcal{G}$ where $\boldsymbol{\theta} := \{\theta_v\}_{v \in V}$. The corresponding activation functions to $\boldsymbol{\theta}$ are

$$\phi_v(T) := \text{Indicator}(\theta_v \leq f_v(T)).$$

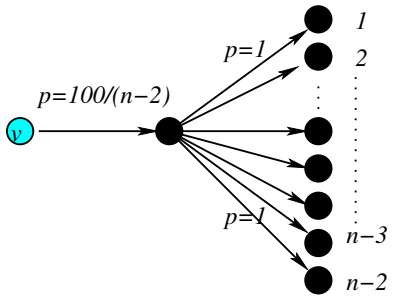
An important subclass are *Independent GT (IGT)* where we require that $\boldsymbol{f}$ are submodular and nodes $v \in V$ have independent threshold values $\theta_v \sim U[0, 1]$. Mossel and Roch [35, 36] proved that IGT models are submodular, which is surprising because the functions $|\mathtt{Reach}^{\tau}(\boldsymbol{\phi}, S)|$ are generally not submodular. Their proof was provided for unrestricted diffusion but extends to the case where we stop the process after τ steps. Finally, Linear threshold (LT) models [26, 31] are a special case of IGT where we have an underlying directed graph and each edge (u, v) is associated with a fixed weight value $b_{uv} \geq 0$ so that for all $v \in V$ $\sum_u b_{uv} \leq 1$ and the functions are defined as the sums $f_v(A) := \sum_{u \in A \cap N(v)} b_{uv}$. Kempe et al showed [31] that each LT model is equivalent to an independent live-edge model.

One of the challenges brought on by the IM formulation is computational efficiency. Kempe et al [31] noted that the IM problem generalizes the classic Max Cover problem even with $\tau = 1$ and a live-edge model with a fixed set of live edges ($p_e = 1$ for all $e \in \mathcal{E}$). Therefore, IM inherits Max Cover's hardness of approximation for ratio better than $1 - (1 - 1/s)^s \geq 1 - 1/e$ [21] for a cover with s sets. On the positive, with submodular models, an approximation ratio of $1 - (1 - 1/s)^s$ can be achieved by the first s nodes of a greedy sequence generated by sequentially adding a node with maximum marginal value [37]. A challenge of applying Greedy with stochastic models, however, is that even point-wise evaluation of the influence function can be computationally intensive. Exact evaluation even for IC models [6] and LT models [8] is #P hard . As for approximation, Kempe et al proposed to work with *averaging oracles*

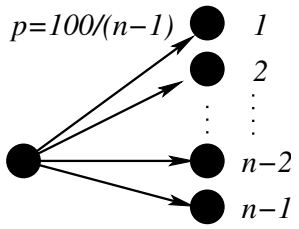
$$\hat{\mathsf{A}}^{\tau}(T) := \frac{1}{\ell} \sum_{i=1}^{\ell} |\mathtt{Reach}^{\tau}(\boldsymbol{\phi}_i, T)|$$

that average the reachability values obtained from a set $\{\boldsymbol{\phi}_i\}_{i=1}^{\ell}$ of i.i.d. simulations. Recall that in the general SDM formulation, a simulation is specified by a set $\boldsymbol{\phi}$ of node activation functions. For live-edge models, a simulation is simply a set of concurrently live edges E. In GT models, a simulation is specified by a set of thresholds $\boldsymbol{\theta}$.

Averaging oracles, which we focus on here, have several appealing properties: First, it is robust compared to estimators tailored to models that satisfy specific assumptions (see related work section) in that for any diffusion model $\mathcal{G}$, also with complex and unknown dependencies (between activation functions of different nodes or between edges in live-edge models), for any set S, $\hat{\mathsf{A}}(S)$ is an unbiased estimate of the exact influence value $\mathtt{I}^{\tau}(S)$ and estimates are accurate as long as the variance of $\mathtt{R}^{\tau}(S)$ is "sufficiently" small. Second, the oracle is constructed directly from simulations and does not require learning or inferring the underlying diffusion model that generated the data [41, 24, 23, 18]. Therefore, the results are not sensitive to modeling assumptions and learning accuracy [9, 27]. Moreover, estimation of



(a) Family of n node graphs. Node v has influence $\mathtt{I}^{\tau=2}(v) = 100$ but variance $\approx 100n$.



(b) Star graphs with n nodes where all edges have marginal probabilities $100/(n-1)$. The center node has influence value 101 and all other nodes have influence 1.

Figure 2 Example live-edge models.

model parameters even for IC models can require a large number of simulations: Consider the family of IC models depicted in the example in Figure 2b where tiny edge probabilities that require many simulations to estimate are critical for IM accuracy. Finally, when the reachability functions $\mathtt{Reach}^\tau(\phi, T)$ are monotone and submodular (as is the case with live-edge models), so is their average $\hat{\mathsf{A}}$, and hence the oracle optimum can be approximated by the greedy algorithm. Prior work addressed the efficiency of working with averaging oracles by improving the efficiency of greedy maximization [32, 25] and applied sketches [10] to efficiently estimate $\hat{\mathsf{A}}(S)$ values [7, 14].

The fundamental question we study here is the *sample complexity* of IM, that is, the number of i.i.d. simulations needed to recover an approximate maximizer of the influence function $\mathtt{I}^\tau$. Formally, for parameters (ϵ, δ), identify a seed set T of size s so that $\Pr\left[\mathtt{I}^\tau(T) \geq (1-\epsilon)\mathrm{OPT}_s^\tau\right] \geq 1-\delta$, where $\mathrm{OPT}_s^\tau := \max_{S||S|\leq s} \mathtt{I}^\tau(S)$ is the exact maximum. Note that the recovery itself is generally computationally hard and the sample complexity only considers the information we can glean from a set of simulations.

Kempe et al provided an upper bound of

$$O\left(\epsilon^{-2} sn \log \frac{n}{\delta}\right) \tag{1}$$

on the sample complexity of the harder *Uniform Relative-Error Estimation (UREE)* problem where for a given (ϵ, δ) we bound the number of simulations so that with probability $1-\delta$, for all subsets S such that $|S| \leq s$, $\hat{\mathsf{A}}(S)$ approximates $\mathtt{I}^\tau(S)$ within relative error of ϵ. The sample complexity of UREE upper bounds that of IM because the oracle maximizer $\arg\max_{S||S|\leq s} \hat{\mathsf{A}}(S)$ must be an approximate maximizer. We provide the argument for the bound (1) here because it is basic and broadly applies to all SDMs: The reachability values $\mathtt{Reach}^\tau(\phi, S)$, and hence their expectation, $\mathtt{I}^\tau(S)$ have values in $[1, n]$. Using the multiplicative Chernoff bound (with values divided by n) we obtain that $O(\epsilon^{-2} n \ln \delta^{-1})$ simulations guarantee a relative error of ϵ with probability at least $(1-\delta)$ for the estimate of any particular set S. Interestingly, this sample complexity bound is tight for point-wise influence estimation even for IC models: The example family of models in Figure 2a is such that $\tau = 2$ and $\Omega(\epsilon^{-2} n)$ simulations are required for estimating the influence value of a single node. The UREE sample complexity bound (1) follows from applying a union bound over all $\binom{n}{s} = O(n^s)$ subsets.

The generic upper bound has prohibitive linear dependence on the number of nodes n (that the example in Figure 2a shows is unavoidable for UREE even for IC models). A simple example shows that we can not hope for an umbrella improvement for IM: Consider the star

graph family of Figure 2b when edges are dependent so that either all edges are live or none is. Clearly $n/100$ simulations are necessary to detect a 1-step approximate maximizer (which must be the actual maximizer). The remaining hope is that we can obtain stronger bounds on the IM sample complexity for models with weaker or no dependencies such as IC and IGT models. This question eluded researchers for nearly two decades.

1.1 Contributions and Overview

We study the sample complexity of influence maximization from averaging oracles computed from i.i.d. simulations. One of our main contributions is an upper bound of

$$O\left(\epsilon^{-2} s\tau \log \frac{n}{\delta}\right) \tag{2}$$

on the IM sample complexity of independent strongly submodular SDMs. Informally, strong submodularity means that the influence function of any "reduced" model (model derived from original one by setting a subset $T \subset V$ of nodes as active) is submodular. The IC and IGT models are special cases of strongly submodular independent SDMs.

Interestingly, we provide similar sample complexity bounds for natural families of models that are not independent: Mixtures of small number of strongly submodular SDMs and what we call *b-dependence* live-edge models that allow for positive dependence of small groups of edges with a shared tail node.

Our bound improves over prior work by replacing the prohibitive linear dependence in the number of nodes n in (1) with the typically much smaller value τ. While on worst-case instances unrestricted diffusions may require $\Omega(n)$ steps, understanding the sample complexity in terms of τ is important: First, IM with explicit step limits [5, 33, 22, 19, 15], is studied for applications where activation time matters. Moreover, due to the "small world" phenomenon [44], in "natural" settings we can expect most activations (even with unrestricted diffusions) to occur within a small number of steps. In the latter case, unrestricted influence values are approximated well by corresponding step-limited influence with $\tau \ll n$.

Our improvement is surprising as generally a linear-in-n number of simulations is necessary for estimating influence values of some nodes or to estimate essential model parameters (for example, the edge probabilities in IC models), and this is the case even when τ is very small. This shows that the maximization problem is in an information sense inherently easier and can circumvent these barriers.

We overview our results and implications – complete proofs can be found in the appendix. We review related work in Section 2 and place it in the context of our results. In Section 3 we formulate quality measures for influence oracles and relate unrestricted and step-limited influence. In particular, we observe that for IM it suffices that the oracle provides good estimates (within a small relative error) of larger influence values. This allows us to circumvent the lower bound for point-wise relative-error estimates shown in Example 2a.

In Section 4 we state our main technical result that for independent strongly submodular SDMs, upper bounds the variance of the reachability of set T $\mathsf{Var}[\mathtt{R}^{\tau}(T)]$ by $\tau \mathtt{I}^{\tau}(T) \max_{v \in V \setminus T} \mathtt{I}^{\tau-1}(v)$. This variance upper bound facilitates estimates with small relative error for sets with larger influence values and additive error for sets with small influence values. We also provide a family of IC models that shows that the linear dependence on τ in the variance bound is necessary. We derive similar variance bounds to mixtures of strongly submodular independent SDMs and b-dependence models. All our subsequent sample complexity bounds apply generically to any SDM (submodular/independent or not) that satisfies variance bounds of this form. In Section 5 we review *averaging oracles* and bound the sample

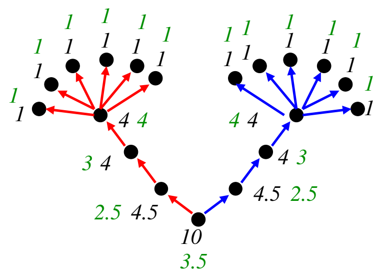


Figure 3 Dependent SDM Example. A live-edge model where with probability $\frac{1}{2}$ all red edges are active and otherwise all blue edges are active. The influence values $\mathtt{I}^4(v)$ are shown in black. Simulation averages and RR samples with full simulations provide unbiased estimates of influence values $\mathrm{E}[\hat{\mathtt{I}}^4(v)] = \mathtt{I}^4(v)$. However, "efficient" RRS, which works with the marginal edge probabilities ($p_e = \frac{1}{2}$) or with decomposed simulations is biased, with $\mathrm{E}[\hat{\mathtt{I}}^4(v)]$ shown in green. We can see that the bias induces large errors and also yields an erroneous maximizer.

complexity using variance upper bounds. In section 6 we present our *median-of-averages* oracle that amplifies the confidence guarantees of the averaging oracle and facilitates a tighter sample complexity bound. In Section 7 we provide a data-adaptive framework that provides guarantees while avoiding the worst-case sample complexity upper bounds on models when a smaller number of simulations suffices. In Section 8 we consider computational efficiency and present a greedy maximization algorithm based on our median-of-averages oracles that returns a $(1-(1-1/s)^s - \epsilon$ approximate maximizer with probability $1-\delta$. The design generically applies to any SDM with a submodular influence function that satisfies the variance bounds.

2 Related work

We focus here on influence estimates obtained from averages of i.i.d. simulations of a SDM. We note that alternative approaches can be more effective for specific families of models. Most notably for IC models, state of the art large-scale greedy approximate maximization algorithms [43, 42, 38, 28] are not based on simulation averages. The estimates are also obtained by building a sample that reflects the influence of each but instead they use a finer building block of i.i.d. *Reverse Reachability (RR) searches*. The random RR search method was proposed in [10] to estimate size of reachability sets in graphs and Borg et al [3] adapted it to IC models.

The method is applicable in principle with any live-edge model: A basic RR search is conducted by selecting a node $v \in V$ uniformly at random and performing a BFS search on reversed edges that is pruned at length τ. The search "flips" edges as their head node is reached according to conditional distribution on $\mathcal{G}$. The index number of the RR search is then added to the sample set of each node that is "reached" by the search. Influence of a subset S can then be unbiasedly estimated from the cardinality of the union of the samples of nodes $v \in S$ and the greedy algorithm can be applied to the sets of samples for approximate maximization. To obtain an approximate influence maximizer we need to perform RR searches until some node has a sample of size $O\left(\epsilon^{-2} s \log(n/\delta)\right)$. In the worst case, this requires $O\left(\epsilon^{-2} sn \log(n/\delta)\right)$ RR searches. For general live-edge models, an independent RR search can always be obtained from a simulation $E \sim \mathcal{G}$ by randomly drawing a node and performing a reverse search from it using edges E. The same simulation, however, can not

generally be reused to generate multiple independent RR searches. This way of obtaining RR searches works for general live-edge models (with arbitrary dependencies) but requires $O(\epsilon^{-2}ns\log(n/\delta))$ simulations, which does not improve over the generic upper bound (1).

The appeal of the RR searches method is that it can be implemented very efficiently for independent live-edge (including IC or LT) models. The total work performed requires only $O(\epsilon^{-2}ms(\log(n/\delta)))$ "edge flips" that can be easily performed using specified edge probabilities p_e for IC models. Moreover, the basic building block of RR searches are local simulations of sets of incoming edges of specified nodes and the full computation requires at most $O(\epsilon^{-2}s\log(n/\delta))$ local simulations for each node. When we have full simulations generated by an independent live-edge model these "local" simulations are independent and the required number of "local simulations" can be obtained by decomposing $O(\epsilon^{-2}s\log(n/\delta))$ full simulations. But the caveat is that this approach breaks the coherence of simulations, as we construct each RR search from components taken from multiple simulations. These "efficient" implementations (i.e. based on decomposed simulations or edge flips according to marginal probabilities) may "catastrophically fail" when dependencies exist: The influence estimates obtained are biased and cause large errors even when the variance is low. Figure 3 shows an example of a simple mixture model (of two degenerate IC models) where "efficient" RRS has large error due to bias but averages of few simulations provide accurate estimates. To summarize, with RRS, the implementation that works with full simulations is robust to dependencies but is inefficient and the efficient implementation breaks ungracefully even with light dependencies. To conclude, both basic approaches to approximate IM, simulation averages and RRS offer distinct advantages: Simulation averages are robust in that they remain unbiased and are accurate on any SDM, including dependent ones, for which the variance is sufficiently small whereas RRS offers more efficiency for purely independent live-edge models.

3 Preliminaries

We consider stochastic diffusion models $\mathcal{G}(V)$ as outlined in the introduction. We denote by $\texttt{Reach}^\tau(\boldsymbol{\phi}, T)$ the τ-steps reachability set of T when we use a specific set $\boldsymbol{\phi}$ of activation functions. We will use the notation $\texttt{Reach}^\tau(T)$ (with the parameter $\boldsymbol{\phi}$ omitted) for the random variable $\texttt{Reach}^\tau(\boldsymbol{\phi}, T)$ obtained when we draw $\boldsymbol{\phi} \sim \mathcal{G}$ according to the model.

3.1 Utility Functions

For simplicity, the discussion in the introduction took the utility of a reachable set to be the number of reachable nodes $\texttt{VReach}^\tau(\boldsymbol{\phi}, T) := |\texttt{Reach}^\tau(\boldsymbol{\phi}, T)|$. Generally, we can consider *utility functions* $H : 2^V \to \Re_+$ that are nonnegative monotone non-decreasing with $H(\emptyset) = 0$:

$$\texttt{VReach}^\tau(\boldsymbol{\phi}, T) := H(\texttt{Reach}^\tau(\boldsymbol{\phi}, T)). \tag{3}$$

Submodular utility is particularly natural and studied by Mossel and Roch [35]. Additive utility is the special case where nodes have nonnegative weights $w : V \to \mathcal{R}^+$ and

$$\texttt{VReach}^\tau(\boldsymbol{\phi}, T) := \sum_{v \in \texttt{Reach}^\tau(\boldsymbol{\phi}, T)} w(v). \tag{4}$$

We consider a diffusion model $\mathcal{G}(V, H)$ together with a utility function H. The random variable $\mathtt{R}^{\tau}_{\mathcal{G}}(T)$ is the utility of the reachable set, that is, $\mathtt{VReach}^{\tau}(\boldsymbol{\phi}, T)$ when $\boldsymbol{\phi} \sim \mathcal{G}$. The influence function is then the expected utility of the reachable set

$$\mathtt{I}^{\tau}(T) := \mathrm{E}[\mathtt{R}^{\tau}(T)] = \mathrm{E}_{\boldsymbol{\phi}\sim\mathcal{G}}\mathtt{VReach}^{\tau}(\boldsymbol{\phi}, T) \ .$$

We denote the maximum influence value of a subset of cardinality s by $\mathrm{OPT}^{\tau}_{s} := \max_{S:|S|\leq s} \mathtt{I}^{\tau}(S)$.

It follows from the definition that for any SDM $\mathcal{G}(V, H)$ with utility H, the influence $\mathtt{I}^{\tau}(T)$ is monotone non-decreasing in τ and in the set T and the optimum values OPT^{τ}_{s} are non-decreasing in τ and s. Generally, influence functions $\mathtt{I}^{\tau}(T)$ of SDMs may not be submodular even when utility is additive. The influence function is known to be submodular for live-edge and for IGT models [35] with submodular utility.

3.2 Reduced Models

We work with the following notion of model reduction. Let $\mathcal{G}(V, H)$ be an independent SDM with submodular utility. For a set of nodes $T \subset V$, we define the *reduced model* $\mathcal{G}'(V', H')$ of $\mathcal{G}$ with respect to T: The reduced model contains the nodes $V' = V \setminus T$. The activation function $\phi'_v \sim \mathcal{G}'$ for $v \in V \setminus T$ are obtained by drawing $\phi_v \sim \mathcal{G}$ conditioned on $\phi_v(T) = 0$ and take

$$\text{for all } S \subset V \setminus (T \cup \{v\}),\ \phi'_v(S) := \phi_v(S \cup T)$$

(Note that since we have independent SDM we can separately consider the distribution of activation functions of each node). The utility in $\mathcal{G}'$ is the marginal utility in $\mathcal{G}$ with respect to T:

$$\text{for all } S \subset V \setminus T,\ H'(S) := H(S \cup T) - H(T).$$

The reduced model $\mathcal{G}'(V', H')$ is also an independent SDM with submodular utility: Activation functions $\boldsymbol{\phi}' \sim \mathcal{G}'$ are independent and monotone and the utility is monotone with $H'(\emptyset) = 0$ and submodular.

3.3 Strongly Submodular SDM

We say that an independent SDM $\mathcal{G}(V, H)$ is *strongly submodular* if the utility function H is submodular and the influence function $\mathtt{I}^{\tau}_{\mathcal{G}'}$ is submodular with any reduced model $\mathcal{G}'$ and step limit $\tau \geq 0$. IC and IGT models are strongly submodular SDMs (see Theorem 19).

The variance and thus sample complexity upper bounds that we present in the sequel apply to any strongly submodular SDM. We will also provide bounds for some dependent families of models. One family is a slight generalization of IC models that we refer to as *b-dependence*. Here edges are partitioned into disjoint groups, where each group contains at most b edges emanating from the same node. The edges in a group must be either all live or none live (are positively dependent).

3.4 Relating Step-Limited and Unrestricted Influence

When unrestricted diffusion from a seed set S is such that most activations occur within τ steps, the unrestricted influence $\mathtt{I}(S)$ is approximated well by τ-step influence $\mathtt{I}^{\tau}(S)$. We can also relate unrestricted influence with small expected steps-to-activation to step-limited

influence: For a seed set S, node v, and length d, we denote by $p(S,v,d)$ the probability that node v is activated in a diffusion from S in step d. For additive utility functions (4) by definition, $\mathtt{I}^\tau(S) = \sum_{v\in V} w(v) \sum_{d\leq \tau} p(S,v,d)$. The *expected length of an activation path* from S (in unrestricted diffusion) is:

$$\overline{D}(S) := \frac{\sum_{v\in V} w(v) \sum_{d\leq n} d \cdot p(S,v,d)}{\mathtt{I}(S)} \ . \tag{5}$$

The following lemma is an immediate consequence of Markov's inequality and shows that τ-stepped influence with $\tau = O(\overline{D}(S))$ approximates well the unrestricted influence:

▶ **Lemma 1.** *For all S and $\epsilon > 0$, $I^{\overline{D}(S)\epsilon^{-1}}(S) \geq (1-\epsilon) I(S)$.*

3.5 Influence Oracles

We say that a set function $\hat{F}$ is an *ϵ-approximation* of another set function F *at a point* T if $\left|\hat{F}(T) - F(T)\right| \leq \epsilon \max\{F(T), \mathrm{OPT}_1(F)\}$, where $\mathrm{OPT}_s(F) := \max_{S||S|\leq s} F(S)$. That is, the estimate $\hat{F}$ has a small relative error for sets T with $F(T) \geq \mathrm{OPT}_1(F)$ and a small absolute error of $\epsilon\mathrm{OPT}_1(F)$ for sets T with $F(T) \leq \mathrm{OPT}_1(F)$. We say that $\hat{F}$ provides a *uniform* ϵ-approximation for all subsets T in a collection C if $\hat{F}$ is an ϵ-approximation for all $T \in C$.

An *influence oracle*, $\hat{\mathtt{I}}^\tau$, is a randomized data structure that is constructed from a set of i.i.d. simulations of a model. The influence oracle, $\hat{\mathtt{I}}^\tau$, defines a set function (we use the same name $\hat{\mathtt{I}}^\tau$ for the set function) that for any input query set $T \subset V$, returns a value $\hat{\mathtt{I}}^\tau(T)$. For $\epsilon < 1$ and $\delta < 1$ we say that an oracle provides (ϵ,δ) *approximation guarantees with respect to I^τ* if for any set T it is an ϵ-approximation with probability at least $1-\delta$. That is

$$\forall T \text{ such that} \mathtt{I}^\tau(T) \geq \mathrm{OPT}_1^\tau, \Pr\left[\frac{\left|\hat{\mathtt{I}}^\tau(T) - \mathtt{I}^\tau(T)\right|}{\mathtt{I}^\tau(T)} \geq \epsilon\right] \leq \delta \ . \tag{6}$$

$$\forall T \text{such that } \mathtt{I}^\tau(T) \leq \mathrm{OPT}_1^\tau, \Pr\left[\left|\hat{I}^\tau(T) - \mathtt{I}^\tau(T)\right| \geq \epsilon \mathrm{OPT}_1^\tau\right] \leq \delta \ . \tag{7}$$

where $\mathrm{OPT}_1^\tau := \mathrm{OPT}_1(I^\tau)$. Example 2a shows that this type of requirement is what we can hope for with an oracle that is constructed from a small number of simulations.

The (ϵ,δ) requirements are *for each* particular set T. If we are interested in stronger guarantees that with probability $(1-\delta)$ the approximation uniformly holds *for all* sets in a collection $\mathcal{C}$, we can use an oracle that provides $(\epsilon, \delta_A = \delta/|\mathcal{C}|)$ guarantees. The ϵ-approximation guarantee for all sets in $\mathcal{C}$ then follow using a union bound argument: The probability that all $|\mathcal{C}|$ sets are approximated correctly is at most $|\mathcal{C}|\delta_A \leq \delta$.

4 Variance Bounds

We consider upper bounds on the variance $\mathsf{Var}\left[\mathtt{R}^\tau(T)\right]$ of the reachability of a set of nodes T that have the following particular form

$$\mathsf{Var}[\mathtt{R}^\tau(T)] \leq c\mathtt{I}^\tau(T) \max\{\mathtt{I}^\tau(T), \max_{v\in V} \mathtt{I}^\tau(v)\} \tag{8}$$

for some $c \geq 1$. The sample complexity bounds we present in the sequel apply to any SDM that satisfies these bounds. In the remaining part of this section we state our variance upper bound for strongly submodular SDMs and extensions and a tight worst-case lower bound for IC models.

4.1 Variance Upper Bound

The following key theorem facilitates our main results. We show that any strongly submodular SDM satisfy the bound (8) with $c = \tau$. The proof is technical and provided in Appendix A.

▶ **Theorem 2** (Variance Upper Bound Lemma). *Let $\mathcal{G}(V, H)$ be a strongly submodular SDM. Then for any step limit $\tau \geq 0$ and a set $T \subset V$ of nodes we have*

$$\mathsf{Var}[R^{\tau}(T)] \leq \tau I^{\tau}(T) \max_{v \in V \setminus T} I^{\tau-1}(v).$$

Some natural dependent SDMs have a variance bound of the form (8): (See Appendix F for proofs.)

▶ **Corollary 3.** *IC models with τ-steps and b-dependence satisfy the bound (8) with $c = 2b\tau$. Any mixture of τ-steps strongly submodular SDMs where each model has probability at least p satisfy the bound (8) with $c = (\tau + 1)/p$.*

4.2 Variance Lower Bound

We provide a family of IC models for which this variance upper bound is asymptotically tight. This shows that the dependence of the variance bound on τ is necessary.

▶ **Theorem 4** (Variance Lower Bound). *For any $\tau > 0$ there is an IC model $\mathcal{G}^{\tau} = (V, \mathcal{E})$ with a node $v \in V$ of maximum influence such that $\mathsf{Var}[R^{\tau}(v)] \geq \frac{1}{12}\tau I^{\tau}(v)^2$.*

Our family of models $\mathcal{G}^{\tau} = (V, \mathcal{E})$ are such that $(V, \mathcal{E})$ is a complete directed binary tree of depth $\tau \geq 1$ rooted at $v \in V$ with all edges directed away from the root and $p_e = 1/2$ for all $e \in \mathcal{E}$. We show (details in Appendix B) that:

$$\mathtt{I}^{\tau}(v) = \tau$$
$$\mathsf{Var}[\mathtt{R}^{\tau}(v)] = \frac{1}{12}\tau(\tau - 1)(2\tau - 1).$$

5 The Averaging Oracle

The *averaging oracle* uses i.i.d. simulations $\{\boldsymbol{\phi}_i\}_{i=1}^{\ell}$. For a query T it returns the average utility of the reachability set of T: $\hat{\mathsf{A}}^{\tau}(T) = \mathsf{Ave}_{i \in [\ell]} \mathtt{VReach}^{\tau}(\boldsymbol{\phi}_i, T) := \frac{1}{\ell}\sum_{i=1}^{\ell} \mathtt{VReach}^{\tau}(\boldsymbol{\phi}_i, T)$. We quantify the approximation guarantees of an averaging oracle in terms of a variance bound of the form (8).

▶ **Lemma 5.** *Consider an SDM that for some $c \geq 1$ satisfies a variance bound of the form (8). Then for any $\epsilon, \delta < 1$, an averaging oracle constructed from $\ell \geq \epsilon^{-2}\delta^{-1}c$ i.i.d. simulations provides (ϵ, δ) guarantees.*

In particular for strongly submodular SDMs, we use the variance bound in Theorem 2 and obtain these approximation guarantees using $\ell \geq \epsilon^{-2}\delta^{-1}\tau$ i.i.d. simulations.

Proof. Using variance properties of the average of i.i.d. random variables, we get that for any query T

$$\mathsf{Var}[\hat{\mathsf{A}}^{\tau}(T)] = \frac{1}{\ell}\mathsf{Var}[\mathtt{R}^{\tau}(T)] \leq \frac{1}{\ell}c\mathtt{I}^{\tau}(T)\max\{\mathtt{I}^{\tau}(T), \mathrm{OPT}_1^{\tau}\} .$$

The claims follow using Chebyshev's inequality that states that for any random variable X and M, $\Pr[|X - \mathrm{E}[X]| \geq \epsilon M] \leq \epsilon^{-2}\mathsf{Var}[X]/M^2$. We apply it to the random variable $\hat{\mathsf{A}}^{\tau}(T)$ that has expectation $I^{\tau}(T)$ and plug in the variance bound. To establish (6) we use $M = \mathtt{I}^{\tau}(T)$ and to establish (7) we use $M = \mathrm{OPT}_1^{\tau}$. ◀

5.1 Sketched Averaging Oracle

For live-edge models with additive utility (4), the query efficiency of the averaging oracle can be improved with off-the-shelf use of τ-step combined reachability sketches [10, 14, 15, 11]. The sketching is according to a sketch-size parameter k that also determines the sketches computation time and accuracy of the estimates that sketches provide. A sketch of size $O(k)$ is computed for each node v so that for any set of nodes S, $\sum_{i=1}^{r} \mathtt{VReach}^{\tau}(E_i, S)$ can be efficiently estimated from the sketches of the nodes $v \in S$. The computation of the sketches from an arbitrary set of simulations $\{E_i\}$ uses at most $\sum_i |E_i| + k \sum_v \max_i d_v(E_i)$ edge traversals, where $d_v(E_i)$ is the maximum in-degree of node v over simulations $\{E_i\}$. In the case of an IC model, the expected number of traversals is $(k+\ell)\sum_e p_e$. Sketching with general node weights can be handled as in [11]. The estimates obtained from the sketches are unbiased with coefficient of variation $1/\sqrt{k-2}$ and are concentrated: Sketches of size $k = O(\epsilon^{-2}\log(\delta^{-1}))$ provide estimates with relative error ϵ with probability $1-\delta$.

6 Confidence Amplification: The Median-of-Averages Oracle

The statistical guarantees we provide for our averaging oracle are derived from variance bounds. The limitation is that the number of simulations we need to provide (ϵ, δ) guarantees is linear in δ^{-1} and therefore the number of simulations we need to provide uniform guarantees (via a union bound argument) grows linearly with the number of subsets. In order to find an approximate optimizer, we would like to have a uniform ϵ-approximation for all the $\binom{n}{s}$ subsets of size at most s but doing so with an averaging oracle would require too many simulations. We adapt to our setting a classic confidence amplification technique [1] to construct an oracle where the number of simulations grows logarithmically in the confidence parameter δ^{-1}.

A *median-of-averages* oracle is specified by a number r of *pools* with ℓ simulations in each pool. The oracle is therefore constructed from $r\ell$ i.i.d. simulations $\boldsymbol{\phi}_{ij}$ for $i \in [r]$ and $j \in [\ell]$.

The simulations of each pool are used in an averaging oracle that for the ith pool ($i \in [r]$) returns the estimates $\hat{\mathsf{A}}_i^{\tau}(T)$. The median-of-averages oracle returns the median value of these r estimates

$$\widehat{\mathsf{mA}}^{\tau}(T) := \operatorname*{Median}_{i\in[r]} \hat{\mathsf{A}}_i^{\tau}(T) = \operatorname*{Median}_{i\in[r]} \operatorname*{Ave}_{j\in[\ell]} \mathtt{VReach}^{\tau}(\boldsymbol{\phi}_{ij}, T) \ . \tag{9}$$

We establish that when the i.i.d simulations are from a model that has variance bound (8) for some $c \geq 1$, the median-of-averages oracle provides (ϵ, δ) approximation guarantees using $112\epsilon^{-2}c\ln\delta^{-1}$ i.i.d. simulations.

▶ **Lemma 6.** *Consider an SDM that for some $c \geq 1$ satisfies the variance bound (8). Then for every ϵ and δ, a median-average oracle $\widehat{\mathsf{mA}}$ organized with $r = 28\ln\delta^{-1}$ pools of $\ell = 4\epsilon^{-2}c$ simulations in each provides (ϵ, δ) approximation guarantees.*

Proof. An averaging oracle with ℓ simulations provides (ϵ, δ_A) approximation guarantees for $\delta_A = 1/4$. Therefore, the probability of correct estimate for any subset is at least $3/4$. We now consider the estimates $\hat{\mathsf{A}}_j$ obtained from the r pools when sorted in increasing order. The estimates that are not correct (too low or too high) will be at the prefix and suffix of the sorted order. The expected number of correct estimates is at least $\mu \geq \frac{3}{4}r$. The probability that the median estimate is not correct is bounded by the probability that number of correct estimates is $\leq r/2$, which is $\leq \frac{2}{3}\mu$. From multiplicative Chernoff bounds, the probability of a sum of Bernoulli random variables beings below $(1-\epsilon')\mu$ is at most $e^{-\epsilon'^2\mu/(2+\epsilon')}$. Using $\epsilon' = 1/3$ we have $\epsilon'^2\mu/(2+\epsilon') = \frac{1}{9}\frac{3}{4}\frac{3}{7}28\ln\delta^{-1} = \ln\delta^{-1}$. ◀

As a corollary, we obtain a sample complexity bound for influence maximization from variance bounds:

▶ **Theorem 7.** *Consider an SDM that satisfies the variance bound* (8) *for some* $c \geq 1$. *Then for any* $\epsilon < 1$ *and* $\delta < 1$, *using* $112\epsilon^{-2}cs \ln \frac{n}{\delta}$ *i.i.d. simulations, we can construct a median-of-averages oracle* $\widehat{mA}$ *such that the oracle optimum*

$$T := \arg \max_{S||S|\leq s} \widehat{mA}(S)$$

satisfies

$$\Pr\left[I^{\tau}(T) \geq (1-2\epsilon)\mathrm{OPT}_s^{\tau}\right] \geq 1-\delta \ .$$

Proof. We construct our median-of-averages oracle with $\ell = 4\epsilon^{-2}c$ and $r = 28 \ln \delta_{MA}^{-1}$ where $\delta_{MA} = \delta/\binom{n}{s}$. From Lemma 6 using a union bound over the $\binom{n}{s}$ sets we obtain that with probability $1-\delta$ the oracle provides a uniform ϵ-approximation for all subsets of size at most s. Let S be a set with maximum influence $\mathtt{I}(S) = \mathrm{OPT}_s^{\tau}$ and let T be the oracle optimum. We have

$$\mathtt{I}(T) \geq (1-\epsilon)\widehat{\mathsf{mA}}(T) \geq (1-\epsilon)\widehat{\mathsf{mA}}(S) \geq (1-\epsilon)^2\mathtt{I}(S) \geq (1-2\epsilon)\mathrm{OPT}_s^{\tau} \ .$$

We comment that the $(1-2\epsilon)$ ratio is not tight and we can obtain a bound closer to $(1-\epsilon)$. This because the particular set S to be approximated more tightly by the oracle (that uses enough simulations to support a union bound). ◀

7 Optimization with Adaptive Sample Size

The bound on the number of simulations we derived in Theorem 7 (through a median-of-averages oracle) and also the naive bound (1) (for the averaging oracle) are worst-case. This is obtained by using enough simulations to have the oracle provide a uniform ϵ-approximation with probability at least $1-\delta$ on any problem instance. To obtain the uniform approximation we applied a union bound over $\binom{n}{s}$ subsets that resulted in an increase in the number of required simulations by an $s \log n$ factor over the base (ϵ, δ) approximation guarantees.

On real data sets a much smaller number of simulations than this worst-case often suffices. We are interested in algorithms that adapt to such data and return a seed set of approximate maximum influence using a respectively smaller number of simulations and while providing statistical guarantees on the quality of the end result. To do so, we apply an adaptive optimization framework [12] (some example applications are [14, 38, 16, 13]). This framework consists of a "wrapper" that take as inputs oracle constructions from simulations and a base algorithm that performs an optimization over an oracle. The wrapper invokes the algorithm on oracles constructed using an increasing number of simulations until a validation condition on the quality of the result is met. The details are provided in Appendix E. We denote by $r(\epsilon, \delta)$ the number of simulations that provides (ϵ, δ) guarantees and we obtain the following results:

▶ **Theorem 8.** *Suppose that on our data the averaging (respectively, median-of-averages) oracle* $\hat{I}$ *has the property that with* r *simulations, with probability at least* $1-\delta$, *the oracle optimum* $T := \arg\max_{S||S|\leq s} \hat{I}(S)$ *satisfies*

$$I^{\tau}(T) \geq (1-\epsilon)\mathrm{OPT}_s^{\tau}.$$

Then with probability at least $1-5\delta$*, when using*

$$2\max\{r, r(\epsilon,\delta)\} + O\left(\epsilon^{-2}c\left(\ln\frac{1}{\delta} + \ln\left(\ln\ln\frac{n}{\delta} + \ln s\right)\right)\right)$$

simulations with the median-of-averages oracle and

$$2\max\{r, r(\epsilon,\delta)\} + O\left(\epsilon^{-2}c\left(\ln\frac{1}{\delta} + \ln\left(\ln\ln\frac{n}{\delta} + \ln n\right)\right)\right)$$

simulations with the averaging oracle, the wrapper outputs a set T *such that* $\mathcal{I}^{\tau}(T) \geq (1-5\epsilon)\mathrm{OPT}^{\tau}_{s}$*.*

The wrapper can also be used with a base algorithm that is an approximation algorithm. For live-edge models, our averaging oracle is monotone and submodular and hence we can apply greedy to efficiently compute a set with approximation ratio at least $1-1/e$ (with respect to the oracle). If we use greedy as our base algorithm we obtain the following:

▶ **Theorem 9.** *If the averaging oracle* $\hat{A}$ *is submodular and has the property that with* $\geq r$ *simulations, with probability at least* $1-\delta$*, it provides a uniform* ϵ*-approximation for all subsets of size at most* s*, then with*

$$2\max\{r, r(\epsilon,\delta)\} + O\left(\epsilon^{-2}c\left(\ln\frac{1}{\delta} + \ln\left(\ln\ln\frac{n}{\delta} + \ln n\right)\right)\right)$$

simulations we can find in polynomial time a $(1-(1-1/s)^s)(1-5\epsilon)$ *approximate solution with confidence* $1-5\delta$*.*

8 Approximate Greedy Maximization

In this section we consider the computational efficiency of maximization over our oracle $\hat{\mathtt{I}}$ that approximates a monotone submodular influence function $\mathtt{I}^{\tau}$. The maximization problem is computationally hard: The brute force method evaluates $\hat{\mathtt{I}}(S)$ on all $\binom{n}{s}$ subsets S of size s in order to find the oracle maximizer. An efficient algorithm for approximate maximization of a monotone submodular function $\hat{F}$ is greedy that sequentially builds a seed set S by adding a node u with maximum marginal contribution $\arg\max_{u\in V}(\hat{F}(S\cup\{u\}) - \hat{F}(S))$ at each step. To implement greedy we only need to evaluate at each step the function on a linear number of subsets $\hat{F}(S\cup\{u\})$ for $u\in V$ and thus overall we do sn evaluations of $\hat{F}$ on subsets. With a monotone and submodular $\hat{F}$, for any $s\geq 1$ the subset T that consists of the first s nodes in a greedy sequence satisfies [37]:

$$\hat{F}(T) \geq (1-(1-1/s)^s)\max_{S||S|\leq s}\hat{F}(S) \geq (1-1/e)\mathrm{OPT}_s(\hat{F})\ .$$

If our functions $\hat{F}$ provides a uniform ϵ-approximation of another function F for all subsets of size at most s, then $F(T)\geq(1-(1-1/s)^s)(1-2\epsilon)\mathrm{OPT}_s(F)$ (See the proof of 7).

The averaging oracle is monotone and submodular [31] when reachability functions are as in live-edge models. Unfortunately our median-of-averages oracle which facilitates tighter bounds on the number of simulations is monotone but may not be submodular even for models where the averaging oracle is submodular. Generally when this is the case, greedy may fail (as highlighted in recent work by Balkanski et al [2]).

Fortunately, greedy is effective on a function $\hat{F}$ that is monotone but not necessarily submodular as long as $\hat{F}$ "closely approximates" a monotone submodular F in that marginal contributions of the form

$$F(u\mid S) := F(S\cup\{u\}) - F(S)$$

are approximated well by $\hat{F}(u\mid S)$ [14]. We apply this to establish the following lemma:

▶ **Lemma 10.** *The greedy algorithm applied to a function $\hat{F}$ that is monotone and provides a uniform ϵ_A-approximation of a monotone submodular function F where $\epsilon_A = \frac{\epsilon(1-\epsilon)}{14s}$ returns a set T such that $F(T) \geq (1-(1-1/s)^s)(1-\epsilon)\mathrm{OPT}_s(F)$.*

Our proof of Lemma 10 generally applies to an approximate oracle $\hat{F}$ of any monotone submodular function F and is presented in Appendix C. For approximate IM we obtain the following as a corollary:

▶ **Theorem 11.** *Consider a submodular SDM $\mathcal{G}(V,H)$ that for some $c \geq 1$ satisfies the variance bound* (8)*. Consider a median-of-averages oracle constructed with $O(\epsilon^{-2}s^3 c \ln \frac{n}{\delta})$ simulations of $\mathcal{G}$ arranged as $r = O(s \ln \frac{n}{\delta})$ pools with $\ell = O(\epsilon^{-2}s^2 c)$ simulations each. Then with probability $1-\delta$, the set T that contains the first s nodes returned by greedy on the oracle satisfies $\mathcal{I}^\tau(T) \geq (1-(1-1/s)^s)(1-\epsilon)\mathrm{OPT}_s^\tau$.*

Proof. From Lemma 6, with appropriate constants, this configuration provides us with $(\epsilon/(14s), \delta)$ approximation guarantees. From Lemma 10 greedy provides the stated approximation ratio. ◀

Greedy on the median-of-averages oracle can be implemented generically for any SDM $\mathcal{G}$ by explicitly maintaining the reachability sets $\texttt{Reach}(\boldsymbol{\phi}_{ij}, \{v\} \cup S)$ for all nodes $v \in V$ in each simulation $\boldsymbol{\phi}_{ij}$ as the greedy selects nodes into the seed set S. For each step, we compute the oracle value (see (9)) and select v for which the value for $\{v\} \cup S$ is maximized:

$$\arg\max_{v \in V \setminus S} \widehat{\mathsf{mA}}^\tau(\{v\} \cup S).$$

We obtain approximation guarantees, however, only when the conditions of monotone submodular influence function and variance bounds are satisfied. For specific families of models, we can consider tailored efficient implementations that incrementally maintain reachability sets and values.

For live-edge models with additive utility (4) we consider an implementation of greedy on a median-of-averages oracle. This can be done by explicit maintenance of reachability sets or by using sketches [10, 14, 15, 11] (see Section 5.1). We obtain the following bounds (proof is deferred to Appendix Section D).

▶ **Theorem 12.** *Let $\mathcal{G}$ be a live-edge model with an additive utility function* (4) *that satifies the variance bound* (8)*. Then greedy on median-of-averages oracle can be implemented with explicit reachability sets in time*

$$O(\epsilon^{-2}s^3 c \ln\left(\frac{n}{\delta}\right)\overline{m}n)\ , \tag{10}$$

where $\overline{m}$ is the average number of edges per simulation (For an IC model, $c = \tau$ and $\mathrm{E}[\overline{m}] = \sum_{e\in\mathcal{E}} p_e$). When using sketches, the time bound is

$$O(\epsilon^{-2}s^3 \ln\frac{n}{\delta}(c\overline{m} + s(m^* + ns)\ln n)), \tag{11}$$

where $m^* = \sum_v \max_{ij} d_v(E_{ij})$. For an IC model, $c = \tau$ and $m^* = \sum_e p_e$ in expectation.

Conclusion

We explore the "sample complexity" of IM on stochastic diffusion models and show that an approximate maximizer (within a small relative error) can be recovered from a small number of simulations as long as the variance is appropriately bounded. We establish the variance

bound for the large class of strongly submodular stochastic diffusion models. This includes IC models (where edges are drawn independently) and IGT models (where node thresholds are drawn independently) and natural extensions that allow for some dependencies. Our sample complexity bound significantly improves over the previous bounds by replacing the linear dependence in the number of nodes by a logarithmic dependence on the number of nodes and linear dependence on the length of the activation paths (which are usually very short). An interesting question for future work is to address the gap between the sample complexity and the larger number of simulations currently needed for greedy maximization.

References

1. N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. *J. Comput. System Sci.*, 58:137–147, 1999.
2. E. Balkanski, A. Rubinstein, and Y. Singer. The limitations of optimization from samples. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, 2017. doi:10.1145/3055399.3055406.
3. C. Borg, M. Brautbar, J. Chayes, and B. Lucier. Maximizing Social Influence in Nearly Optimal Time. In *SODA*, 2014.
4. W. Chen, L. V. S. Lakshmanan, and C. Castillo. *Information and Influence Propagation in Social Networks*. Morgan & Claypool, 2013.
5. W. Chen, W. Lu, and Y. Zhang. Time-critical influence maximization in social networks with time-delayed diffusion process. In *AAAI*, 2012.
6. W. Chen, C. Wang, and Y. Wang. Scalable Influence Maximization for Prevalent Viral marketing in Large-Scale Social Networks. In *KDD*. ACM, 2010.
7. W. Chen, Y. Wang, and S. Yang. Efficient influence maximization in social networks. In *KDD*. ACM, 2009.
8. W. Chen, Y. Yuan, and L. Zhang. Scalable Influence Maximization in Social Networks under the Linear Threshold Model. In *ICDM*, 2010.
9. Wei Chen, Tian Lin, Zihan Tan, Mingfei Zhao, and Xuren Zhou. Robust Influence Maximization. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16. ACM, 2016. doi:10.1145/2939672.2939745.
10. E. Cohen. Size-Estimation Framework with Applications to Transitive Closure and Reachability. *J. Comput. System Sci.*, 55:441–453, 1997.
11. E. Cohen. All-Distances Sketches, Revisited: HIP estimators for massive graphs analysis. *TKDE*, 2015. arXiv:1306.3284.
12. E. Cohen. Multi-Objective Weighted Sampling. In *HotWeb*. IEEE, 2015. full version: arXiv:1509.07445.
13. E. Cohen, S. Chechik, and H. Kaplan. Clustering Small Samples with Quality Guarantees: Adaptivity with One2all pps. In *AAAI*, 2018. arXiv:1706.03607.
14. E. Cohen, D. Delling, T. Pajor, and R. F. Werneck. Sketch-based Influence Maximization and Computation: Scaling up with Guarantees. In *CIKM*. ACM, 2014. arXiv:1408.6282.
15. E. Cohen, D. Delling, T. Pajor, and R. F. Werneck. Distance-Based Influence in Networks: Computation and Maximization. Technical Report cs.SI/1410.6976, arXiv, 2015. arXiv:1410.06976.
16. E. Cohen, N. Grossuag, and H. Kaplan. Processing Top-k Queries from Samples. In *Proceedings of the 2006 ACM conference on Emerging network experiment and technology (CoNext)*. ACM, 2006.
17. P. Domingos and M. Richardson. Mining the network value of customers. In *KDD*. ACM, 2001.
18. N. Du, Y. Liang, M-F. Balcan, and L. Song. Influence Function Learning in Information Diffusion Networks. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, 2014.

19 N. Du, L. Song, M. Gomez-Rodriguez, and H. Zha. Scalable Influence Estimation in Continuous-Time Diffusion Networks. In *NIPS*. NIPS, 2013.

20 D. Easley and J. Kleinberg. *Networks, Crowds, and Markets: Reasoning About a Highly Connected World*. Cambridge University Press, New York, NY, USA, 2010.

21 U. Feige. A threshold of $\ln n$ for approximating set cover. *J. Assoc. Comput. Mach.*, 45:634–652, 1998.

22 M. Gomez-Rodriguez, D. Balduzzi, and B. Schölkopf. Uncovering the Temporal Dynamics of Diffusion Networks. In *ICML*, 2011.

23 M. Gomez-Rodriguez, J. Leskovec, and A. Krause. Inferring Networks of Diffusion and Influence. In *KDD*, 2010.

24 A. Goyal, F. Bonchi, and L. V. S. Lakshmanan. Learning influence probabilities in social networks. In *WSDM*, 2010.

25 A. Goyal, W. Lu, and L.V.S. Lakshmanan. CELF++: Optimizing the Greedy Algorithm for Influence Maximization in Social Networks. In *WWW*. ACM, 2011.

26 M. Granovetter. Threshold Models of Collective Behavior. *American Journal of Sociology*, 83(6), 1978.

27 Xinran He and David Kempe. Robust Influence Maximization. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16. ACM, 2016. doi:10.1145/2939672.2939760.

28 K. Huang, S. Wang, G. S. Bevilacqua, X. Xiao, and L. K. S. Lakshmanan. Revisiting the Stop-and-Stare Algorithms for Influence Maximization. *PVLDB*, 10, 2017.

29 M. O. Jackson. *Social and economic networks*. Princeton University Press, 2010.

30 K. Jung, W. Heo, and W. Chen. IRIE: Scalable and robust influence maximization in social networks. In *ICDM*. ACM, 2012.

31 D. Kempe, J. M. Kleinberg, and É. Tardos. Maximizing the spread of influence through a social network. In *KDD*. ACM, 2003.

32 J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and Glance N. Cost-effective outbreak detection in networks. In *KDD*. ACM, 2007.

33 B. Liu, G. Cong, X. Dong, and Y. Zeng. Time Constrained Influence Maximization in Social Networks. In *ICDM*, 2012.

34 B. Mirzasoleiman, A. Karbasi, R. Sarkar, and A. Krause. Distributed Submodular Maximization: Identifying Representative Elements in Massive Data. In *NIPS*, 2013.

35 Elchanan Mossel and Sébastien Roch. On the submodularity of influence in social networks. In *STOC*, 2007.

36 Elchanan Mossel and Sebastien Roch. Submodularity of Infuence in Social Networks: From Local to Global. arXiv:math/0612046, July 2009. (Accessed on 08/16/2019).

37 G. Nemhauser, L. Wolsey, and M. Fisher. An analysis of the approximations of maximizing submodular set functions. *Mathematical Programming*, 14, 1978.

38 H. T. Nguyen, M. T. Thai, and T. N. Dinh. Stop-and-Stare: Optimal Sampling Algorithms for Viral Marketing in Billion-scale Networks. In *SIGMOD*. ACM, 2016.

39 H. T. Nguyen, M. T. Thai, and T. N. Dinh. A Billion-Scale Approximation Algorithm for Maximizing Benefit in Viral Marketing. *IEEE/ACM Trans. Netw.*, 25(4), 2017.

40 M. Richardson and P. Domingos. Mining knowledge-sharing sites for viral marketing. In *KDD*. ACM, 2002.

41 K. Saito, R. Nakano, and M. Kimura. Prediction of Information Diffusion Probabilities for Independent Cascade Model. In *Knowledge-Based Intelligent Information and Engineering Systems (KES)*, volume 5179 of *Lecture Notes in Computer Science*. Springer, 2008.

42 Y. Tang, Y. Shi, and X. Xiao. Influence Maximization in Near-Linear Time: A Martingale Approach. In *SIGMOD*, 2015.

43 Y. Tang, X. Xiao, and Y. Shi. Influence Maximization: Near-Optimal Time Complexity Meets Practical Efficiency. In *SIGMOD*, 2014.

44 J. Travers and S. Milgram. An Experimental Study of the Small World Problem. *Sociometry*, 32:425–443, December 1969. doi:10.2307/2786545.

A Variance Upper Bound: Proof of Theorem 2

In this section we prove Theorem 2 which upper bounds the variance in strongly submodular SDMs. We start by bounding the variance in a more basic setting of a submodular function over a random subset in Section A.1 (Theorem 14). This will be an ingredient in our main proof provided in Section A.3.

We will be using the following basic tools:

▶ **Lemma 13.** *If X, Y are two random variables on the same probability space and the variance of Y is finite, then:*

$$\mathrm{E}[Y] = \mathrm{E}[\mathrm{E}[Y|X]] \quad \textit{(total expectation)}$$

$$\mathsf{Var}[Y] = \mathrm{E}[\mathsf{Var}[Y|X]] + \mathsf{Var}[\mathrm{E}[Y|X]] \quad \textit{(total variance)}$$

where $\mathrm{E}[Y|X]$ is a random variable that gets the expectation of Y conditioned the value of X and $\mathsf{Var}[Y|X]$ is a random variable that gets the variance of Y conditioned the value of X.

When X is a Bernoulli random variable $X \sim Ber(p)$ then Lemma 13 gives that

$$\mathsf{Var}[Y] = \mathrm{E}[\mathsf{Var}[Y|X]] + \mathsf{Var}[\mathrm{E}[Y|X]] = p\mathrm{V}_1 + (1-p)V_0 + p(1-p)(\mathrm{E}_1 - \mathrm{E}_0)^2.$$

where $\mathrm{V}_0 = \mathsf{Var}[Y|X=0]$, $\mathrm{V}_1 = \mathsf{Var}[Y|X=1]$, $\mathrm{E}_0 = \mathrm{E}[Y|X=0]$, and $\mathrm{E}_1 = \mathrm{E}[Y|X=1]$.

A.1 Submodular Monotone Functions on Random Subsets

Let $S = \{a_i\}_{1<i\leq t}$ be a set with t elements and let $P = \{p_i\}_{1<i\leq t}$ be a set of t probabilities such that p_i is associated with the element a_i. Let X be a random subset of S that contains a_i with probability p_i independently for each $i = 1, ..., t$. That is

$$\forall A \in 2^S : \Pr[X = A] = \prod_{a_i \in A} p_i \prod_{a_i \notin A} (1 - p_i).$$

We say that X is a *random subset of S using probabilities P*.

A *submodular monotone function* f over S is a function with the following properties:

1. $f : 2^S \to R^+$
2. For every $A, B \subset S$ with $A \subset B$ and for every $x \in S \setminus B$ we have that $f(A \cup x) - f(A) \geq f(B \cup x) - f(B)$
3. $A \subset B \Rightarrow f(A) \leq f(B)$

For any singelton $\{a\} \in S$ we write $f(a)$ instead of $f(\{a\})$. Let $\mathtt{M}_f = \max_i f(a_i) - f(\emptyset)$. Our purpose in this subsection is to establish the following:

▶ **Theorem 14.** *Let X be a random subset of S using probabilities P and let f be a submodular monotone function. Then*

$$\mathsf{Var}[f(X)] \leq \mathtt{M}_f \mathrm{E}\left[f(X) - f(\emptyset)\right].$$

We give the following additional definitions and lemmas before proving this theorem.

Let $S_{-i} = S \setminus \{a_i\}$ and let $P_{-i} = P \setminus \{p_i\}$. We define X_{-i} to be a random subset of S_{-i} using the probabilities P_{-i}. Let f_i^0, f_i^1 be submodular functions over S_{-i} defined by $f_i^0(A) = f(A)$ and $f_i^1(A) = f(A \cup \{a_i\})$. Let

$$\mathrm{E}_i^1 = \mathrm{E}[f_i^1(X_{-i})] = \sum_{A \in 2^{S \setminus \{a_i\}}} \Pr[X_{-i} = A] f_i^1(A).$$

and

$$\mathrm{E}_i^0 = \mathrm{E}[f_i^0(X_{-i})] = \sum_{A \in 2^{S \setminus \{a_i\}}} \Pr[X_{-i} = A] f_i^0(A).$$

By our definitions $\mathrm{E}[f(X_{-i})] = \mathrm{E}_i^0$ and from total expectation (Lemma 13), $\mathrm{E}[f(X)] = p_i \mathrm{E}_i^1 + (1 - p_i)\mathrm{E}_i^0$.

▶ **Lemma 15.** *let f be a submodular monotone function over S and X a random subset of S using probabilities P. Then,*

$$\forall i : \mathrm{E}_i^1 - \mathrm{E}_i^0 \leq f(a_i) - f(\emptyset) \leq \mathtt{M}_f.$$

Proof. Since X is obtained by drawing the elements in S independently it follows that

$$\mathrm{E}_i^1 - \mathrm{E}_i^0 = \sum_{A \in 2^{S_{-i}}} \Pr[X_{-i} = A] \big[f(A \cup \{a_i\}) - f(A) \big] \underbrace{\leq}_{\text{submodularity}}$$

$$\sum_{A \in 2^{S \setminus a_i}} \Pr[X_{-i} = A] \big[f(a_i) - f(\emptyset) \big] \leq f(a_i) - f(\emptyset) \leq \mathtt{M}_f.$$ ◀

▶ **Lemma 16.** *for any submodular monotone function f over S and for any index i we have that $\mathtt{M}_{f_i^0} \leq \mathtt{M}_f$ and $\mathtt{M}_{f_i^1} \leq \mathtt{M}_f$.*

Proof. The first inequality follows immediately from our definition since

$$\mathtt{M}_{f_i^0} = \max_{j \neq i} f_i^0(a_j) - f_i^0(\emptyset) = \max_{j \neq i} f(a_j) - f(\emptyset) \leq \mathtt{M}_f.$$

For the second inequality we use submodularity as follows

$$\mathtt{M}_{f_i^1} = \max_{j \neq i} f_i^1(a_j) - f_i^1(\emptyset) = \max_{j \neq i} f(\{a_j \cup a_i\}) - f(a_i) \underbrace{\leq}_{\text{sub-modularity}} \max_j f(a_j) - f(\emptyset) = \mathtt{M}_f.$$

◀

We are now ready for the proof of Theorem 14.

Proof of Theorem 14. The proof is by induction on the size of S.
Base case. Let $S = \{a_1\}, P = \{p_1\}$ we have that

$$\mathrm{E}[f(X)] = p_1 f(a_1) + (1 - p_1) f(\emptyset),$$

and

$$\begin{aligned} \mathsf{Var}[f(X)] &= \mathrm{E}[f(X)^2] - \mathrm{E}^2[f(X)] = p_1 f^2(a_1) + (1 - p_1) f^2(\emptyset) - \big[p_1 f(a_1) + (1 - p_1) f(\emptyset) \big]^2 \\ &= p_1(1 - p_1) \big[f(a_1) - f(\emptyset) \big]^2 \leq \mathtt{M}_f p_1 (1 - p_1) \big[f(a_1) - f(\emptyset) \big]. \end{aligned}$$

It is left to prove that $\mathrm{E}\left[f(X)-f(\emptyset)\right] \geq p_1(1-p_1)\left[f(a_1)-f(\emptyset)\right]$, and indeed we have that

$$\mathrm{E}\left[f(X)-f(\emptyset)\right] = p_1\left[f(a_1)-f(\emptyset)\right] \geq p_1(1-p_1)\left[f(a_1)-f(\emptyset)\right].$$

Inductive Step. Assume the lemma holds for sets of size ℓ and any submodular function f and probabilities P. For a set S with $\ell+1$ elements and a submodular function f over S. Let $j \leq i$ be an arbitrary index.
From the total variance formula in Lemma 13 we know that

$$\mathsf{Var}[f(X)] = p_j \mathrm{V}_j^1 + (1-p_j)\mathrm{V}_j^0 + p_j(1-p_j)\left[\mathrm{E}_j^1 - \mathrm{E}_j^0\right]^2 \ , \tag{12}$$

where $\mathrm{E}_j^1 = \mathrm{E}[f_j^1(X_{-j})]$, $\mathrm{E}_j^0 = \mathrm{E}[f_j^0(X_{-j})]$, $\mathrm{V}_j^1 = \mathsf{Var}[f_j^1(X_{-j})]$, and $\mathrm{V}_j^0 = \mathsf{Var}[f_j^0(X_{-j})]$. By applying the induction hypothesis to S_{-j} with probabilities P_{-j} and $|S_{-j}| = \ell$ and f_j^0 and f_j^1 we get that

$$\mathrm{V}_j^0 \leq \mathtt{M}_{f_j^0}\left[\mathrm{E}_j^0 - f_j^0(\emptyset)\right] \underbrace{\leq}_{\text{Lemma 16}} \mathtt{M}_f\left[\mathrm{E}_j^0 - f(\emptyset)\right],$$

and

$$\mathrm{V}_j^1 \leq \mathtt{M}_{f_j^1}\left[\mathrm{E}_j^1 - f_j^1(\emptyset)\right] \underbrace{\leq}_{\text{Lemma 16}} \mathtt{M}_f\left[\mathrm{E}_j^1 - f(a_j)\right].$$

Substituting these bounds in Equation (12) we get that

$$\begin{aligned}
\mathsf{Var}[f(X)] &\leq p_j \mathtt{M}_f\left[\mathrm{E}_j^1 - f(a_j)\right] + (1-p_j)\mathtt{M}_f\left[\mathrm{E}_j^0 - f(\emptyset)\right] + p_j(1-p_j)\left[\mathrm{E}_j^1 - \mathrm{E}_j^0\right]^2 \\
&= \mathtt{M}_f\left[p_j\mathrm{E}_j^1 + (1-p_j)\mathrm{E}_j^0 - f(\emptyset)\right] + p_j(1-p_j)\left[\mathrm{E}_j^1 - \mathrm{E}_j^0\right]^2 - p_j\mathtt{M}_f\left[f(a_j) - f(\emptyset)\right] \\
&\underbrace{=}_{\text{total expectation}} \mathtt{M}_f\left[\mathrm{E}[f(X)] - f(\emptyset)\right] + p_j(1-p_j)\left[\mathrm{E}_j^1 - \mathrm{E}_j^0\right]^2 - p_j\mathtt{M}_f\left[f(a_j) - f(\emptyset)\right] \\
&\underbrace{\leq}_{\text{Lemma 15}} \mathtt{M}_f\left[\mathrm{E}[f(X)] - f(\emptyset)\right] + p_j\mathtt{M}_f\left[f(a_j) - f(\emptyset)\right]\left[(1-p_j) - 1\right] \\
&\leq \mathtt{M}_f[\mathrm{E}[f(X)] - f(\emptyset)].
\end{aligned}$$

◀

A.2 Properties of Reduced Diffusion Models

We establish some properties of reduced independent SDMs that are needed for our upper bound.

We first show that influence values of nodes in a reduced model can only be lower than respective values in the original model:

▶ **Lemma 17.** *Let $\mathcal{G}'(V \setminus T, H')$ be a reduction of a model $\mathcal{G}(V,H)$.*

For all $v \in V \setminus T$ and $t \geq 0$, $\mathcal{I}^t_{\mathcal{G}'}(v) \leq \mathcal{I}^t_{\mathcal{G}}(v)$. (13)

Proof. Note that $\mathcal{G}'$ is obtained from $\mathcal{G}$ by removing nodes. Therefore respective reachability sets given $\boldsymbol{\phi}$ are such that those in $\mathcal{G}'$ can only be subsets of those in $\mathcal{G}$:

$$\mathtt{Reach}_{\mathcal{G}'}(v, \boldsymbol{\phi}') \subset \mathtt{Reach}_{\mathcal{G}}(v, \boldsymbol{\phi}).$$

Then from monotonicity and submodularity of H we get

$$H'(\mathtt{Reach}_{\mathcal{G}'}(v, \boldsymbol{\phi}')) \leq H'(\mathtt{Reach}_{\mathcal{G}}(v, \boldsymbol{\phi})) \leq H(\mathtt{Reach}_{\mathcal{G}}(v, \boldsymbol{\phi})).$$

(second inequality follows from monotonicity and submodularity of H so that for all $A \subset V \setminus T$, $H'(A) \leq H(A)$.) Therefore,

$$\mathtt{I}^t_{\mathcal{G}'}(v) = \mathrm{E}[\mathtt{VReach}^t_{\mathcal{G}'}(v)] \leq \mathrm{E}[\mathtt{VReach}^t_{\mathcal{G}}(v)] = \mathtt{I}^t_{\mathcal{G}}(v). \quad \blacktriangleleft$$

A convenient property is that reduction preserves strong monotone submodularity:

▶ **Lemma 18.** *A reduction of a strongly monotone submodular model is also strongly monotone submodular.*

Proof. A reduced model with respsect to T_2 of a reduced model of $\mathcal{G}$ with respect to T_1 is a reduced model of $\mathcal{G}$ with respect to $T_1 \cup T_2$. Also note that the reduced utility function H' is also monotone and submodular. ◀

We next show that IC or IGT models with submodular utility are closed under reduction:

▶ **Theorem 19.** *IC and IGT models with submodular utility are strongly submodular SDMs.*

Proof. We first show that IC/IGT models are independent SDMs. In the introduction we expressed IC and IGT models as SDMs: A live-edge model is expressed as an SDM using $\phi_v(T) = 1$ if and only if there is an edge from a node in T to v. The model is independent if for all v the edges incoming to v are independent of all other edges. In IC models all edges are independent and hence IC models are independent SDMs. Recall (from the Introduction) that an IGT model is expressed as an SDM using $\phi_v(T) := \mathrm{Indicator}(\theta_v \leq f_v(T))$. In an IGT model the thresholds θ_v are independent random variables, and hence ϕ_v are independent. Hence, an IGT model is an independent SDM. Submodularity of influence when utlity is submodular is established for IC models in [31] and for IGT models in [35].

Reduction of any model preserves submodularity of the utility and in particular this holds for reduced IC/IGT models. What remains to show is that a reduced IC/IGT model is also an IC/IGT model (respectively). This would conclude the proof of strong submodularity since any IC and IGT models with submodular utility has a submodular influence functions.

To establish this remaining claim we consider IC/IGT models and express the reduction in terms of the activation functions as one in terms of the respective family of models.

We first consider IC models. The reduced IC model $\mathcal{G}'(V \setminus T, \mathcal{E} \setminus (V \times T \cup T \times V)$ is obtained from $\mathcal{G}(V)$ by deleting the nodes T and their incident edges and keeping p_e on remaining edges. This is clearly an IC model. It remains to show that this is equivalent to the reduction of the distribution of activation functions. The conditioning that $\phi_v(T) = 0$ is equivalent to live-edge set E with no edges from T to V. For such edge set for any $S \subset V \setminus (T \cup \{v\})$ we have $\phi'_v(S) = \phi_v(S \cup T) = \phi_v(S)$ which corresponds to E having at least one edge from S to v. From independence of edges, the conditional distribution is also independent and retains the same inclusion probabilities.

We next consider IGT models. The reduction $\mathcal{G}'(V \setminus T, \{f'_v\})$ in terms of activation functions distribution is equivalent to functions is equivalent to modifying the functions so that

$$f'_v(S) := f_v(S \cup T) - f_v(T).$$

The reduced model is clearly an IGT model. The conditioning that $\phi_v(T) = 0$ means that $\theta_v > f_v(T)$. Therefore, the conditional distribution of θ_v provided it was not activated in the first step is uniform on $[f_v(T), 1]$. The probability that $\theta_v > f_v(S \cup T)$ given this conditioning is equal to the probability that $\theta'_v > f_v(S \cup T) - f_v(T) = f'_v(S)$. ◀

A.3 Upper Bound on the Variance in Strongly Submodular SDM

Let $\mathcal{G}(V,H)$ be a τ-stepped diffusion model. We denote by $\mathtt{M}^{\tau}_{\mathcal{G}}(\bar{T})$ the maximum influence of a single node in $\mathcal{G}$ that is not included in T:

$$\mathtt{M}^{\tau}_{\mathcal{G}}(\bar{T}) = \max_{v \in V \setminus T} \mathtt{I}^{\tau}_{\mathcal{G}}(v)$$

As before, we omit $\mathcal{G}$ if it can be understood from the context. We prove the following theorem which is a restatement of Theorem 2.

▶ **Theorem 20.** *Let $\mathcal{G}(V,H)$ be a strongly submodular SDM. Then for any $\tau \geq 0$ and a set of nodes $T \subset V$:*

$$\mathsf{Var}[R^{\tau}(T)] \leq \tau M^{\tau-1}(\bar{T}) I^{\tau}(T).$$

The remaining part of this Subsection contains the proof of the Theorem.

Let T be a set of nodes, and let

$$N(T) = \{v \in V \setminus T \mid \Pr[\phi_v(T) = 1] > 0\}$$

be the nodes that have nonzero probability to be activated if T is active. For the special case of IC models, $N(T) = \{v \notin T \mid \exists (u,v) \in \mathcal{E}, u \in T\}$ is the set of outgoing neighbors of T.

We first consider the case where $N(T)$ is empty. In this case, $\mathtt{Reach}^{\tau}(T) = T$ for all $\tau \geq 0$. Therefore, $\mathsf{Var}[\mathtt{R}^{\tau}(T)] = 0$, $\mathtt{I}^{\tau}(T) = H(T) \geq 0$, and $\mathtt{M}^{\tau-1}(\bar{T}) = 0$ and the claim holds.

We now assume that $N(T)$ is not empty and give a proof by induction on τ.

A.3.1 Base case ($\tau = 1$)

Let

$$p_v := \Pr[\phi_v(T) = 1]$$

be the probability that node v is activated in step 1 provided that the set of nodes T was active at step 0. From independence of the model, the events of activating different nodes $v \in N(T)$ at step 1 are independent. We have that the set of nodes that is active at step 1 is a random subset S of $N(T)$ with probabilities $\{p_v\}$ as defined in Subsection A.1. Moreover, from monotonicity and submodularity of H, the function $f(S) := H(T \cup S) - H(T)$ is monotone and submodular with $f(\emptyset) = 0$. We can therefore apply Theorem 14 to bound the variance of $f(S)$:

$$\mathsf{Var}[f(S)] \leq M_f \mathrm{E}[f(S)]. \tag{14}$$

We now note that

$$\mathrm{E}[f(S)] = \mathrm{E}[H(T \cup S)] - H(T) = \mathtt{I}^1(T) - I^0(T)$$

and

$$\mathsf{Var}[f(S)] = \mathsf{Var}[H(T \cup S)] = \mathsf{Var}[\mathtt{R}^1(T)].$$

For all $v \in N(T)$ we have $f(v) = H(T \cup \{v\}) - H(T) \leq H(v) = \mathtt{I}^0(v)$. Therefore

$$M_f := \max_{v \in N(T)} f(v) \leq \max_{v \in N(T)} \mathtt{I}^0(v) = M^0(\bar{T}).$$

Substituting in (14) we obtain the claim

$$\mathsf{Var}[\mathtt{R}^1(T)] \leq M^0(\bar{T}) \mathtt{I}^1(T).$$

A.3.2 Inductive step

We define $\texttt{Reach}^t_{\mathcal{G}}(T \mid A)$ to be the random variable that is the t-steps reachability of T in a diffusion on $\mathcal{G}$ seeded with T that is conditioned on the event that exactly the nodes $A \subset N(T)$ (and no other nodes) are activated in step 1. Equivalently, we condition on $\boldsymbol{\phi}$ such that for $v \setminus (T \cup A)$, $\phi_v(T) = 0$ and for $v \in A$, $\phi_v(T) = 1$. We respectively define $\texttt{R}^t_{\mathcal{G}}(T \mid A)$ to be the random variable $H(\texttt{Reach}^t_{\mathcal{G}}(T \mid A))$. From definition, we have

$$\texttt{I}^t_{\mathcal{G}}(T) = \sum_{A \subset N(T)} \Pr[\texttt{Reach}^1_{\mathcal{G}}(T) = A \cup T] \mathrm{E}[\texttt{R}^t_{\mathcal{G}}(T \mid A)] = \mathrm{E}_A \mathrm{E}[\texttt{R}^t_{\mathcal{G}}(T \mid A)] \ . \tag{15}$$

We consider the reduced model $\mathcal{G}'$ of $\mathcal{G}$ with respect to T and show that the conditioned $t \geq 1$ steps diffusion from T in $\mathcal{G}$ is equivalent to the unconditioned $t-1$ steps diffusion from A in $\mathcal{G}'$:

▶ **Lemma 21.** *For any $A \subset N(T)$ and $t \geq 1$, the random variables $\texttt{Reach}^{t-1}_{\mathcal{G}'}(A)$ and $\texttt{Reach}^t_{\mathcal{G}}(T|A) \setminus \{T\}$ have identical distribution overs subsets. The random variables $\texttt{R}^{t-1}_{\mathcal{G}'}(A)$ and $\texttt{R}^t_{\mathcal{G}}(T|A) - H(T)$ have identical distributions over values.*

Proof. We first consider $t = 1$. For a draw of conditioned activation functions we have $\texttt{Reach}^1_{\mathcal{G}}(T|A) = T \cup A$. By definition, we also have $\texttt{Reach}^0_{\mathcal{G}'}(A) = A$ and the claim holds.

We next consider $t > 1$. We first observe that in both situations, (i) the reduced model $\mathcal{G}'$ when seeded with A and (ii) the conditioned diffusion in $\mathcal{G}$ seeded with T such that the nodes A are activated in the first step, the progression is determined only by the activation functions on the nodes $V \setminus (T \cup A)$.

We next argue that the distribution of activation functions projected on the nodes $V \setminus (T \cup A)$ is the same in both situations. From independence of $\mathcal{G}$ it suffices to consider separately the activation functions of each node. From definition of a reduced model, we draw for each $v \in V \setminus T$, $\phi_v \sim \mathcal{G}$ conditioned on $\phi_v(T) = 0$. This is exactly what we get for the conditioned diffusion in $\mathcal{G}$.

We can thus match the supports (sets of activations functions) in both situations so that $\boldsymbol{\phi}$ and $\boldsymbol{\phi}'$ are matched when the projections on $V \setminus (T \cup A)$ is the same. The starting points are at steps 0 of the reduced model and step 1 of the conditioned process is A, the progression of new activations is thus the same. Therefore, for any step $t \geq 1$,

$$\texttt{Reach}^t_{\mathcal{G}}(T \mid A, \boldsymbol{\phi}) \setminus T = \texttt{Reach}^{t-1}_{\mathcal{G}'}(A, \boldsymbol{\phi}')$$

and the first claim follows.

For the second claim, note that $\texttt{R}^t_{\mathcal{G}}(T|A) = H(\texttt{Reach}^t_{\mathcal{G}}(T|A))$ and thus

$$\texttt{R}^{t-1}_{\mathcal{G}'}(A) = H'(\texttt{Reach}^{t-1}_{\mathcal{G}'}(A)) = H(\texttt{Reach}^{t-1}_{\mathcal{G}'}(A) \cup T) - H(T) = H(\texttt{Reach}^t_{\mathcal{G}}(T|A)) - H(T)$$

where the equalities are those of distributions. ◀

As immediate corollaries we can relate expectations and variance of as follows:

$$\begin{aligned} \texttt{I}^{\tau-1}_{\mathcal{G}'}(A) &= \mathrm{E}[\texttt{R}^{\tau-1}_{\mathcal{G}'}(A)] = \mathrm{E}[H'(\texttt{Reach}^{\tau-1}_{\mathcal{G}'}(A))] = \mathrm{E}\left[H(\texttt{Reach}^{\tau}_{\mathcal{G}}(T|A))\right] - H(T) \\ &= \mathrm{E}[\texttt{R}^{\tau-1}_{\mathcal{G}}(T|A)] - H(T). \end{aligned} \tag{16}$$

$$\begin{aligned} \mathsf{Var}[\texttt{R}^{\tau-1}_{\mathcal{G}'}(A)] &= \mathsf{Var}[H'(\texttt{Reach}^{\tau-1}_{\mathcal{G}'}(A))] = \mathsf{Var}[H'(\texttt{Reach}^{\tau}_{\mathcal{G}}(T|A) \setminus \{T\})] \\ &= \mathsf{Var}[H(\texttt{Reach}^{\tau}_{\mathcal{G}}(T|A)) - H(T)] = \mathsf{Var}[H(\texttt{Reach}^{\tau}_{\mathcal{G}}(T|A))] = \mathsf{Var}[\texttt{R}^{\tau}_{\mathcal{G}}(T|A)] \ . \end{aligned} \tag{17}$$

A.3.2.1 Total Variance

We define the random variable A to be the subset of $N(T)$ which is activated after the first step. Note that A is a random subset of $N(T)$ using probabilities p_v for $v \in N(T)$ as defined in Section A.1. By the total variance formula we get that

$$\mathsf{Var}[\mathtt{R}^{\tau}_{\mathcal{G}}(T)] = \mathsf{Var}_A[\mathrm{E}[\mathtt{R}^{\tau}_{\mathcal{G}}(T|A)]] + \mathrm{E}_A[\mathsf{Var}[\mathtt{R}^{\tau}_{\mathcal{G}}(T|A)]]. \tag{18}$$

We bound the total variance by separately bounding the two terms.

A.3.2.2 Bound on the first term of the total variance

We consider the reduced model $\mathcal{G}'$ with respect to T and a restriction of the influence function $\mathtt{I}^{\tau-1}_{\mathcal{G}'}$ to the domain that is subsets $A \subseteq N(T)$:

$$f(A) := \mathtt{I}^{\tau-1}_{\mathcal{G}'}(A).$$

From Lemma 21, this function represents the expected marginal utility value of nodes which are not in T that are activated after τ steps if we activate T at step 0 and the set A at step 1.

We first observe that f is monotone and submodular. This because strong monotone submodularity of our model implies that the reduced model is also strongly monotone and submodular, and a restriction of a monotone and submodular function is also monotone and submodular. We establish two helpful properties of f. First,

$$f(\emptyset) = 0\ , \tag{19}$$

which holds for any influence function. Second, using Lemma 17 we obtain

$$\max_{v \in N(T)} f(v) \underbrace{\leq}_{(13)} \max_{v \in N(T)} \mathtt{I}^{\tau-1}_{\mathcal{G}}(v) \leq \mathtt{M}^{\tau-1}(\bar{T})\ . \tag{20}$$

We are now ready to bound the first term of the total variance (18). Our monotone submodular function f and the random subset A using probabilities p_v satisfy the conditions of Theorem 14.

$$\mathsf{Var}_A[\mathrm{E}[\mathtt{R}^{\tau}(T|A)]] = \mathsf{Var}_A[f(A) + H(T)] = \mathsf{Var}_A[f(A)] \underbrace{\leq}_{\text{Theorem 14}} (\max_{v \in N(T)} f(v))\mathrm{E}_A\left[f(A) - f(\emptyset)\right]$$
$$\underbrace{\leq}_{(19),\ (20)} \mathtt{M}^{\tau-1}(\bar{T})\mathrm{E}_A\left[\mathrm{E}[\mathtt{R}^{\tau}(T|A) - H(T)]\right] = \mathtt{M}^{\tau-1}(\bar{T})\left(\mathtt{I}^{\tau}_{\mathcal{G}}(T) - H(T)\right) \tag{21}$$

A.3.2.3 Bound on the second term of the total variance

We next bound the second term of (18) which is the expectation of the variance conditioned on A:

$$
\begin{aligned}
\mathrm{E}_A[\mathsf{Var}[\mathtt{R}^\tau_{\mathcal{G}}(T \mid A)]] &= \sum_{S \subset N(T)} \Pr[A = S]\, \mathsf{Var}[\mathtt{R}^\tau_{\mathcal{G}}(T \mid S)] \\
&= \sum_{S \subset N(T)} \Pr[A = S] \mathrm{E}[\mathtt{R}^\tau_{\mathcal{G}}(T \mid S) - H(T)] \frac{\mathsf{Var}[\mathtt{R}^\tau_{\mathcal{G}}(T \mid S)]}{\mathrm{E}[\mathtt{R}^\tau_{\mathcal{G}}(T \mid S) - H(T)]} \\
&\leq \max_{S \subset N(T)} \frac{\mathsf{Var}[\mathtt{R}^\tau_{\mathcal{G}}(T \mid S)]}{\mathrm{E}[\mathtt{R}^\tau_{\mathcal{G}}(T \mid S) - H(T)]} \sum_{S \subset N(T)} \Pr[A = S] \mathrm{E}[\mathtt{R}^\tau_{\mathcal{G}}(T \mid S) - H(T)] \\
&= \mathrm{E}_A\left[\mathrm{E}[\mathtt{R}^\tau_{\mathcal{G}}(T \mid A) - H(T)]\right] \max_{S \subset N(T)} \frac{\mathsf{Var}[\mathtt{R}^\tau_{\mathcal{G}}(T \mid S)]}{\mathrm{E}[\mathtt{R}^\tau_{\mathcal{G}}(T \mid S) - H(T)]} \\
&\underbrace{=}_{(15),(16),(17)} (\mathtt{I}^\tau_{\mathcal{G}}(T) - H(T)) \max_{S \subset N(T)} \frac{\mathsf{Var}[\mathtt{R}^{\tau-1}_{\mathcal{G}'}(S)]}{\mathtt{I}[\mathtt{R}^{\tau-1}_{\mathcal{G}'}(S)]} \\
&= (\mathtt{I}^\tau_{\mathcal{G}}(T) - H(T)) \frac{\mathsf{Var}[\mathtt{R}^{\tau-1}_{\mathcal{G}'}(S')]}{\mathtt{I}[\mathtt{R}^{\tau-1}_{\mathcal{G}'}(S')]} \qquad (22)
\end{aligned}
$$

Where we take

$$S' = \arg \max_{S \subset N(T)} \frac{\mathsf{Var}[\mathtt{R}^{\tau-1}_{\mathcal{G}'}(S)]}{\mathtt{I}[\mathtt{R}^{\tau-1}_{\mathcal{G}'}(S)]}$$

to be the subset that maximizes the ratio.

Using the induction hypothesis on $(\tau-1)$-stepped influence we get

$$\mathsf{Var}[\mathtt{R}^{\tau-1}_{\mathcal{G}'}(S')] \leq (\tau-1)\mathtt{M}^{\tau-2}_{\mathcal{G}'}(\bar{S'})\mathtt{I}[\mathtt{R}^{\tau-1}_{\mathcal{G}'}(S')]. \qquad (23)$$

We now relate the maximum influence of nodes in the original and reduced models:

$$\mathtt{M}^{\tau-2}_{\mathcal{G}'}(\bar{S'}) = \max_{v \in V \setminus (T \cup S')} \mathtt{I}^{\tau-2}_{\mathcal{G}'}(v) \underbrace{\leq}_{(13)} \max_{v \in V \setminus (T \cup S')} \mathtt{I}^{\tau-2}_{\mathcal{G}}(v) \leq \mathtt{M}^{\tau-2}_{\mathcal{G}}(\overline{T \cup S'}) \leq \mathtt{M}^{\tau-1}_{\mathcal{G}}(\bar{T}). \qquad (24)$$

From (22) using (23) and (24) we obtain

$$\mathrm{E}_A[\mathsf{Var}[\mathtt{R}^\tau(T|A)]] \underbrace{\leq}_{(23)} \left(\mathtt{I}^\tau_{\mathcal{G}}(T) - H(T)\right)(\tau-1)\mathtt{M}^{\tau-2}_{\mathcal{G}'}(\bar{S'}) \underbrace{\leq}_{(24)} \left(\mathtt{I}^\tau_{\mathcal{G}}(T) - H(T)\right)(\tau-1)\mathtt{M}^{\tau-1}_{\mathcal{G}}(\bar{T})\,. \qquad (25)$$

A.3.2.4 Combining the bounds of the first and second terms

The claim of the Theorem follows using total variance (18) and the bounds on the first term (21) and second term (25).

B Variance lower bound construction

▶ **Lemma 22.** *Let $\mathcal{G}$ be complete binary tree where each edge has probability $\frac{1}{2}$ and let $h(u)$ be the height of the node u. Then, $\mathcal{I}^\tau(u) = h(u)$ and $\mathsf{Var}[R^\tau(u)] = \frac{1}{2}\sum_{i=0}^{h(u)-1} i^2 = \frac{(h(u)-1)h(u)(2h(u)-1)}{12}$.*

Proof. By induction on the height of the of the node.

Base step ($h(u) = 1$). It is clear that $\mathtt{I}^1(u) = 1$ and $\mathsf{Var}\left[\mathtt{R}^1(u)\right] = 0$ since u is a leaf.

Inductive step. u has two neighbors and each is reached with probability $\frac{1}{2}$. Let v_1 and v_2 be the neighbors of u and let X_1, X_2 be random variables that indicate if $(u, v_1), (u, v_2)$ were activated respectively. The variables X_1, X_2 are Bernoulli random variables with $p = \frac{1}{2}$, hence, $\mathrm{E}[X_1] = \mathrm{E}[X_2] = \frac{1}{2}$ and $\mathsf{Var}[X_1] = \mathsf{Var}[X_2] = \frac{1}{4}$. Since the graph is a tree, the reachabilities of v_1 and v_2 are independent random variables, so we can simply write:

$$\mathtt{R}^\tau(u) = 1 + X_1\mathtt{R}^{\tau-1}(v_1) + X_2\mathtt{R}^{\tau-1}(v_2)$$

The variable $\mathtt{R}^{\tau-1}(v_1)$ and $\mathtt{R}^{\tau-1}(v_2)$ are identical and $X_1, \mathtt{R}^{\tau-1}(v_1)$ and $X_2, \mathtt{R}^{\tau-1}(v_2)$ are independent random variables, Thus,

$$\mathtt{I}^\tau(u) = \mathrm{E}[\mathtt{R}^\tau(u)] = 1 + \mathrm{E}[X_1]\mathrm{E}[\mathtt{R}^{\tau-1}(v_1)] + \mathrm{E}[X_2]\mathrm{E}[\mathtt{R}^{\tau-1}(v_2)] = 1 + 2\frac{1}{2}\mathrm{E}[\mathtt{R}^{\tau-1}(v_1)]$$

$$\underbrace{=}_{\text{induction's hypothesis}} 1 + h(u) - 1 = h(u)$$

The computation of the variance is similar:

$$\mathsf{Var}[\mathtt{R}^\tau(u)] = \mathsf{Var}[X_1\mathtt{R}^{\tau-1}(v_1)] + \mathsf{Var}[X_2\mathtt{R}^{\tau-1}(v_2)] = 2\,\mathsf{Var}[X_1\mathtt{R}^{\tau-1}(v_1)]$$

For two independent random variables A, B holds that: $\mathsf{Var}[AB] = \mathsf{Var}[A]\,\mathsf{Var}[B] + \mathsf{Var}[A]\mathrm{E}^2[A] + \mathsf{Var}[B]\mathrm{E}^2[B]$, we have that:

$$\mathsf{Var}[\mathtt{R}^\tau(u)] = 2\left(\mathsf{Var}[X_1]\,\mathsf{Var}[\mathtt{R}^{\tau-1}(v_1)] + \mathsf{Var}[X_1]\mathrm{E}^2[\mathtt{R}^{\tau-1}(v_1)] + \mathsf{Var}[\mathtt{R}^{\tau-1}(v_1)]\mathrm{E}^2[X_1]\right)$$

$$= 2\left(\frac{1}{2}\,\mathsf{Var}[\mathtt{R}^{\tau-1}(v_1)] + \frac{1}{4}\mathrm{E}^2[\mathtt{R}^{\tau-1}(v_1)]\right) = \mathsf{Var}[\mathtt{R}^{\tau-1}(v_1)] + \frac{1}{2}(h(u)-1)^2$$

$$\underbrace{=}_{\text{induction's hypothesis}} \frac{1}{2}\sum_{i=0}^{h(u)-1} i^2 = \frac{(h(u)-1)h(u)(2h(u)-1)}{12}.$$

◀

▶ **Theorem 23.** *There is a model $\mathcal{G}$ and a set of nodes T such that $\frac{\mathsf{Var}[\mathtt{R}^\tau(T)]}{\mathtt{M}^\tau(T)\mathtt{I}^\tau(T)} \geq \frac{\tau}{12}$.*

Proof. Lemma 22 shows that for every node $u \in \mathcal{G}$, $\mathtt{I}^\tau(u) = h(u)$ and $\mathsf{Var}[\mathtt{R}^\tau(u)] = \frac{(h(u)-1)h(u)(2h(u)-1)}{12}$. It follows that the root r has the largest influence $\mathtt{I}^\tau(r) = \tau$ and $\mathsf{Var}[\mathtt{R}^\tau(u)] = \frac{(\tau-1)t(2\tau-1)}{12}$, Furthermore $\mathtt{M}^\tau(\bar{r}) = \tau - 1$ since the nodes of the largest influence in $V \setminus r$ are the children of r. We conclude that:

$$\frac{\mathsf{Var}[\mathtt{R}^\tau(r)]}{\mathtt{M}^\tau(\bar{r})\mathtt{I}^\tau(r)} = \frac{(\tau-1)\tau(2\tau-1)}{\tau(\tau-1)12} = \frac{2\tau-1}{12} \underbrace{\geq}_{\tau\geq 1} \frac{\tau}{12}.$$

◀

C Greedy Optimization with an Approximately Submodular Oracle

In this section we present the proof of Lemma 10. We show that our approximation guarantees imply that the application of greedy on $\hat{F}$ generates a sequence that is an approximate greedy sequence (in the sense of Lemma 24) with respect to F.

We first state a helpful Lemma [14] that establishes that it suffices that $\hat{F}(u \mid S)$ to approximate the marginal contributions

$$F(u \mid S) := F(S \cup \{u\}) - F(S)$$

.

▶ **Lemma 24** ([14]). *Given a monotone submodular function F, an approximate greedy algorithm that for some $\epsilon \in [0,1)$ selects at each step an element u such that $F(u \mid S) \geq (1-\epsilon)\max_v F(v \mid S)$ has approximation ratio $\geq (1-(1-1/s)^s)(1-\epsilon)$.*

Proof. It is easy to see that the approximation ratio of ϵ-approximate greedy is $1-(1-(1-\epsilon)/s)^s$. It therefore suffices to establish that this expression is larger than $(1-(1-1/s)^s)(1-\epsilon)$ for $\epsilon \in [0,1]$. Equivalently, we need to show that for all $s \geq 2$ and $x \in [0,1]$

$$(1-(1-x)/s)^s - (1-x)(1-1/s)^s - x \leq 0 \ .$$

This follows from equality holding for $x=0$ and $x=1$ and the function being concave up (second derivative is positive). ◀

Proof of Lemma 10. Consider a monotone non-negative $\hat{F}$ that is a uniform ϵ_A-approximation of a monotone non-negative F with $\epsilon_A = \frac{\epsilon(1-\epsilon)}{14s}$. By definition of ϵ-approximation (see Section 3.5), $\left|\hat{F}(T) - F(T)\right| \geq \epsilon_A \max\{F(T), \mathrm{OPT}_1(F)\}$ for all S with $|S| \leq s$. Therefore,

$$\text{if } F(S) \geq (1-\epsilon)\mathrm{OPT}_1(F) \text{ then } \frac{\left|\hat{F}(S) - F(S)\right|}{F(S)} \leq \frac{\epsilon}{14s} \tag{26}$$

$$\text{if } F(S) \leq (1-\epsilon)\mathrm{OPT}_1(F) \text{ then } \hat{F}(S) \leq \left(1-\frac{\epsilon}{2}\right)\mathrm{OPT}_1(F) \ . \tag{27}$$

Inequality (26) follows immediately when $F(S) \geq \mathrm{OPT}_1(F)$ because the relative error is at most $\epsilon_A \leq \frac{\epsilon}{14s}$. For $(1-\epsilon)\mathrm{OPT}_1(F) \leq F(S) < \mathrm{OPT}_1(F)$ we have absolute error being at most $\epsilon_A \mathrm{OPT}_1(S)$ which is a relative error of at most $\epsilon_A/(1-\epsilon) \leq \frac{\epsilon}{14s}$. Inequality (27) follows from the absolute error being at most $\epsilon_A \mathrm{OPT}_1(F)$ and $\epsilon_A \leq \epsilon/2$.

We establish that these conditions imply that greedy on $\hat{F}$ on the prefix of the greedy sequence where $F(S) \leq \frac{3}{4}\mathrm{OPT}_s(F)$ is actually approximate greedy (as in the conditions of Lemma 24) with respect to F. Note that $1-(1-1/s)^s \geq 3/4$ for $s \geq 2$ and thus the prefix restriction does not limit generality. The claim will then follow from Lemma 24.

For $s=1$, it follows from Equations (26) and (27) , that the first element of a greedy sequence with respect to $\hat{F}$, $\arg\max_u \hat{F}(u)$, satisfies $\hat{F}(u) \geq (1-\frac{\epsilon}{14})\mathrm{OPT}_1(F)$. Therefore from the second iteration and on, we have a set S for which the relative error bounds in Equation (26)) applies.

We consider the marginal contributions $F(u \mid S)$ for any node u. We have

$$\begin{aligned} |\hat{F}(S) - F(S)| &\leq \frac{\epsilon}{14s}F(S) \\ |\hat{F}(S\cup\{u\}) - F(S\cup\{u\})| &\leq \frac{\epsilon}{14s}F(S\cup\{u\}) = \frac{\epsilon}{14s}(F(S)+F(u \mid S)) \ . \end{aligned}$$

We use these inequalities to bound the absolute error of (any) marginal influence estimate by

$$\begin{aligned} \left|\hat{F}(u \mid S) - F(u \mid S)\right| &= \left|\hat{F}(S\cup\{u\}) - \hat{F}(S) - F(S\cup\{u\}) + F(S)\right| \\ &\leq \left|\hat{F}(S\cup\{u\}) - F(S\cup\{u\}\right| + \left|\hat{F}(S) - F(S)\right| \\ &\leq \frac{\epsilon}{7s}F(S) + \frac{\epsilon}{14s}F(u \mid S) \ . \end{aligned} \tag{28}$$

We now consider the node $v = \arg\max_{u\in V} F(u \mid S)$ with maximum marginal contribution to S with respect to F and its contribution value

$$\Delta := F(v \mid S) \geq \frac{1}{s}(\mathrm{OPT}_s - F(s)) \ .$$

Thus, when $F(S) \leq \frac{3}{4}\mathrm{OPT}_s$,

$$\Delta \geq \frac{1}{3s}F(S). \tag{29}$$

By applying (28) to v we get that

$$\hat{F}(v \mid S) \geq \Delta - \frac{\epsilon}{7s}F(S) - \frac{\epsilon}{14s}\Delta$$

Therefore the node $v' = \arg\max_v \hat{F}(v' \mid S)$ with maximum marginal contribution according to $\hat{F}$ satisfies

$$\hat{F}(v' \mid S) \geq \hat{F}(v \mid S) \geq \Delta - \frac{\epsilon}{7s}F(S) - \frac{\epsilon}{14s}\Delta.$$

By using (28) again, substituting (29), and using that fact that $s \geq 2$:

$$F(v' \mid S) \geq \Delta - 2\frac{\epsilon}{7s}F(S) - 2\frac{\epsilon}{14s}\Delta \tag{30}$$

$$\geq \Delta - \epsilon\Delta(\frac{1}{7s} + \frac{6}{7}) \geq \Delta\,(1-\epsilon)\ . \tag{31}$$

Therefore, the greedy sequence according to $\hat{F}$ is an approximate greedy sequence according to F and satisfies the conditions of Lemma 24. Therefore the resulting sequence yields an approximation ratio at least $(1-(1-1/s)^s)(1-\epsilon)$. ◀

D Greedy for Live-Edge Models

Proof of Theorem 12. For the first bound, we explicitly maintain for each node $u \in V$, for each pool, the reachability set of u in the simulations of the pool (and its cardinality). The dominant term in the cost of computation is performing a BFS from each node in each of the $r\ell$ simulations that is truncated at distance τ. The total computation time is

$$O(r\ell\overline{m}n) = O(\epsilon^{-2}s^3 c \ln\left(\frac{n}{\delta}\right)\overline{m}n)\ , \tag{32}$$

where

$$\overline{m} = \frac{1}{\ell r}\sum_{i=1}^{r}\sum_{j=1}^{\ell}|E_{ij}|$$

is the average number of edges per simulation. For an IC model, $\mathrm{E}[\overline{m}] = \sum_{e\in\mathcal{E}} p_e$. When a node u is selected into the seed set we remove all nodes in its reachability set from the reachability sets of all other nodes. The removal cost can be "charged" to the initial reachability computation.

The dependence of the computation time on the graph size can be improved by using combined reachability sketches [10, 14, 15, 11] instead of maintaining the reachability sets explicitly (see Section 5.1). The sketch size needed in order to provide the required accuracy of $O(\epsilon/s)$ (as in Theorem 10) uniformly for all subsets of size at most s is $k = O(\epsilon^{-2}s^3 \ln n)$. We compute a sketch for each node in each of the r pools, so in total we have rn node sketches. The construction time of these sketches has a term $\sum_{ij}|E_{ij}| = r\ell\overline{m}$ linear in the total size of simulations and a term for sketch constructions which is a product of the number of pools r and the construction time for each pool. The per-pool construction time is as described in Section 5.1 and is bounded by k (sketch size) visits for each node, each involving reverse

traversals of incoming edges of the node in some simulation. The per-pool construction time for an IC model is $O(k(n + \sum_e p_e))$ in expectation. The time with arbitrary simulations for pool i is $O(k(n + \sum_v \max_j d_v(E_{ij})))$. In total over all pools, the construction time is dominated by $O(r(\ell\overline{m} + k(n + m^*))$, where $m^* = \sum_v \max_{ij} d_v(E_{ij})$ for arbitrary simulations and $m^* = \sum_e p_e$ for simulations generated by an IC model.

The sketches improve the computation time of greedy. Each iteration of greedy uses the (precomputed) union sketch of the current seed set S in each pool. It then examines the sketches of each $v \in V$ to compute the estimate of the averaging oracle $\hat{\mathsf{A}}(S \cup \{v\})$ in each pool. This operations takes $O(knr)$ in total for the iteration. Therefore s iterations of greedy maximization takes $O(knrs)$ using the sketches. Combining the construction cost of the sketches using $r = O(s \ln \frac{n}{\delta})$ and $\ell = O(\epsilon^{-2} s^2 c)$ and the greedy implementation over the sketches we obtain a total bound on the computation time of

$$\begin{aligned} O(r\ell\overline{m} + kr(m^* + ns)) &= O(r(\ell\overline{m} + k(m^* + ns))) \\ &= O(s \ln \frac{n}{\delta}(\epsilon^{-2} s^2 c\overline{m} + \epsilon^{-2} s^3 (m^* + ns) \ln n) \\ &= O(\epsilon^{-2} s^3 \ln \frac{n}{\delta}(c\overline{m} + s(m^* + ns) \ln n)) \ . \end{aligned}$$ ◀

E Optimization with Adaptive Sample Size

The pseudocode for our wrapper is provided in Algorithm 1. The inputs to the wrapper are a base algorithm $\mathcal{A}$ and two constructions of oracles from sets of simulations. The first construction produces an oracle, $\hat{F}_v$, that we use for validation. The second construction produces oracles, $\hat{F}_x$, that are provided as input to $\mathcal{A}$ to perform the optimization. The oracles provide an approximation of our influence function $\mathtt{I}^\tau(S)$ with non-uniform guarantees. For specified (ϵ, δ) we use the expressions $r_v(\epsilon, \delta)$ or $r_x(\epsilon, \delta)$ for the number of simulations required to obtain (ϵ, δ) guarantees (in the sense of Section 3.5). This gives us a relation between ϵ, δ, and a number of simulations. When constructing an oracle with a given number of simulations r and a specified ϵ, we can determine the confidence δ we have from ϵ and r. The oracles that we consider have the property that for a fixed ϵ, δ decreases at least linearly with the number of simulations. (i.e., when we double the number of simulations δ decreases by at least a factor of 2.)

The wrapper first determines an upper bound ($\lceil \log_2 M/r_x(\epsilon, \delta) \rceil$) on the maximum number of iterations it performs (based on the initial number and the simulation budget) and constructs a validation oracle that provides guarantees for a small number of sets (queries) which equals this maximum number of iterations. It then starts with a set $\mathcal{R}$ of $r_x(\epsilon, \delta)$ simulations that suffice for the oracle $\hat{F}_x$ to provide (non-uniform) (ϵ, δ) approximation guarantees. The wrapper repeats the following: It constructs an "optimization" oracle $\hat{F}_x$ using the set of simulations $\mathcal{R}$ and applies $\mathcal{A}$ over $\hat{F}_x$ to obtain a set T. The wrapper terminates when $\hat{F}_v(T)$ is close to $\hat{F}_x(T)$ or when our simulation budget of M is exceeded. Otherwise, we double the number of simulations in our set $\mathcal{R}$ and repeat.

The wrapped algorithm $\mathcal{A}$ can be an exact or approximate optimizer. It is applied to the oracle function and therefore its quality guarantees are with respect to how well the oracle value $\hat{F}_x(T)$ of the output set T approximates the oracle optimum $\max_{S||S|\leq s} \hat{F}_x(S)$. The wrapper extends the approximation guarantees that $\mathcal{A}$ provides (with respect to the oracle) to a guarantee with respect to the influence function while avoiding the worst-case number of simulations needed for a uniform approximation.

We first establish some basic properties.

Algorithm 1 Optimization Wrapper.

Input: (i) Two oracle constructions from simulations: $\hat{F}_v$ (validation) and $\hat{F}_x$ (optimization) that with $r_v(\epsilon,\delta)$ (resp., $r_x(\epsilon,\delta)$) simulations provide (ϵ,δ) guarantees. (ii) An optimization algorithm $\mathcal{A}$ that applies to the optimization oracle $\hat{F}_x$ and returns a subset. (iii) M: Bound on maximum number of simulations. (iv) Parameters $\epsilon > 0$ and $\delta > 0$.

$r \leftarrow r_x(\epsilon,\delta)$ `// #simulations for` $\hat{F}_x$ `to provide` (ϵ,δ) `guarantees`

`// Build validation oracle`

$\delta_v \leftarrow \frac{\delta}{\lceil \log_2 M/r \rceil}$; $r_v \leftarrow r_v(\epsilon,\delta_v)$ `// #simulations for validation oracle`

$\hat{F}_v \leftarrow$ validation oracle from r_v i.i.d simulations that provides (ϵ,δ_v) guarantees

$\mathcal{R} \leftarrow \perp$ `// Initialize set of i.i.d simulations for optimization`

repeat

  Add r fresh i.i.d simulations to set $\mathcal{R}$

  $\hat{F}_x \leftarrow$ optimization oracle from simulations $\mathcal{R}$ that provides $(\epsilon,*)$ guarantees `// * determined by` $|\mathcal{R}|$

  $T \leftarrow \mathcal{A}(\hat{F}_x)$ `// Optimize over the oracle`

  if $\hat{F}_v(T) \geq \frac{(1-2\epsilon)}{1+\epsilon}\hat{F}_x(T)$ **then**

    return T, $\hat{F}_v(T)$

  else

    $r \leftarrow 2r$

until $|\mathcal{R}| + r_v > M$

▶ **Lemma 25.** *Let S be a set with maximum influence (with $I^\tau(S) = \mathrm{OPT}_s^\tau$). With probability at least $1-2\delta$, all the optimization oracles $\hat{F}_x$ constructed by the wrapper have $(1-\epsilon)\mathrm{OPT}_s^\tau \leq \hat{F}_x(S) \leq (1+\epsilon)\mathrm{OPT}_s^\tau$.*

Proof. The probability that $(1-\epsilon)\mathrm{OPT}_s^\tau \leq \hat{F}_x(S) \leq (1+\epsilon)\mathrm{OPT}_s^\tau$ fails for the first oracle is at most δ. The number of $\hat{F}_x$ uses simulations doubles in each iteration and all our constructions are such that the confidence parameter δ decreases at least linearly with the number of simulations. We therefore obtain that the sequence of failure probabilities for $(1-\epsilon)\mathrm{OPT}_s^\tau \leq \hat{F}_x(S) \leq (1+\epsilon)\mathrm{OPT}_s^\tau$ is geometric and sums up to at most 2δ. ◀

As an immediate corollary we obtain:

▶ **Corollary 26.** *Under the conditions of Lemma 25, the oracle optimum in all iterations satisfies*

$$\max_{T||T|\leq s} \hat{F}_x(T) \geq (1-\epsilon)\mathrm{OPT}_s^\tau \ .$$

The following is immediate from the construction of the validation oracle.

▶ **Lemma 27.** *With probability at least $1-\delta$, the validation oracle has relative error at most ϵ on all tests in which the input set T is such that $I^\tau(T) \geq \mathrm{OPT}_1^\tau$ and absolute error at most $\epsilon\mathrm{OPT}_1^\tau$ otherwise.*

Proof. The wrapper performs at most $\lceil \log_2 M/r \rceil$ iterations before it stops, in each iteration the validation oracle fails to provide an ϵ-approximation with probability at most δ_v. Therefore, by union bound, the probability that the algorithm fails to provide an ϵ-approximation in at least one round is at most $\delta_v \lceil \log_2 M/r \rceil \leq \delta$. ◀

▶ **Lemma 28.** *Assume that our data and our optimization oracle with r or more simulations, are such that with probability at least $1-\delta$, the optimum of the oracle is an approximate optimizer, that is:*

$$(1+\epsilon)\mathrm{OPT}_s^\tau \geq \max_{S||S|\leq s} \hat{F}_x(S) \geq (1-\epsilon)\mathrm{OPT}_s^\tau \tag{33}$$

$$\mathcal{I}^\tau(\arg\max_{S||S|\leq s} \hat{F}_x(S)) \geq (1-\epsilon)\mathrm{OPT}_s^\tau \tag{34}$$

and assume that the algorithm $\mathcal{A}$ returns the oracle optimum. Then with probability at least $1-5\delta$, the wrapper terminates after at most $2\max\{r, r_x(\epsilon,\delta)\} + r_v$ simulations and returns T such that $\mathcal{I}^\tau(T) \geq (1-5\epsilon)\mathrm{OPT}_s^\tau$.

Proof. First we show that the wrapper returns a set T with the required properties with probability at most $1-3\delta$ and then we show that the number of iterations the wrapper does before it stops is smaller than M with probability of at most $1-2\delta$.

From Lemma 25, with probability at least $1-2\delta$ in all iterations $\mathcal{A}$ returns T for which $\hat{F}_x(T) \geq (1-\epsilon)\mathrm{OPT}_s^\tau$. The validation succeeds only if $\hat{F}_v(T) \geq \frac{(1-2\epsilon)}{1+\epsilon}\hat{F}_x(T) \geq \frac{(1-\epsilon)(1-2\epsilon)}{1+\epsilon}\mathrm{OPT}_s^\tau$. From Lemma 27 with probability at least $1-\delta$ in all iterations we have

$$\hat{F}_v(T) \leq \max\{(1+\epsilon)\mathtt{I}^\tau(T), \mathtt{I}^\tau(T) + \epsilon\mathrm{OPT}_1^\tau\} .$$

Therefore, with probability $1-3\delta$ the set T returned by the wrapper satisfies

$$\frac{(1-\epsilon)(1-2\epsilon)}{1+\epsilon}\mathrm{OPT}_s^\tau \leq \max\{(1+\epsilon)\mathtt{I}^\tau(T), \mathtt{I}^\tau(T) + \epsilon\mathrm{OPT}_1^\tau\} .$$

If $(1+\epsilon)\mathtt{I}^\tau(T) > \mathtt{I}^\tau(T) + \epsilon\mathrm{OPT}_1^\tau$ then $\mathtt{I}^\tau(T) \geq \frac{(1-\epsilon)(1-2\epsilon)}{(1+\epsilon)^2}\mathrm{OPT}_s^\tau \geq (1-5\epsilon)\mathrm{OPT}_s^\tau$. Otherwise, we have that $\mathtt{I}^\tau(T) \geq \left(\frac{(1-\epsilon)(1-2\epsilon)}{1+\epsilon} - \epsilon\right)\mathrm{OPT}_s^\tau \geq (1-5\epsilon)\mathrm{OPT}_s^\tau$.

We have to show that with probability at least $1-2\delta$ within $2\max\{r, r_x(\epsilon,\delta)\} + r_v$ simulations the wrapper returns such a set T to finish the proof. Consider the first iteration where $|\mathcal{R}| \geq r$. By Equations (33) and (34) with probability at least $1-\delta$ we have that $\mathtt{I}^\tau(T) \geq (1-\epsilon)\mathrm{OPT}_s^\tau$ and $(1+\epsilon)\mathrm{OPT}_s^\tau \geq \hat{F}_x(T) \geq (1-\epsilon)\mathrm{OPT}_s^\tau$ .By Lemma 27 we have that with probability at least $1-\delta$, the validation oracle satisfies that $\hat{F}_v(T) \geq (1-\epsilon)\mathtt{I}^\tau(T)$ or $\hat{F}_v(T) \geq \mathtt{I}^\tau(T) - \epsilon\mathrm{OPT}_s^1 \geq \mathtt{I}^\tau(T) - \epsilon\mathrm{OPT}_s^\tau$. By the last two statements we have that with probability of at least $1-2\delta$:

$$\hat{F}_v(T) \geq (1-\epsilon)\mathtt{I}^\tau(T) \geq (1-\epsilon)^2\mathrm{OPT}_s^\tau \geq (1-2\epsilon)\mathrm{OPT}_s^\tau \geq \frac{(1-2\epsilon)}{1+\epsilon}\hat{F}_x(T)$$

or

$$\hat{F}_v(T) \geq \mathtt{I}^\tau(T) - \epsilon\mathrm{OPT}_s^\tau \geq (1-2\epsilon)\mathrm{OPT}_s^\tau \geq \frac{(1-2\epsilon)}{1+\epsilon}\hat{F}_x(T). \qquad \blacktriangleleft$$

Theorem 8, which we restate below to provide reading fluency, now follows as a corollary.

▶ **Theorem 29** (Theorem 8)**.** *Suppose that on our data the averaging (respectively, median-of-averages) oracle $\hat{F}$ has the property that with r simulations, with probability at least $1-\delta$, the oracle optimum $T := \arg\max_{S||S|\leq s} \hat{F}(S)$ satisfies*

$$\mathcal{I}^\tau(T) \quad \geq \quad (1-\epsilon)\mathrm{OPT}_s^\tau.$$

Then with probability at least $1-5\delta$*, when using* $2\max\{r, r(\epsilon,\delta)\} + O\left(\epsilon^{-2}c\left(\ln\frac{1}{\delta}+\ln\left(\ln\ln\frac{n}{\delta}+\ln s\right)\right)\right)$ *simulations with the median-of-averages oracle and* $2\max\{r, r(\epsilon,\delta)\} + O\left(\epsilon^{-2}c\left(\ln\frac{1}{\delta}+\ln\left(\ln\ln\frac{n}{\delta}+\ln n\right)\right)\right)$ *simulations with the averaging oracle, the wrapper outputs a set* T *such that* $\mathcal{I}^\tau(T) \geq (1-5\epsilon)\mathrm{OPT}_s^\tau$*.*

Proof of Theorem 8. We analyze here the number of simulations required using the averaging oracles and the median-of-averages oracles, in both cases we use median-of-averages oracles for validation. In both cases $r_v = r(\epsilon,\delta_v) = O\left(\epsilon^{-2}c\log\frac{1}{\delta_v}\right)$, where $\delta_v = \frac{\delta}{\lceil\log_2\frac{M}{r_x}\rceil}$. By Lemma 28 the number of simulations is at most $2\max\{r, r_x(\epsilon,\delta)\} + r_v$. M and r_x get different values for each oracle.

Median-of-averages oracles analysis. We have that $r_x = O(\epsilon^{-2}c\ln\delta^{-1})$ by Lemma 6 and we set $M = O(\epsilon^{-2}cs\ln\frac{n}{\delta})$ by Theorem 7. Simple calculation shows that:

$$\frac{1}{\delta_v} = \frac{\lceil\ln\frac{s\ln\frac{n}{\delta}}{\ln\frac{1}{\delta}}\rceil}{\delta} \leq \frac{1}{\delta}\left(\ln s + \ln\left(\ln\frac{n}{\delta}\right)\right).$$

Therefore,

$$r_v = O\left(\epsilon^{-2}c\left(\ln\frac{1}{\delta}+\ln\left(\ln\ln\frac{n}{\delta}+\ln s\right)\right)\right).$$

Averaging oracles analysis. We have $r_x = O(\epsilon^{-2}c\delta^{-1})$ and we set $M = O\left(\epsilon^{-2}sn\ln\frac{n}{\delta}\right)$ according to the respective worst-case guarantees on the number of simulations specified in (1). A simple calculations shows:

$$\frac{1}{\delta_v} = \frac{\lceil\ln\frac{\delta sn\ln\frac{n}{\delta}}{c}\rceil}{\delta} \leq \frac{1}{\delta}\left(\ln\ln\frac{n}{\delta} + 2\ln n\right)$$

Therefore,

$$r_v = O\left(\epsilon^{-2}c\left(\ln\frac{1}{\delta}+\ln\left(\ln\ln\frac{n}{\delta}+\ln n\right)\right)\right). \quad \blacktriangleleft$$

We next consider cases where the algorithm $\mathcal{A}$ is approximate (may not return the oracle optimizer). We assume in these cases that the optimization oracles $\hat{F}_x$ when constructed with a given number of simulations provide, with high probability, uniform ϵ-approximation for all $\binom{n}{s}$ subsets of cardinality at most s:

$$\forall T \text{ such that } |T|<s,\ \left|\hat{F}(T) - \mathtt{I}^\tau(T)\right| \leq \epsilon\max\{\mathtt{I}^\tau(T), \mathrm{OPT}_1^\tau\}.$$

We first show that a very weak assumption on $\mathcal{A}$ suffices to guarantee termination with good probability.

▶ **Lemma 30.** *If the optimization oracle* $\hat{F}_x$ *when constructed with* r *or more simulations provides uniform* ϵ*-approximation with probability at least* $1-\delta$*, and the algorithm* $\mathcal{A}$ *returns* T *such that* $\hat{F}_x(T) \geq (1-\epsilon)\mathrm{OPT}_1^\tau$*. Then with probability at least* $1-2\delta$ *the wrapper will terminate after using at most* $2\max\{r, r_x(\epsilon,\delta)\} + r_v$ *simulations.*

Proof. Consider the first iteration where $\hat{F}_x$ is constructed using at least r simulations. Let T be the set that $\mathcal{A}$ returns at this iteration. Since $\hat{F}_x$ provides uniform ϵ-approximation we have that $\hat{F}_x(T) \leq (1+\epsilon)\mathtt{I}^\tau(T)$ with probability at least $1-\delta$. Combining this with our assumption we get that $\mathtt{I}^\tau(T) \geq \mathrm{OPT}_1^\tau$ and by Lemma 27 we have that with probability at least $1-\delta$ if $\mathtt{I}^\tau(T) \geq \mathrm{OPT}_1^\tau$ then $\hat{F}(T) \geq (1-\epsilon)\mathtt{I}^\tau(T)$ and if $\frac{1-\epsilon}{1+\epsilon}\mathrm{OPT}_1^\tau \leq \mathtt{I}^\tau(T) \leq \mathrm{OPT}_1^\tau$ then $\hat{F}_v(T) \geq \mathtt{I}^\tau(T) - \epsilon\mathrm{OPT}_1^\tau \geq \mathtt{I}^\tau(T) - \frac{\epsilon(1+\epsilon)}{1-\epsilon}\mathtt{I}^\tau(T)$. Combining we obtain that $\hat{F}_v(T) \geq \frac{1-2\epsilon}{1+\epsilon}\hat{F}_x(T)$, and thus the validation condition holds. ◀

We next consider algorithms $\mathcal{A}$ that guarantees some approximation ratio ρ.

▶ **Theorem 31.** *Suppose that our optimization oracle when constructed with r or more simulations provides uniform ϵ-approximation with probability at least $1-\delta$. Assume now that the algorithm $\mathcal{A}$ returns a set T such that*

$$\hat{F}_x(T) \geq \rho \max_{S||S|\leq s} \hat{F}_x(S) \ .$$

Then, the set T returned by our wrapper satisfies $\mathtt{I}^\tau(T) \geq \rho(1-5\epsilon)\mathrm{OPT}_s^\tau$ with probability of at least $(1-3\delta)$.

Proof. Consider an optimal set S (with $\mathtt{I}^\tau(S) = \mathrm{OPT}_s^\tau$). By Lemma 25 with probability at least $1-2\delta$ all our oracles have $(1-\epsilon)\mathrm{OPT}_s^\tau \leq \hat{F}_x(S) \leq (1+\epsilon)\mathrm{OPT}_s^\tau$ are within $(1\pm\epsilon)\mathrm{OPT}_s^\tau$. By the assumption, the sets T returned by $\mathcal{A}$ in all iterations have $\hat{F}_x(T) \geq \rho \max_{S||S|\leq s} \hat{F}_x(S) \geq \rho(1-\epsilon)\mathrm{OPT}_s^\tau$. When the wrapper stops we have that $\hat{F}_x(T) \leq \frac{1+\epsilon}{1-2\epsilon}\hat{F}_v(T)$ and by Lemma 27 we have with probability at least $1-\delta$ that $\hat{F}_v(T) \leq \max\{(1+\epsilon)\mathtt{I}^\tau(T), \mathtt{I}^\tau(T) + \epsilon\mathrm{OPT}_1^\tau\}$.

Combining, we have that with probability at least $1-3\delta$,

$$\rho(1-\epsilon)\mathrm{OPT}_s^\tau \leq \rho\hat{F}_x(S) \leq \hat{F}_x(T) \leq \frac{1+\epsilon}{1-2\epsilon}\hat{F}_v(T) \leq \frac{1+\epsilon}{1-2\epsilon}\max\{(1-\epsilon)\mathtt{I}^\tau(T), \mathtt{I}^\tau(T) + \epsilon\mathrm{OPT}_s^\tau\}.$$

Now, a simple calculation shows that $\mathtt{I}^\tau(T) \geq \rho(1-5\epsilon)\mathrm{OPT}_s^\tau$. ◀

We can prove now Theorem 9 (restated for reading fluency):

▶ **Theorem 32** (Theorem 9). *If the averaging oracle $\hat{A}$ has the property that with $\geq r$ simulations, with probability at least $1-\delta$, it provides a uniform ϵ-approximation for all subsets of size at most s, then with $2\max\{r, r(\epsilon,\delta)\} + O\left(\epsilon^{-2}c\left(\ln\frac{1}{\delta} + \ln\left(\ln\ln\frac{n}{\delta} + \ln n\right)\right)\right)$ simulations we can find in polynomial time a $(1-(1-1/s)^s)(1-5\epsilon)$ approximate solution with confidence $1-5\delta$.*

Proof. The averaging oracle is monotone and submodular [31] and therefore greedy can efficiently recover a set T such that $\hat{F}_x(T) \geq (1-(1-1/s)^s)\max_{S||S|\leq S}\hat{F}_x(S)$.

By Lemma 30, the wrapper terminates using at most $2\max\{r, r_x(\epsilon,\delta)\} + r_v$ with probably at least $1-2\delta$. Applying Theorem 31 with $\rho = (1-(1-1/s)^s)$, we get that $\mathtt{I}^\tau(T) \geq (1-(1-1/s)^s)(1-5\epsilon)\mathrm{OPT}_s^\tau$ with probability at least $1-3\delta$. Hence, with probability at least $1-5\delta$ the wrapper applied with greedy finds $(1-(1-1/s)^s)(1-5\epsilon)$-approximate solution using $2\max\{r, r_x(\epsilon,\delta)\} + r_v$ simulations. ◀

F Variance Bounds for Dependent Models

In this section we provide a proof for Corollary 3. We consider a natural extensions of IC models, b-dependence, that allow for some dependencies between edges and mixtures of IC and IGT models. For these extensions, we establish upper bounds of the form (8) on the variance of the reachability of a set of nodes.

We bound the variance by constructing for each dependent model a corresponding IC model and then apply the variance upper bound established in Section A for IC models.

For mixture models we provide a generic derivation that bounds the variance of the mixture by variance of components.

F.1 b-Dependence Models

The first family we consider are *b-dependence* models, which we define as follows. We assume that all edges with the same tail node are partitioned into disjoint groups where each group is of size at most b. The edges of each group B are either all active together with probability p_B or none is active with probability $1 - p_B$. The special case where all groups are of size 1 corresponds to an IC model (where all edges are independent).

▶ **Theorem 33.** *Let $\mathcal{G}$ be a b-dependence model for some $b \geq 1$. For every set T we have that:*

$$\mathit{Var}[R^{\tau}_{\mathcal{G}}(T)] \leq 2b\tau I^{\tau}_{\mathcal{G}}(T) \max_{v \in V \setminus T} I^{\tau}_{\mathcal{G}}(v)$$

Proof. We construct an IC model $\mathcal{G}'$ from the given b-dependence model $\mathcal{G}$. The model $\mathcal{G}'$ is defined over the set of nodes V of $\mathcal{G}$ together with an additional set D of dummy nodes. The construction has the properties that 2τ-step influence in $\mathcal{G}'$ from a set of nodes $T \subseteq V$ is equal to τ-stepped influence of T in $\mathcal{G}$. Furthermore, the variances of the sizes of the 2τ-step reachbility of T in $\mathcal{G}'$ is the same as the variance of the τ step reachability of T in $\mathcal{G}$. The influence of each dummy node in $\mathcal{G}'$ is at most $b \max_{v \in V} \mathtt{I}^{\tau}(v)$. The claim follows from these properties and Theorem 2.

Here is a formal description of our reduction.

We start by putting in $\mathcal{G}'$ the set V of the nodes of $\mathcal{G}$. Then for every group $B = \{(u, v_1), (u, v_2), ..., (u, v_\ell)\}$ in $\mathcal{G}$ we do the following:

1. Add a new dummy node v_B to $\mathcal{G}'$, and add to $\mathcal{G}'$ the edge (u, v_B) and give it the probability p_B. We assign weight 0 to v_B so that it does not contribute to the reachability of any set of nodes.
2. we create edges (v_B, v_i) for every $1 \leq i \leq \ell$, each such edge has probability 1.

Let $T \subset V$ be a set of nodes in $\mathcal{G}$. It follows from our construction that for any set of nodes $B \subset V$ the probability that $\mathtt{R}^{2\tau}_{\mathcal{G}'}(T) = B$ is the same as the probability that $\mathtt{R}^{\tau}_{\mathcal{G}}(T) = B$. This implies that for any $T \subseteq V$

$$\mathtt{I}^{2\tau}_{\mathcal{G}'}(T) = \mathtt{I}^{\tau}_{\mathcal{G}}(T) \ ,$$

and

$$\mathsf{Var}[\mathtt{R}^{2\tau}_{\mathcal{G}'}(T)] = \mathsf{Var}[\mathtt{R}^{\tau}_{\mathcal{G}}(T)] \ .$$

Each dummy node is connected to at most k original nodes, hence, $\mathtt{I}^{2\tau}_{\mathcal{G}'}(v)$ is bounded by $b \max_{v \in V} \mathtt{I}^{2\tau}_{\mathcal{G}'}(v)$. By Theorem 20 it follows that for every set of nodes T in $\mathcal{G}'$:

$$\mathsf{Var}[\mathtt{R}^{2\tau}_{\mathcal{G}'}(T)] \leq 2\tau \mathtt{I}^{2\tau}_{\mathcal{G}'}(T) \max_{v \in V \setminus T} \mathtt{I}^{2\tau}_{\mathcal{G}'}(v).$$

Combining all these observations together, we get that

$$\mathsf{Var}[\mathtt{R}^{\tau}_{\mathcal{G}}(T)] \leq 2b\tau \mathtt{I}^{\tau}_{\mathcal{G}}(T) \max_{v \in V \setminus T} \mathtt{I}^{\tau}_{\mathcal{G}}(v) \qquad \blacktriangleleft$$

This Theorem can be generalized to more complex dependencies. For example it holds for any distribution on subsets of the outgoing edges from each node that we can realize by a distribution on disjoint subsets where we draw each subset with certain probability, and take the union of the subset which we draw.

F.2 Mixture of IC and IGT Models

The second family of dependent models we consider is a mixture of IC and IGT models.

Consider a set of models $\mathcal{G}_i(V)$ for $i \in [r]$ and respective probabilities p_i such that $\sum_{i=1}^{r} p_i = 1$. We define a mixture model $\mathcal{G}(V)$ as follows. To draw $\boldsymbol{\phi} \sim \mathcal{G}$, we first draw $i \in [r]$ according to probabilities p_i and then return $\boldsymbol{\phi} \sim \mathcal{G}_i$.

We provide two proofs for the variance bound of the mixture. The first is direct and applies to any mixture of models that satisfies the variance bound of Theorem 2), and in particular to mixtures of strongly submodular SDMs. The second proof is specific to live-edge models and based on a reduction to an IC model.

▶ **Theorem 34.** *Consider a model $\mathcal{G}$ that is a mixture of r models $\mathcal{G}_i$ with probabilities p_i that satisfy the variance bound of Theorem 2. Then for all $T \subset V$,*

$$\mathit{Var}[R^{\tau}_{\mathcal{G}}(T)] \leq \frac{\tau+1}{\min_i p_i} I^{\tau}_{\mathcal{G}}(T) \max_{v \in V} I^{\tau}_{\mathcal{G}}(v)$$

Proof. We first relate the influence of T in the mixture model to the influence of T in the components.

$$\mathtt{I}^{\tau}_{\mathcal{G}}(T) = \mathrm{E}[\mathtt{R}^{\tau}_{\mathcal{G}}(T)] = \sum_{i=1}^{r} p_i \mathrm{E}[\mathtt{R}^{\tau}_{\mathcal{G}_i}(T)] = \sum_{i=1}^{r} p_i \mathtt{I}^{\tau}_{\mathcal{G}_i}(T) \ . \tag{35}$$

This holds to any set T and any τ. Therefore we also obtain the inequality

$$\mathtt{M}^{\tau}_{\mathcal{G}}(\bar{T}) = \max_{v \in V \setminus T} I^{\tau}_{\mathcal{G}}(v) = \max_{v \in V \setminus T} \sum_{i=1}^{r} p_i \mathtt{I}^{\tau}_{\mathcal{G}_i}(v) \leq \sum_{i=1}^{r} p_i \max_{v \in V \setminus T} \mathtt{I}^{\tau}_{\mathcal{G}_i}(v) = \sum_{i=1}^{r} p_i \mathtt{M}^{\tau}_{\mathcal{G}_i}(\bar{T}) \ . \tag{36}$$

It also follows that we can bound the influence values on the component by the respective ones in the mixture: $\mathtt{I}^{\tau}_{\mathcal{G}_i}(T) \leq \frac{1}{p_i} \mathtt{I}^{\tau}_{\mathcal{G}}(T)$ and thus

$$\max_i \mathtt{I}^{\tau}_{\mathcal{G}_i}(T) \quad \leq \quad \frac{1}{\min_i p_i} \mathtt{I}^{\tau}_{\mathcal{G}}(T) \tag{37}$$

$$\max_i \mathtt{M}^{\tau}_{\mathcal{G}_i}(\overline{T}) \quad \leq \quad \frac{1}{\min_i p_i} \mathtt{M}^{\tau}_{\mathcal{G}}(\overline{T}) \ . \tag{38}$$

The random variable $\mathtt{R}^{\tau}_{\mathcal{G}}(T)$ can be expressed as a sum of of r products of random variables:

$$\mathtt{R}^{\tau}_{\mathcal{G}}(T) = \sum_{i=1}^{r} X_i \mathtt{R}^{\tau}_{\mathcal{G}_i}(T) \ ,$$

where X_i are Bernoulli with probabilities p_i. The random variables $\{\mathtt{R}^{\tau}_{\mathcal{G}_i}(T)\}$ are independent of each other and also are independent from (the joint distribution of) $\{X_i\}$. The variables $\{X_i\}$ have negative dependence as $\sum_i X_i = 1$ and thus the products $X_i \mathtt{R}^{\tau}_{\mathcal{G}_i}(T)$ are also negatively dependent and thus

$$\mathsf{Var}[\mathtt{R}^{\tau}_{\mathcal{G}}(T)] \leq \sum_i \mathsf{Var}[X_i \mathtt{R}^{\tau}_{\mathcal{G}_i}(T)] \ .$$

We will instead bound the variance of a surrogate random variable

$$Y = \sum_i X_i \mathtt{R}^{\tau}_{\mathcal{G}_i}(T)$$

that has the same sum of products but with the variables $\{X_i\}$ being independent of each other and hence the products are also independent. We have

$$\mathsf{Var}[Y] = \sum_i \mathsf{Var}[X_i \mathtt{R}^\tau_{\mathcal{G}_i}(T)] \geq \mathsf{Var}[\mathtt{R}^\tau_{\mathcal{G}}(T)] \tag{39}$$

We next express the variance of each product using variance properties of the product of two independent random variables. For $i \in [r]$:

$$\begin{aligned}
\mathsf{Var}[X_i \mathtt{R}^\tau_{\mathcal{G}_i}(T)] &= \mathsf{Var}[X_i]\,\mathsf{Var}[\mathtt{R}^\tau_{\mathcal{G}_i}(T)] + \mathrm{E}[X_i]^2\,\mathsf{Var}[\mathtt{R}^\tau_{\mathcal{G}_i}(T)]^2 + \mathsf{Var}[X_i]\mathrm{E}[\mathtt{R}^\tau_{\mathcal{G}_i}(T)]^2 \\
&= p_i(1-p_i)\,\mathsf{Var}[\mathtt{R}^\tau_{\mathcal{G}_i}(T)] + p_i^2\,\mathsf{Var}[\mathtt{R}^\tau_{\mathcal{G}_i}(T)] + p_i(1-p_i)\mathtt{I}^\tau_{\mathcal{G}_i}(T)^2 \\
&= p_i\,\mathsf{Var}[\mathtt{R}^\tau_{\mathcal{G}_i}(T)] + p_i(1-p_i)\mathtt{I}^\tau_{\mathcal{G}_i}(T)^2 \ .
\end{aligned}$$

Therefore, invoking Theorem 2 to bound the variance for each IC model $\mathcal{G}_i$ and then using (37) and (38) and finally using (35) and (36) we get

$$\begin{aligned}
\mathsf{Var}[Y] &= \sum_{i=1}^r p_i \left(\mathsf{Var}[\mathtt{R}^\tau_{\mathcal{G}_i}(T)] + p_i(1-p_i)\mathtt{I}^\tau_{\mathcal{G}_i}(T)^2\right) \\
&\leq \sum_{i=1}^r p_i\,\mathsf{Var}[\mathtt{R}^\tau_{\mathcal{G}_i}(T)] + \sum_{i=1}^r p_i \mathtt{I}^\tau_{\mathcal{G}_i}(T)^2 \\
&\underbrace{\leq}_{\text{Theorem 2}} \sum_{i=1}^r p_i \tau \mathtt{M}^{\tau-1}_{\mathcal{G}_i}(\bar{T})\mathtt{I}^\tau_{\mathcal{G}_i}(T) + \sum_{i=1}^r p_i \mathtt{I}^\tau_{\mathcal{G}_i}(T)^2 \\
&\underbrace{\leq}_{(37),\ (38)} \frac{\tau}{\min_i p_i}\mathtt{I}^\tau_{\mathcal{G}}(T)\sum_{i=1}^r p_i \mathtt{M}^{\tau-1}_{\mathcal{G}_i}(\bar{T}) + \frac{1}{\min_i p_i}\mathtt{I}^\tau_{\mathcal{G}}(T)\sum_{i=1}^r p_i \mathtt{I}^\tau_{\mathcal{G}_i}(T) \\
&\underbrace{\leq}_{(35)\ ,\ (36)} \frac{1}{\min_i p_i}\mathtt{I}^\tau_{\mathcal{G}}(T)\left(\tau \mathtt{M}^{\tau-1}_{\mathcal{G}}(\bar{T}) + \mathtt{I}^\tau_{\mathcal{G}}(T)\right) \leq \frac{\tau+1}{\min_i p_i}\mathtt{I}^\tau_{\mathcal{G}}(T)\max_{v\in V}\mathtt{I}^\tau_{\mathcal{G}}(v)
\end{aligned}$$

◀

We next give a different proof (of a slightly different bound) for live-edge models using a reduction to an IC model. Consider a set of τ-steps models $\mathcal{G}_i(V, \mathcal{E}_i)$ for $i \in [r]$ and respective probabilities p_i such that $\sum_{i=1}^r p_i = 1$. We define a mixture model $\mathcal{G}(V, \bigcup_i \mathcal{E}_i)$ as follows. To draw $E \sim \mathcal{G}$, we first draw $i \in [r]$ according to probabilities p_i and then return $E \sim \mathcal{G}_i$.

▶ **Theorem 35.** *Consider a model $\mathcal{G}$ that is a mixture of r IC models $\mathcal{G}_i$ with probabilities p_i. Then for all $T \subset V$,*

$$\mathit{Var}[R^\tau_{\mathcal{G}}(T)] \leq \frac{\tau+1}{\min_i p_i} I^\tau_{\mathcal{G}}(T)\max\{I^\tau_{\mathcal{G}}(T), \max_{v\in V} I^\tau_{\mathcal{G}}(v)\}$$

Proof. We first argue that we can assume without loss of generality that T is a single node and $\bigcup_i \mathcal{E}_i$ does not contain edges that are incoming to T. We can transform a general case $\mathcal{G}$ and T to this form by contracting all nodes in T into a single node and deleting all edges that are incoming to T. We then retain the same conditional distribution on the remaining edges. Note that this transformation preserves the distribution of $\mathtt{R}^\tau_{\mathcal{G}}(T)$ and hence also its expectation and variance. The influence values $I^\tau_{\mathcal{G}}(v)$ of nodes $v \in V \setminus T$ can only decrease. Finally, the transformed model is also a mixture of correspondingly transformed IC models, where in each such model the distribution of $\mathtt{R}^\tau_{\mathcal{G}_i}(T)$ remains the same and influence values $I^\tau_{\mathcal{G}_i}(v)$ can only decrease. It follows that the claimed variance bound for the transformed model implies the same bound for the original model.

We construct a new IC model $\mathcal{G}'$ with respect to (a single node) T as follows. The new model has nodes $V' = \{v\} \cup \bigcup_i V_i$, where each V_i is a map of V. We create an instantiation of each of our IC models $\mathcal{G}_i$ with set of nodes V_i and edges $\mathcal{E}_i$ with the probabilities as in the model $\mathcal{G}_i$. The new IC model $\mathcal{G}'$ has a root node v with weight 0 and for each $i \in [r]$, there is an edge (v, T_i) with probability p_i, where T_i is the image of T in the copy of $\mathcal{G}_i$. We can see that

$$I^{\tau+1}_{\mathcal{G}'}(v) = \mathtt{I}^{\tau}_{\mathcal{G}}(T) = \sum_{i=1}^{r} p_i \mathtt{I}^{\tau}_{\mathcal{G}_i}(T) \ , \tag{40}$$

that is, the $\tau + 1$ steps influence of v in the constructed IC model $\mathcal{G}'$ is equal to the τ steps influence of T in the mixture model $\mathcal{G}$.

We next consider the variance of the random variables $\mathtt{R}^{\tau}_{\mathcal{G}}(T)$ and $\mathtt{R}^{\tau+1}_{\mathcal{G}'}(v)$. Both these random variables are a sum of r products of random variables:

$$\sum_{i=1}^{r} X_i \mathtt{R}^{\tau}_{\mathcal{G}_i}(T) \ ,$$

where X_i are Bernoulli with probabilities p_i. In both cases the random variables $\{\mathtt{R}^{\tau}_{\mathcal{G}_i}(T)\}$ are independent of each other and also are independent from (the joint distribution of) $\{X_i\}$. But in the case of $\mathtt{R}^{\tau+1}_{\mathcal{G}'}(v)$ the random variables X_i are independent and hence also the products are independent and in the case of $\mathtt{R}^{\tau}_{\mathcal{G}}(T)$, the variables $\{X_i\}$ have negative dependence as $\sum_i X_i = 1$ and thus the products $X_i \mathtt{R}^{\tau}_{\mathcal{G}_i}(T)$ are also negatively dependent. Therefore,

$$\mathsf{Var}[\mathtt{R}^{\tau}_{\mathcal{G}}(T)] \leq \mathsf{Var}[\mathtt{R}^{\tau+1}_{\mathcal{G}'}(v)] = \sum_i \mathsf{Var}[X_i \mathtt{R}^{\tau}_{\mathcal{G}_i}(T)] \ . \tag{41}$$

Finally, we bound $\mathtt{M}^{\tau}_{\mathcal{G}'}(\bar{v})$ by considering the maximum influence of a node other than v in the constructed model $\mathcal{G}'$. For T_i we have

$$\mathtt{I}^{\tau}_{\mathcal{G}'}(T_i) = \mathtt{I}^{\tau}_{\mathcal{G}_i}(T) \leq \frac{1}{p_i} \mathtt{I}^{\tau}_{\mathcal{G}}(T) \ , \tag{42}$$

where the last inequality follows from (40). We next consider nodes $z_i \in V_i$ that is a map of a node $z \in V$.

$$\mathtt{I}^{\tau}_{\mathcal{G}'}(z_i) = \mathtt{I}^{\tau}_{\mathcal{G}_i}(z) \leq \frac{1}{p_i} \mathtt{I}^{\tau}_{\mathcal{G}}(z) \ . \tag{43}$$

The last inequality follows because for any node $z \in v$ we have $\mathtt{I}^{\tau}_{\mathcal{G}}(z) = \sum_{i=1}^{r} p_i I^{\tau}_{\mathcal{G}_i}(z)$. Combining (42) and (43) we get

$$\begin{aligned} \mathtt{M}^{\tau}_{\mathcal{G}'}(\bar{v}) &= \max_{u \in V' \setminus \{v\}} \mathtt{I}^{\tau}_{\mathcal{G}'}(u) \leq \max_{u \in V} \max_i \mathtt{I}^{\tau}_{\mathcal{G}_i}(u) \leq \max_{u \in V} \frac{1}{\min_i p_i} \mathtt{I}^{\tau}_{\mathcal{G}}(u) \\ &= \frac{1}{\min_i p_i} \max\{I^{\tau}_{\mathcal{G}}(T), \max_{z \in V} \mathtt{I}^{\tau}_{\mathcal{G}}(z)\} \ . \end{aligned} \tag{44}$$

To conclude, we invoke Theorem 2 for the IC model $\mathcal{G}'$:

$$\mathsf{Var}[\mathtt{R}^{\tau}_{\mathcal{G}}(T)] \underbrace{\leq}_{(41)} \mathsf{Var}[\mathtt{R}^{\tau+1}_{\mathcal{G}'}(v)] \underbrace{\leq}_{\text{Theorem 2}} (\tau+1) I^{\tau+1}_{\mathcal{G}'}(v) \mathtt{M}^{\tau}_{\mathcal{G}'}(\bar{v}) \ .$$

We then apply inequalities (44) and the equality (40) to obtain the claim. ◀

On Oblivious Amplification of Coin-Tossing Protocols

Nir Bitansky[1]
Tel Aviv University, Israel
nirbitan@tau.ac.il

Nathan Geier
Tel Aviv University, Israel
nathangeier@mail.tau.ac.il

Abstract

We consider the problem of amplifying two-party coin-tossing protocols: given a protocol where it is possible to bias the common output by at most ρ, we aim to obtain a new protocol where the output can be biased by at most $\rho^\star < \rho$. We rule out the existence of a natural type of amplifiers called *oblivious amplifiers* for every $\rho^\star < \rho$. Such amplifiers ignore the way that the underlying ρ-bias protocol works and can only invoke an oracle that provides ρ-bias bits.

We provide two proofs of this impossibility. The first is by a reduction to the impossibility of deterministic randomness extraction from Santha-Vazirani sources. The second is a direct proof that is more general and also rules outs certain types of asymmetric amplification. In addition, it gives yet another proof for the Santha-Vazirani impossibility.

2012 ACM Subject Classification Theory of computation → Cryptographic protocols; Theory of computation → Oracles and decision trees

Keywords and phrases Coin Tossing, Amplification, Lower Bound, Santha Vazirani

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.30

Funding This work was supported in part by ISF grant 18/484, and by Len Blavatnik and the Blavatnik Family Foundation.
Nir Bitansky: Supported by the Alon Young Faculty Fellowship.

Acknowledgements We thank Itay Sason for helpful discussions.

1 Introduction

Hardness amplification is a foundational problem in cryptography: given a cryptographic primitive that is *weakly secure* in some sense, we would like to make it *strongly secure*. Two famous examples of such amplification procedures are Yao's amplification of weak one-way functions into strong ones and Yao's Xor lemma for amplifying unpredictability [17]. Other examples of primitives with known amplification procedures include public-key encryption [5, 11], oblivious transfer [4, 16], commitments [4, 8] and cryptographic arguments [6].

Coin-Tossing Protocols

We consider amplification of coin-tossing protocols [3]. Such protocols allow two parties to jointly toss an unbiased coin, such that even if one party is malicious and diverges from the protocol, it cannot significantly bias the outcome. Concretely, a *ρ-bias coin-tossing protocol* is such that a cheating party cannot force the common outcome to be any specific bit b with probability greater than $1/2 + \rho$.

[1] Member of the Check Point Institute of Information Security.

It is common knowledge that coin-tossing protocols against unbounded (or even PSPACE) adversaries cannot have bias $\rho < 1/2$. Accordingly, the common definition addresses efficient adversaries. Here coin-tossing protocols with negligible bias have been long known assuming one-way functions [3, 9, 13]. In fact, a long line of work shows that one-way functions are, in fact, necessary provided that the bias is at most $\rho < 1/2 - \Omega(1)$ [10, 12, 7, 2]. Whether one-way functions are necessary for any non-trivial bias $\rho < 1/2 - 1/\text{poly}(n)$ (or even for some $\rho = 1/2 - o(1)$) remains an open problem.

Amplifying Coin Tossing

A coin-tossing amplifier should take a coin-tossing protocol π with (non-trivial) bias $\rho < 1/2$ and transform it into a new coin-tossing protocol $\pi^\star$ with smaller bias $\rho^\star < \rho$. One specific way for obtaining coin-tossing amplifiers is to first derive from the protocol π a one-way function f_π, and then construct optimal coin tossing from f_π. Indeed, following the known results mentioned above this would work for any

$$\text{negl}(n) < \rho^\star < \rho < 1/2 - \Omega(1) \ .$$

In this work, we ask whether there exist "more direct" amplification procedures, which we call *oblivious amplifiers*. Such amplifiers completely ignore the way that the underlying protocol π works, they only obtain oracle access to the result – the adversary can adaptively bias the resulting bit of each oracle invocation as long as the bias is bounded by ρ. The amplifier is required to satisfy an information theoretic guarantee: *unbounded* attackers cannot bias the common output of the new protocol π^* by more than $\rho^* < \rho$. Addressing unbounded attackers is a natural choice as we wish to avoid computational assumptions (certainly, one-way functions, which would trivialize the problem).

We find the model of oblivious amplifiers quite natural and similar to other settings, such as Yao's Xor lemma [17], where amplifiers satisfy an information theoretic guarantee. In particular, constructing such amplifiers seems like a natural route toward fully understanding the complexity of coin tossing. In particular, the existence of such efficient amplifiers for

$$\rho^\star \leq 1/2 - \Omega(1) < 1/2 - 1/\text{poly}(n) \leq \rho$$

would completely resolve the question, showing that any non-trivial coin tossing is equivalent to one-way functions.

1.1 Results

We show that oblivious coin-tossing amplifiers do not exist.

▶ **Theorem 1** (Informal). *There do not exist oblivious coin-tossing amplifiers for any $\rho^* < \rho$.*

Our theorem can further be extended to rule out oblivious amplification of *weak coin tossing* [2] that essentially requires that one side cannot bias toward 0 and the other cannot bias toward 1. (See Remark 6.)

A More General Lower Bound

We give two proofs of the above theorem. The first is by a simple reduction to the impossibility of *deterministic randomness extraction from Santha-Vazirani sources* [15]. We also give a direct proof that provides a more general lower bound. In particular, we quantify the tradeoff between improving the potential bias caused by specific party A toward a specific bit b, at

the account of making it worst for other party B and bit $1-b$. This also rules out oblivious amplification for asymmetric notions of coin tossing where we allow different bounds on the bias for the two parties. The tradeoff is explained in the technical overview below and expressed in Figure 2.

The alternative proof also gives yet another proof of the Santha-Vazirani impossibility for deterministic extraction (in addition to several existing proofs [15, 14, 1]).

1.2 Technical Overview

We now give a brief overview of our proofs.

Modeling Protocols as Trees

We model any possible amplifier protocol $\pi^\star$ as a full binary tree. The correspondence is natural: the root corresponds to the beginning of the protocol before any message is sent. Every inner node in the tree is either

- Controlled by one of the two parties A or B, meaning that it is this party's turn to send a message, without loss of generality, a single bit.
- Representing an oracle call to the underlying protocol π resulting in a common bit.

Whenever a message is sent or an oracle call is made, we move in accordance to the left or right child, until reaching a leaf labeled by the common outcome of the protocol. Here the execution ends.

Each node is associated with some (honest) distribution on the next bit to be sent or produced by the oracle. The adversary can gain control over the nodes representing one of the parties and arbitrarily fix their distributions. For any oracle node, the adversary can fix an arbitrary distribution provided that it has bias at most ρ. Note that every node corresponds to some partial execution of the protocol, represented by the path from the root to this node, and the adversarial response is adaptively fixed according to this path. (See illustration in Figure 1.)

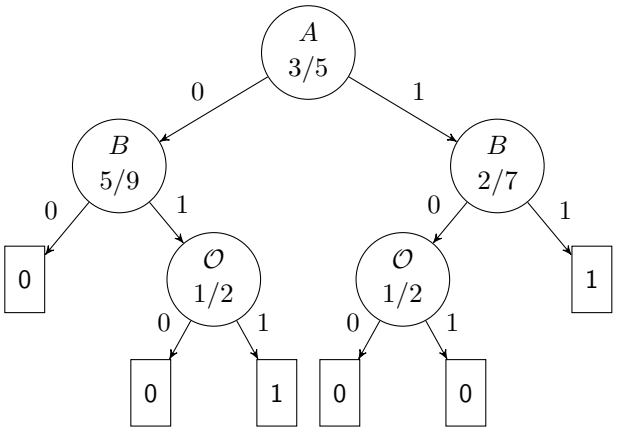


Figure 1 An example of a protocol represented by a tree. Each node specifies which party should send a bit as well as the distribution of this bit (when the party is honest). Alternatively, the parties may invoke the oracle and obtain a joint bit. The adversary may take control of one of the parties and arbitrarily replace its distributions; it can also replace the oracle call distributions provided that they remain ρ-biased.

The above model does not explicitly capture protocols where honest parties may store private coins through the interaction. However, we can transform any private-coin protocol to a protocol in the above model by considering parties that do not explicitly keep private randomness; instead, in every node they resample their private randomness conditioned on the execution transcript so far and answer according to that string. The transformation has no effect on the distribution of transcript and in particular on the outcome bit. However, the transformation may make the new protocol inefficient. This is not an issue as our lower bound will in fact hold for inefficient protocols as well.

A Reduction to Deterministic Santha-Vazirani Extractors

In their seminal work, Santha and Vazirani considered the problem of extracting a statistically uniform bit from a sequence of biased random bits $\mathbf{X} = X_1, \ldots, X_n$ such that X_i has bounded bias ρ conditioned on any fixing of $X_1, \ldots, X_{i-1}$. Such random sources $\mathbf{X}$ are called Santha-Vazirani Sources. They proved that there exist no deterministic extractors for such sources, namely, for any deterministic Boolean function E, there exists a Santha-Vazirani source such that $E(\mathbf{X})$ is far from uniform. In fact, they proved that $E(\mathbf{X})$ is ρ-biased.

Indeed, this bares similarity to the setting of oblivious coin-tossing amplification where the ρ-biased input bits naturally correspond to oracle calls to π. The gap between the two models is that a coin-tossing oblivious amplifier further allows interaction between two *randomized* parties. To bridge this gap, we prove that any oblivious amplifier can be turned into a non-interactive oblivious amplifier.

To convey the basic idea, imagine that we have an amplifier tree where the root belongs to party A (namely, it sends the first message). We argue that at least one of its left or right subtrees also has bias at most $\rho^\star$, and thus we can reduce one round of interaction. In fact, *both* subtrees must be such that an adversary controlling A can bias the outcome by at most $\rho^\star$, or else an adversary controlling A in the original protocol could have simply always chosen the worse subtree in order to bias the outcome by more than $\rho^\star$. We then observe that at least one of the subtrees must be such that an adversary controlling B cannot bias by more than $\rho^\star$. Indeed, the bias caused by an adversary controlling B in the original protocol is just a convex combination of the its bias in the two corresponding subtrees.

The Second Proof and a More General Lower Bound

Our second proof of the theorem pours more light on the tradeoff between how much each party can bias the protocol towards a specific bit. We first give a correspondence between protocols and points on the plane. For a protocol π, let x_π be the maximum probability a malicious A^* (controlling A) can force output 0 and y_π be the maximum probability a malicious B^* can force output 1. We assign the protocol π coordinates (x_π, y_π). Considering the tree representing the protocol π, the point in the plane corresponding to the protocol π is determined by the points corresponding to its children themselves as protocols, as well as the operation at its root. (This is roughly done following a similar argument to the one used above, where we either take the maximum of two subtrees, or a convex combination thereof.)

Our basic Theorem 1 and its proof show that all points corresponding to protocols lie outside the axis-aligned square given by $x = 0$, $y = 0$, $x = 0.5 + \rho$, $y = 0.5 + \rho$. Our more general analysis, shows a more accurate picture – all points lie above a certain piecewise linear f (See Figure 2). In a bit more detail, we prove by induction on the depth of the tree that the point corresponding to every protocol lies above the function f, by showing closure of points above f to each of the operations that may appear at the root, and also

that protocol leaves (the base case) lie above f. When considering a non-leaf protocol, we get from the induction that the sub-protocols rooted at its children lie above f, and since the set of points above f is closed under the operation at the protocols' root, the point corresponding to the protocol itself also lies above f.

This more general result shows that we cannot hope to improve the x_π-value of the oracle without strictly hurting its y_π-value, and furthermore gives some lower bound $g(\Delta_x)$, for every $0 \leq \Delta_x \leq x_\pi$, on how much y_π must increase in order to reduce x_π by Δ_x.

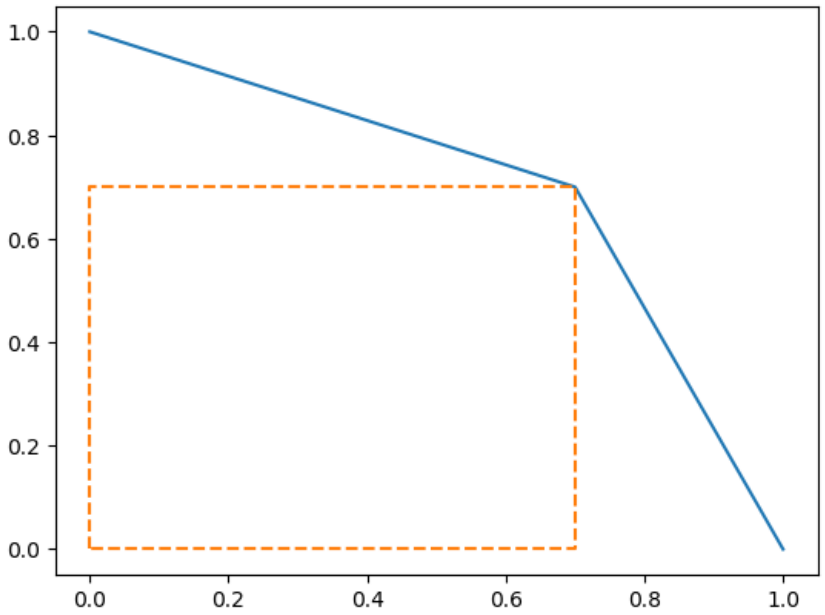


Figure 2 For the sake of this illustration, we set $\rho = 0.2$. The X-axis represents the ability of a malicious A^* to bias the common output toward 0, and the Y-axis represents the ability of a malicious B^* to bias the common output toward 1. While our basic Theorem 1 argues that all protocols lie outside the dotted square, our more general theorem shows that they in fact lie above the piece-wise linear function represented by the continuous line.

Organization

In Section 2, we define the relevant notion of oblivious amplifiers and their correspondence to trees. In Section 3, we prove the main impossibility result by a reduction to the impossibility of deterministic Santha-Vazirani extraction. In Section 4, we give our alternative (direct) proof leading to a more general lower bound.

2 Definitions

Throughout the paper, we denote by $Ber(p)$ the Bernoulli distribution with parameter p, namely, the value 1 gets probability p and 0 gets $1 - p$.

▶ **Definition 2** (Common-output two-party protocol with oracle access to ρ-biased bits)**.** *We model a common-output two-party protocol with oracle access to ρ-biased bits using full binary trees such that:*

- *Every leaf is labeled with either "0" or "1", which represent the common output of the protocol upon reaching that leaf.*
- *Every inner node v is labeled with "(P_v, q_v)" where*
  1. *$P_v \in \{A, B, \mathcal{O}\}$ represents whether it is A's turn to speak, B's turn to speak, or an oracle call.*
  2. *In an honest execution, the bit sent after reaching that node should be distributed according to $Ber(q_v)$. If $P_v = \mathcal{O}$ then q_v must be 0.5, otherwise q_v may be an arbitrary probability.*

Execution. *An execution of the protocol follows a path along the tree, starting at the root and going left/right in accordance to the bits sent, when 0 means left and 1 means right, until reaching a leaf which determines the common output of that execution.*

Adversarial behavior. *A malicious party can ignore q_v when it is their turn to speak and send an arbitrarily distributed bit instead. Also, whenever the oracle is called, the malicious party may change the output distribution of that call from $Ber(0.5)$ to $Ber(q_v)$ for any q_v such that $|q_v - 0.5| \leq \rho$. Note that q_v may depend on all the bits sent up to that call*

▶ **Definition 3.** *Given a protocol π, A^* an adversarial behavior of A and $b \in \{0,1\}$, we denote by $\pi(A^*, b)$ the probability of π's output being b when executed with A^* and an honest B. Next, define $Opt(\pi, A^*, b) := \max_{A^*} \pi(A^*, b)$, where the maximum is taken over all adversarial behaviors A^* of A. $\pi(B^*, b)$ and $Opt(\pi, B^*, b)$ are defined symmetrically for B.*

▶ **Definition 4** ((α, β)-oblivious coin-tossing amplifier)**.** *A common-output two-party protocol π with oracle access to α-biased bits is an (α, β)-oblivious coin-tossing amplifier if*

$$\max_{P^*, b} Opt(\pi, P^*, b) \leq 0.5 + \beta \ ,$$

where $P^* \in \{A^*, B^*\}$ and $b \in \{0, 1\}$. In addition, we require that the expected output of π over an honest execution is 0.5.

▶ Remark 5. Actually, for the lower bound, we do not use the fact that the expected output of π over an honest execution is 0.5. We note that bounded biases already makes coin-tossing protocols non-trivial.

▶ Remark 6. We may also consider the notion of (α, β)-oblivious weak coin-tossing amplifiers, which aim to produce a β-bias weak coin tossing (CT) protocol instead of standard β-CT. In the weak CT setting, we know in advance that A^* aims to bias the result toward 0 while B^* aims to bias the result toward 1. We only need to bound their ability to bias the output in their chosen direction. The difference is in the requirement

$$\max_{P^*, b} Opt(\pi, P^*, b) \leq 0.5 + \beta \ ,$$

which is replaced with

$$\max\{Opt(\pi, A^*, 0), Opt(\pi, B^*, 1)\} \leq 0.5 + \beta \ .$$

3 Impossibility of Oblivious Amplifiers

In this section, we state and prove our main result regarding the impossibility of non-trivial oblivious coin-tossing amplifiers.

▶ **Theorem 7** (main). *For every common-output two-party protocol π with oracle access to ρ-biased bits, either $Opt(\pi, A^*, 0) \geq 0.5 + \rho$ or $Opt(\pi, B^*, 1) \geq 0.5 + \rho$. In particular, there do not exist (α, β)-oblivious coin-tossing amplifier, for any $\beta < \alpha$.*

Toward proving the theorem, we first analyze in Section 3.1 the best possible (unbounded) attacks. Then in Section 3.2, we give the reduction to the impossibility of deterministic extraction from Santha-Vazirani sources.

3.1 The Optimal Attacks

Let π be a common-output two-party protocol with oracle access to ρ-biased bits, and denote its root by r. Also, denote by π_0 and π_1 the protocols rooted at the left and right children of r. The best strategy of a malicious A^* to make the protocol π output 0 is to bias the first bit b as much as possible towards $\arg\max_{z\in\{0,1\}} Opt(\pi_z, A^*, 0)$, and after b is sent continue to recursively apply the best strategy at the resulting child. The best strategy of a malicious B^* to make the protocol π output 1 is symmetrical. Thus, we have that:

- If r is labeled with "A, p", then A can completely bias the bit while B cannot change its distribution at all, so

$$Opt(\pi, A^*, 0) = \max_{z\in\{0,1\}} Opt(\pi_z, A^*, 0)$$
$$Opt(\pi, B^*, 1) = \mathbb{E}_{z\sim Ber(p)}\left[Opt(\pi_z, B^*, 1)\right]$$

- If r is labeled with "B, p", then B can completely bias the bit while A cannot change its distribution at all, so

$$Opt(\pi, A^*, 0) = \mathbb{E}_{z\sim Ber(p)}\left[Opt(\pi_z, A^*, 0)\right]$$
$$Opt(\pi, B^*, 1) = \max_{z\in\{0,1\}} Opt(\pi_z, B^*, 1)$$

- If r is labeled with "$\mathcal{O}$", then both A and B can bias the bit by at most ρ, so

$$Opt(\pi, A^*, 0) = (0.5-\rho)\cdot \min_{z\in\{0,1\}} Opt_z(\pi, A^*, 0) + (0.5+\rho)\cdot \max_{z\in\{0,1\}} Opt(\pi_z, A^*, 0)$$
$$Opt(\pi, B^*, 1) = (0.5-\rho)\cdot \min_{z\in\{0,1\}} Opt(\pi_z, B^*, 1) + (0.5+\rho)\cdot \max_{z\in\{0,1\}} Opt(\pi_z, B^*, 1)$$

- If r is a leaf labeled with "0", then

$$Opt(\pi, A^*, 0) = 1$$
$$Opt(\pi, B^*, 1) = 0$$

- If r is a leaf labeled with "1", then

$$Opt(\pi, A^*, 0) = 0$$
$$Opt(\pi, B^*, 1) = 1$$

3.2 Reduction to Deterministic Santha-Vazirani Extraction

The following theorem states that allowing interaction does not help in creating more secure protocols, in the sense that it does not allow us to generate a better protocol, in either $Opt(\pi, A^*, 0)$ or $Opt(\pi, B^*, 1)$, than protocols already existing without interaction.

▶ **Theorem 8.** *For every common-output two-party protocol π with oracle access to ρ-biased bits, there exists a common-output two-party protocol π' with oracle access to ρ-biased bits, with no inner nodes labeled by either "A, p" or "B, p", and such that $Opt(\pi', A^*, 0) \leq Opt(\pi, A^*, 0)$ and $Opt(\pi', B^*, 1) \leq Opt(\pi, B^*, 1)$*

In other words, allowing interaction does not help in producing protocols where it is harder for A^* to cheat towards 0 or for B^* to cheat towards 1.

Proof. We show how given any protocol π with "A, p"/"B, p" turns, we can strictly reduce its number of "A, p"/"B, p" turns without increasing its cheating probabilities $Opt(\pi, A^*, 0)$ and $Opt(\pi, B^*, 1)$. In essence, we show that for every inner node labeled with "A, p"/"B, p" there exists a child such that replacing the subtree rooted at that node with the subtree rooted at its child results a protocol that is only harder to cheat in, for both A^* towards 0 and B^* towards 1. In more detail: First, we pick some arbitrary inner node of π labeled with either "A, p" or "B, p". Denote this node by N, and its left and right children by N_0 and N_1, respectively. If N is labeled by "A, p" we choose $z := \arg\min_{b\in\{0,1\}} Opt(N_b, B^*, 1)$, otherwise choose $z := \arg\min_{b\in\{0,1\}} Opt(N_b, A^*, 0)$. We take π and replace in it the subtree rooted at N with the subtree rooted at N_z. Essentially, we fixed the bit sent at node N to always be z, instead of letting A/B choose it. We make two observations:

1. Both $Opt(N_z, A^*, 0) \leq Opt(N, A^*, 0)$ and $Opt(N_z, B^*, 1) \leq Opt(N, B^*, 1)$.
   To see this, assume w.l.o.g that N is labeled with "A, p" (symmetrical argument for "B, p"). We have that $Opt(N, A^*, 0) = \max_{b\in\{0,1\}} Opt(N_b, A^*, 0)$ and in particular $Opt(N, A^*, 0) \geq Opt(N_z, A^*, 0)$. Also, since we have that
   $$Opt(N, B^*, 1) = \mathbb{E}_{b\sim Ber(p)} [Opt(N_b, B^*, 1)]$$
   and we chose $z := \arg\min_{b\in\{0,1\}} Opt(N_b, B^*, 1)$ in case N is labeled with "A, p", then $Opt(N, B^*, 1) \geq Opt(N_z, B^*, 1)$.
   In other words, at N_z it is both harder for A^* to cheat towards 0 and for B^* to cheat towards 1.
2. Let π and T be protocols, N be some node of π, and π' be the protocol resulted by replacing in π the subtree rooted at N with T. If $Opt(T, A^*, 0) \leq Opt(N, A^*, 0)$ then $Opt(\pi', A^*, 0) \leq Opt(\pi, A^*, 0)$ and similarly if $Opt(T, B^*, 1) \leq Opt(N, B^*, 1)$ then $Opt(\pi', B^*, 1) \leq Opt(\pi, B^*, 1)$. The reason for this is that the value $Opt(R, A^*, 0)$ of an inner node R, for all three operations ("A, p","B, p","$\mathcal{O}$"), is a monotone function of the values $Opt(R_0, A^*, 0), Opt(R_1, A^*, 0)$ of its children. By not increasing the $(A^*, 0)$-value of some node, you cannot increase the $(A^*, 0)$-value of its father, which in turn will not increase the $(A^*, 0)$-value of its father and so on.
   In other words, by taking a protocol and replacing the sub-protocol run at some partial transcript with another sub-protocol where it is not easier for A^*/B^* to cheat towards 0/1, we result a protocol where it is not easier for A^*/B^* to cheat towards 0/1.

Combining these observations, we reach the conclusion that by taking a protocol and repeatedly getting rid of its "A, p"/"B, p" nodes by replacing the subtree rooted at them with the subtree rooted at a correctly chosen child of theirs, we can get rid of all the "A, p"/"B, p" turns without increasing the cheating probabilities of A^* towards 0 and of B^* towards 1. ◀

▶ **Definition 9** (SV Source)**.** *For $0 \leq \rho \leq 0.5$, a source $X = X_1, \ldots, X_n$ of length n (A random variable taking values in $\{0,1\}^n$) is a Santha-Vazirani (SV) source with bias ρ if for every $i \in [n]$ and every $x_1, \ldots, x_{i-1} \in \{0,1\}$, the bias of X_i conditioned on $X_1 = x_1, \ldots, X_{i-1} = x_{i-1}$ is at most ρ. That is,*

$$|\mathbb{E}[X_i | X_1 = x_1, \ldots, X_{i-1} = x_{i-1}] - 0.5| \leq \rho \ .$$

▶ **Theorem 10** ([15], [14]). *For every function $f : \{0,1\}^n \to \{0,1\}$ and every $0 \leq \rho \leq 0.5$ there exists an SV source $X = X_1, \ldots, X_n$ of length n and bias ρ such that $f(X)$ has bias at least ρ.*

We remark that this version of the statement is taken from Reingold, Vadhan and Wigderson [14], and that differently from them we refer to the bias as the distance from 0.5 and not double that amount.

The following theorem implies that there are no non-trivial oblivious coin-tossing amplifiers comprised solely of oracle calls.

▶ **Theorem 11.** *For every common-output two-party protocol π with oracle access to ρ-biased bits and no inner nodes labeled by either "A, p" or "B, p", either $Opt(\pi, A^*, 0) \geq 0.5 + \rho$ or $Opt(\pi, B^*, 1) \geq 0.5 + \rho$*

In fact, in a protocol with no "A, p"/"B, p" turns there is no longer a difference between A and B so either $Opt(\pi, A^*, 0) = Opt(\pi, B^*, 0) \geq 0.5 + \rho$ or $Opt(\pi, A^*, 1) = Opt(\pi, B^*, 1) \geq 0.5 + \rho$

Proof. Let n be an upper bound on the number of oracle calls made by π in any execution. We can think w.l.o.g of π as always making exactly n oracle calls, with results denoted by $o_1, \ldots, o_n$, and then outputting $f(o_1, \ldots, o_n)$ for some function f. The impossibility result of deterministic extraction from Santha-Vazirani sources, Theorem 10, guarantees that there exists an SV source $X = X_1, \ldots, X_n$ of length n and bias ρ such that $f(X)$ has bias at least ρ. Therefore, if the malicious party can make sure the oracle calls' output distribution is X, they can succeed in biasing the output of π by at least ρ. (If it is towards 0 then $Opt(\pi, A^*, 0) \geq 0.5 + \rho$ and if it is towards 1 then $Opt(\pi, B^*, 1) \geq 0.5 + \rho$). The malicious party can make sure the oracle calls' output distribution is X by doing it bit-by-bit: After $0 \leq i \leq n - 1$ calls with results $x_1, \ldots, x_i$ they ask the next oracle call to be distributed according to $X_{i+1} | X_1 = x_1, \ldots, X_i = x_i$, and their request from the ρ-biased bit oracle is legal by the definition of SV sources with bias ρ. ◀

Combining Theorem 8 and Theorem 11, Theorem 7 follows.

4 A More General Lower Bound

In this section, we give a direct proof by induction which gives some insight on the possible relations between $Opt(\pi, A^*, 0)$ and $Opt(\pi, B^*, 1)$ and the trade-offs we can get between them.

We correspond every protocol with a point on the plane using $Opt(\pi, A^*, 0)$ as the x-coordinate and $Opt(\pi, B^*, 1)$ as the y-coordinate. Our main theorem translates into showing that all protocol points lie outside or on the axis-aligned square with boundaries $x = 0.5 + \rho, x = 0$ and $y = 0.5 + \rho, y = 0$, but still inside of the first (i.e., $(+,+)$) quadrant (the second part is obvious from the definition). The area outside the square (but still inside the first quadrant) is not closed under the oracle-call operation (defined later), and thus the naïve induction attempt fails. Instead, we will strengthen the induction hypothesis into showing that all points lie above $f := \min\left(1 - \frac{0.5-\rho}{0.5+\rho} \cdot x, \frac{0.5+\rho}{0.5-\rho} - \frac{0.5+\rho}{0.5-\rho} \cdot x\right)$, which is both closed under all operations (now the induction works) and lies above the square (so it implies what we want, and more). See Figure 2 for a graphic representation of the square and f.

▶ **Theorem 12.** *For every common-output two-party protocol π with oracle access to ρ-biased bits, we have that* $Opt(\pi, B^*, 1) \geq \min(1 - \frac{0.5-\rho}{0.5+\rho} \cdot Opt(\pi, A^*, 0),\ \frac{0.5+\rho}{0.5-\rho} - \frac{0.5+\rho}{0.5-\rho} \cdot Opt(\pi, A^*, 0))$

Equivalently, if we take ℓ_1 to be the line between $(0, 1)$ and $(0.5 + \rho, 0.5 + \rho)$

$$\text{so } \ell_1 \text{ is } y = 1 - \frac{0.5 - \rho}{0.5 + \rho} \cdot x$$

and ℓ_2 to be the line between $(0.5 + \rho, 0.5 + \rho)$ and $(1, 0)$

$$\text{so } \ell_2 \text{ is } y = \frac{0.5 + \rho}{0.5 - \rho} - \frac{0.5 + \rho}{0.5 - \rho} \cdot x$$

then the point $(Opt(\pi, A^*, 0), Opt(\pi, B^*, 1))$ lies above (not strictly) at least one of ℓ_1, ℓ_2.

▶ **Corollary 13.** *Theorem 7 follows.*

Proof. Since both ℓ_1 and ℓ_2 are strictly decreasing (negative slope) and pass through the point $(0.5 + \rho, 0.5 + \rho)$, then $x < 0.5+\rho$ implies $\ell_1(x), \ell_2(x) > 0.5+\rho$. Thus $Opt(\pi, A^*, 0) < 0.5+\rho$ implies $Opt(\pi, B^*, 1) \geq \min\left(\ell_1(Opt(\pi, A^*, 0)), \ell_2(Opt(\pi, A^*, 0))\right) > 0.5 + \rho$ ◀

Proof of Theorem 12. Proof by induction on the depth.
Leaves labeled with "0" lie on ℓ_2 and leaves labeled with "1" lie on ℓ_1.
If π is not a leaf, denote its root by r, by π_0 and π_1 the protocols rooted at the left and right children of r, and let $f := \min(\ell_1, \ell_2)$. Also, denote

$$\begin{aligned} x_0, y_0 &= Opt(\pi_0, A^*, 0), Opt(\pi_0, B^*, 1) \\ x_1, y_1 &= Opt(\pi_1, A^*, 0), Opt(\pi_1, B^*, 1) \\ x', y' &= Opt(\pi, A^*, 0), Opt(\pi, B^*, 1) \end{aligned}$$

We assume (induction hypothesis) that $y_0 \geq f(x_0)$, $y_1 \geq f(x_1)$ and want to show that $y' \geq f(x')$. There are three types of operations to consider:

- If r is labeled with "A, p"

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \max(x_0, x_1) \\ (1-p) \cdot y_0 + p \cdot y_1 \end{pmatrix}$$

- If r is labeled with "B, p"

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} (1-p) \cdot x_0 + p \cdot x_1 \\ \max(y_0, y_1) \end{pmatrix}$$

- If r is labeled with "$\mathcal{O}$"

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} (0.5-\rho) \cdot \min(x_0, x_1) + (0.5+\rho) \cdot \max(x_0, x_1) \\ (0.5-\rho) \cdot \min(y_0, y_1) + (0.5+\rho) \cdot \max(y_0, y_1) \end{pmatrix}$$

We may assume without loss of generality that the points $(x_0, y_0), (x_1, y_1)$ lie on f and not strictly above it, namely, that $y_0 = f(x_0)$, $y_1 = f(x_1)$. This is true because y' is non-decreasing in y_0, y_1 in all three operations, so if $y' \geq f(x')$ for $y_0 = f(x_0)$, $y_1 = f(x_1)$, then we also have that $y' \geq f(x')$ for any $y_0 \geq f(x_0)$, $y_1 \geq f(x_1)$.
(Note that $f(x')$ is unaffected by changes to y_0, y_1)

Let $(x_0, y_0), (x_1, y_1)$ be any two points on f. Since ℓ_1 and ℓ_2 are decreasing, f is also decreasing. Let $i = \arg\max_{z \in \{0,1\}} x_z$, so we have that $x_i > x_{1-i}$ and $y_i < y_{1-i}$.

- The first operation gives us $(x', y') = (x_i, (1-p)\cdot y_0 + p \cdot y_1)$, and since

$$y' = (1-p)\cdot y_0 + p \cdot y_1 > y_i = \min_{z\in\{0,1\}} y_z$$

for every $0 < p < 1$, we have $y' > y_i = f(x_i) = f(x')$.
- The second operation gives us $(x', y') = ((1-p)\cdot x_0 + p \cdot x_1, y_{1-i})$, and since

$$x' = (1-p)\cdot x_0 + p \cdot x_1 > x_{1-i} = \min_{z\in\{0,1\}} x_z$$

for every $0 < p < 1$, we have $f(x') < f(x_{1-i}) = y_{1-i} = y'$.
- The third operation gives us

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} (0.5-\rho)\cdot x_{1-i} + (0.5+\rho)\cdot x_i \\ (0.5-\rho)\cdot y_i + (0.5+\rho)\cdot y_{1-i} \end{pmatrix}$$

We separate into two cases:

1. Both (x_0, y_0) and (x_1, y_1) lie on either ℓ_1 or ℓ_2. Denote this common line by ℓ. Notice that every convex combination of $(x_0, y_0), (x_1, y_1)$ also lies on ℓ, and in particular

$$\begin{pmatrix} x_{0.5+\rho} \\ y_{0.5+\rho} \end{pmatrix} = (0.5-\rho)\cdot\begin{pmatrix} x_0 \\ y_0 \end{pmatrix} + (0.5+\rho)\begin{pmatrix} x_1 \\ y_1 \end{pmatrix} = \begin{pmatrix} (0.5-\rho)\cdot x_0 + (0.5+\rho)\cdot x_1 \\ (0.5-\rho)\cdot y_0 + (0.5+\rho)\cdot y_1 \end{pmatrix}$$

lies on ℓ. Since $x_{1-i} = \min(x_0, x_1)$ and $x_i = \max(x_0, x_1)$, we have that

$$x' = (0.5-\rho)\cdot x_{1-i} + (0.5+\rho)\cdot x_i \geq (0.5-\rho)\cdot x_0 + (0.5+\rho)\cdot x_1 = x_{0.5+\rho}$$

Since $y_i = \min(y_0, y_1)$ and $y_{1-i} = \max(y_0, y_1)$, we have that

$$y' = (0.5-\rho)\cdot y_i + (0.5+\rho)\cdot y_{1-i} \geq (0.5-\rho)\cdot y_0 + (0.5+\rho)\cdot y_1 = y_{0.5+\rho}$$

Overall, we conclude that $f(x') \leq \ell(x') \leq \ell(x_{0.5+\rho}) = y_{0.5+\rho} \leq y'$.
2. One of $(x_0, y_0), (x_1, y_1)$ lies on ℓ_1 and the other lies on ℓ_2. The lines ℓ_1, ℓ_2 intersect at $(0.5+\rho, 0.5+\rho)$ and the slope of ℓ_2 is steeper, so we can conclude that

$$f(x) = \min(\ell_1, \ell_2) = \begin{cases} \ell_1(x) & x \leq 0.5+\rho \\ \ell_2(x) & x > 0.5+\rho \end{cases}$$

and that $x_{1-i} < 0.5+\rho < x_i$. We have that

$$\begin{aligned} y' &= (0.5-\rho)\cdot y_i + (0.5+\rho)\cdot y_{1-i} = (0.5-\rho)\cdot \ell_2(x_i) + (0.5+\rho)\cdot \ell_1(x_{1-i}) = \\ &= (0.5-\rho)\cdot\left(\frac{0.5+\rho}{0.5-\rho} - \frac{0.5+\rho}{0.5-\rho}\cdot x_i\right) + (0.5+\rho)\cdot\left(1 - \frac{0.5-\rho}{0.5+\rho}\cdot x_{1-i}\right) = \\ &= 0.5+\rho-(0.5+\rho)\cdot x_i + 0.5+\rho-(0.5-\rho)\cdot x_{1-i} = \\ &= 1+2\rho-((0.5+\rho)\cdot x_i + (0.5-\rho)\cdot x_{1-i}) = 1+2\rho-x' \end{aligned}$$

Therefore, the point (x', y') lies on the line $y = 1+2\rho-x$ which we will denote by ℓ. Notice that ℓ also passes through $(0.5+\rho, 0.5+\rho)$, and that the slope of ℓ is steeper than that of ℓ_1 but more moderate than that of ℓ_2, and thus $\ell_1 \leq \ell$ for $x \leq 0.5+\rho$, and $\ell_2 \leq \ell$ for $x \geq 0.5+\rho$, and overall $f(x) \leq \ell(x)$. This implies that $f(x') \leq \ell(x') = y'$. ◀

References

1 Per Austrin, Kai-Min Chung, Mohammad Mahmoody, Rafael Pass, and Karn Seth. On the Impossibility of Cryptography with Tamperable Randomness. *Algorithmica*, 79(4):1052–1101, 2017. doi:10.1007/s00453-016-0219-7.

2 Itay Berman, Iftach Haitner, and Aris Tentes. Coin Flipping of *Any* Constant Bias Implies One-Way Functions. *J. ACM*, 65(3):14:1–14:95, 2018. doi:10.1145/2979676.

3 Manuel Blum. Coin Flipping by Telephone. In Allen Gersho, editor, *Advances in Cryptology: A Report on CRYPTO 81, CRYPTO 81, IEEE Workshop on Communications Security, Santa Barbara, California, USA, August 24-26, 1981.*, pages 11–15. U. C. Santa Barbara, Dept. of Elec. and Computer Eng., ECE Report No 82-04, 1981.

4 Ivan Damgård, Joe Kilian, and Louis Salvail. On the (Im)possibility of Basing Oblivious Transfer and Bit Commitment on Weakened Security Assumptions. In Jacques Stern, editor, *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding*, volume 1592 of *Lecture Notes in Computer Science*, pages 56–73. Springer, 1999. doi:10.1007/3-540-48910-X_5.

5 Cynthia Dwork, Moni Naor, and Omer Reingold. Immunizing Encryption Schemes from Decryption Errors. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, volume 3027 of *Lecture Notes in Computer Science*, pages 342–360. Springer, 2004. doi:10.1007/978-3-540-24676-3_21.

6 Iftach Haitner. A Parallel Repetition Theorem for Any Interactive Argument. *SIAM J. Comput.*, 42(6):2487–2501, 2013. doi:10.1137/100810630.

7 Iftach Haitner and Eran Omri. Coin Flipping with Constant Bias Implies One-Way Functions. *SIAM J. Comput.*, 43(2):389–409, 2014. doi:10.1137/120887631.

8 Shai Halevi and Tal Rabin. Degradation and Amplification of Computational Hardness. In Ran Canetti, editor, *Theory of Cryptography, Fifth Theory of Cryptography Conference, TCC 2008, New York, USA, March 19-21, 2008.*, volume 4948 of *Lecture Notes in Computer Science*, pages 626–643. Springer, 2008. doi:10.1007/978-3-540-78524-8_34.

9 Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A Pseudorandom Generator from any One-way Function. *SIAM J. Comput.*, 28(4):1364–1396, 1999. doi:10.1137/S0097539793244708.

10 Russell Impagliazzo and Michael Luby. One-way Functions are Essential for Complexity Based Cryptography (Extended Abstract). In *30th Annual Symposium on Foundations of Computer Science, Research Triangle Park, North Carolina, USA, 30 October - 1 November 1989*, pages 230–235. IEEE Computer Society, 1989. doi:10.1109/SFCS.1989.63483.

11 Huijia Lin and Stefano Tessaro. Amplification of Chosen-Ciphertext Security. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, volume 7881 of *Lecture Notes in Computer Science*, pages 503–519. Springer, 2013. doi:10.1007/978-3-642-38348-9_30.

12 Hemanta K. Maji, Manoj Prabhakaran, and Amit Sahai. On the Computational Complexity of Coin Flipping. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*, pages 613–622. IEEE Computer Society, 2010. doi:10.1109/FOCS.2010.64.

13 Moni Naor. Bit Commitment Using Pseudorandomness. *J. Cryptology*, 4(2):151–158, 1991. doi:10.1007/BF00196774.

14 Omer Reingold, Salil Vadhan, and Avi Wigderson. A note on extracting randomness from Santha-Vazirani sources. *Unpublished manuscript*, 2004.

15 Miklos Santha and Umesh V. Vazirani. Generating Quasi-random Sequences from Semi-random Sources. *J. Comput. Syst. Sci.*, 33(1):75–87, 1986. doi:10.1016/0022-0000(86)90044-9.

16 Jürg Wullschleger. *Oblivious-transfer amplification.* PhD thesis, Universität Zürich, 2008.

17 Andrew Chi-Chih Yao. Theory and Applications of Trapdoor Functions (Extended Abstract). In *23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, USA, 3-5 November 1982*, pages 80–91. IEEE Computer Society, 1982. doi:10.1109/SFCS.1982.45.

A New Analysis of Differential Privacy's Generalization Guarantees

Christopher Jung
University of Pennsylvania, Philadelphia, PA, USA
chrjung@seas.upenn.edu

Katrina Ligett
The Hebrew University, Jerusalem, Israel
katrina@cs.huji.ac.il

Seth Neel
University of Pennsylvania, Philadelphia, PA, USA
sethneel@wharton.upenn.edu

Aaron Roth
University of Pennsylvania, Philadelphia, PA, USA
aaroth@cis.upenn.edu

Saeed Sharifi-Malvajerdi
University of Pennsylvania, Philadelphia, PA, USA
saeedsh@wharton.upenn.edu

Moshe Shenfeld
The Hebrew University, Jerusalem, Israel
moshe.shenfeld@mail.huji.ac.il

Abstract

We give a new proof of the "transfer theorem" underlying adaptive data analysis: that any mechanism for answering adaptively chosen statistical queries that is differentially private and sample-accurate is also accurate out-of-sample. Our new proof is elementary and gives structural insights that we expect will be useful elsewhere. We show: 1) that differential privacy ensures that the expectation of any query on the *conditional distribution* on datasets induced by the transcript of the interaction is close to its expectation on the data distribution, and 2) sample accuracy on its own ensures that any query answer produced by the mechanism is close to the expectation of the query on the conditional distribution. This second claim follows from a thought experiment in which we imagine that the dataset is resampled from the conditional distribution after the mechanism has committed to its answers. The transfer theorem then follows by summing these two bounds, and in particular, avoids the "monitor argument" used to derive high probability bounds in prior work.

An upshot of our new proof technique is that the concrete bounds we obtain are substantially better than the best previously known bounds, even though the improvements are in the constants, rather than the asymptotics (which are known to be tight). As we show, our new bounds outperform the naive "sample-splitting" baseline at dramatically smaller dataset sizes compared to the previous state of the art, bringing techniques from this literature closer to practicality.

2012 ACM Subject Classification Theory of computation → Sample complexity and generalization bounds

Keywords and phrases Differential Privacy, Adaptive Data Analysis, Transfer Theorem

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.31

Related Version arXiv version available at `https://arxiv.org/abs/1909.03577`.

Funding *Christopher Jung*: Supported in part by NSF grant AF-1763307.
Katrina Ligett: Supported in part by Israel Science Foundation (ISF) grant #1044/16, the United States Air Force and DARPA under contracts FA8750-16-C-0022 and FA8750-19-2-0222, and the Federmann Cyber Security Center in conjunction with the Israel national cyber directorate.

Seth Neel: Supported in part by an NSF Graduate Research Fellowship.
Aaron Roth: Supported in part by NSF grant AF-1763314, the United States Air Force and DARPA under Contract No FA8750-16-C-0022, and a grant from the Sloan Foundation.
Moshe Shenfeld: Supported in part by Israel Science Foundation (ISF) grant #1044/16, the United States Air Force and DARPA under contracts FA8750-16-C-0022 and FA8750-19-2-0222, and the Federmann Cyber Security Center in conjunction with the Israel national cyber directorate. Any opinions,findings and conclusions or recommendations expressed in this materialare those of the author(s) and do not necessarily reflect the views ofthe United States Air Force and DARPA.

Acknowledgements We thank Adam Smith for helpful conversations at an early stage of this work, and Daniel Roy for helpful feedback on the presentation of the result.

1 Introduction

Many data analysis pipelines are *adaptive*: the choice of which analysis to run next depends on the outcome of previous analyses. Common examples include variable selection for regression problems and hyper-parameter optimization in large-scale machine learning problems: in both cases, common practice involves repeatedly evaluating a series of models on the same dataset. Unfortunately, this kind of adaptive re-use of data invalidates many traditional methods of avoiding over-fitting and false discovery, and has been blamed in part for the recent flood of non-reproducible findings in the empirical sciences [14].

There is a simple way around this problem: don't re-use data. This idea suggests a baseline called *data splitting*: to perform k analyses on a dataset, randomly partition the dataset into k disjoint parts, and perform each analysis on a fresh part. The standard "holdout method" is the special case of $k = 2$. Unfortunately, this natural baseline makes poor use of data: in particular, the data requirements of this method grow *linearly* with the number of analyses k to be performed.

A recent literature starting with Dwork et al. [6] shows how to give a significant asymptotic improvement over this baseline via a connection to differential privacy: rather than computing and reporting exact sample quantities, perturb these quantities with noise. This line of work established a powerful *transfer theorem*, that informally says that any analysis that is simultaneously differentially private and accurate *in-sample* will also be accurate *out-of-sample*. The best analysis of this technique shows that for a broad class of analyses and a target accuracy goal, the data requirements grow only with $\sqrt{k}$ – a quadratic improvement over the baseline [1]. Moreover, it is known that in the worst case, this cannot be improved asymptotically [15, 23]. Unfortunately, thus far this literature has had little impact on practice. One major reason for this is that although the more sophisticated techniques from this literature give asymptotic improvements over the sample-splitting baseline, the concrete bounds do not actually improve on the baseline until the dataset is enormous. This remains true even after optimizing the constants that arise from the arguments of Dwork et al. [6] or Bassily et al. [1], and appears to be a fundamental limitation of their proof techniques [20]. In this paper, we give a new proof of the transfer theorem connecting differential privacy and in-sample accuracy to out-of-sample accuracy. Our proof is based on a simple insight that arises from imagining a "resampling" experiment, and in particular yields an improved concrete bound that beats the sample-splitting baseline at dramatically smaller data set sizes n compared to prior work. In fact, at reasonable dataset sizes, the magnitude of the improvement arising from our new theorem is significantly larger than the improvement between the bounds of Bassily et al. [1] and Dwork et al. [6]: see Figure 1.

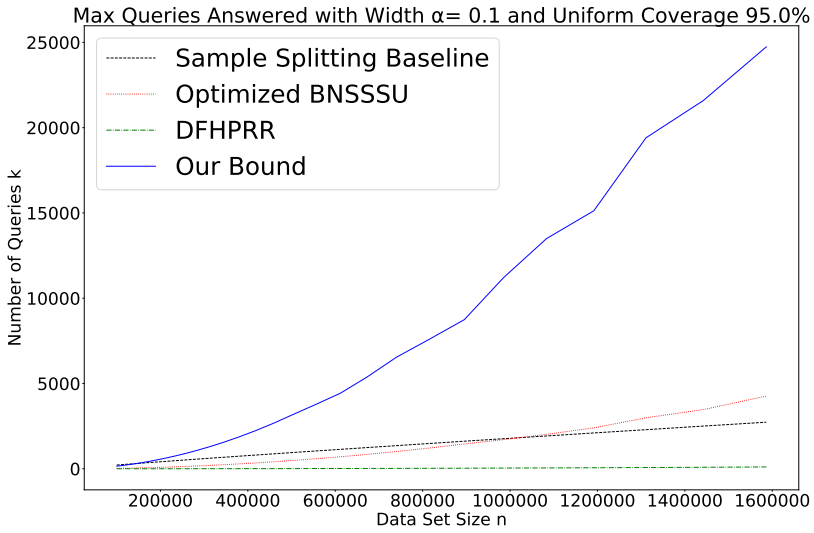


Figure 1 A comparison of the number of adaptive linear queries that can be answered using the Gaussian mechanism as analyzed by our transfer theorem (Theorem 9), the numerically optimized variant of the bound from Bassily et al. [1] (Optimized BNSSSU) as derived in [20], and the original transfer theorem from Dwork et al. [6] (DFHPRR). We plot for each dataset size n, the number of queries k that can be answered while guaranteeing confidence intervals around the answer that have width $\alpha = 0.1$ and uniform coverage probability $1 - \beta = 0.95$. We compare with the naive sample splitting baseline that simply splits the dataset into k pieces and answers each query with the empirical answer on a fresh piece.

1.1 Proof Techniques

Prior Work

Consider an unknown data distribution $\mathcal{P}$ over a data-domain $\mathcal{X}$, and a dataset $S \sim \mathcal{P}^n$ consisting of n i.i.d. draws from $\mathcal{P}$. It is a folklore observation (attributed to Frank McSherry) that if a predicate $q : \mathcal{X} \to [0, 1]$ is selected by an ϵ-differentially private algorithm M acting on S, then it will generalize *in expectation* (or *have low bias*) in the sense that $|\mathbb{E}_{q \sim M(S)}[\mathbb{E}_{x \sim \mathcal{P}}[q(x)] - \frac{1}{n}\sum_{x \in S} q(x)]| \approx \epsilon$. But bounds on bias are not enough to yield tight confidence intervals, and so prior work has focused on strengthening the above observation into a high probability bound. For small ϵ, the optimal bound has the asymptotic form: $\Pr_{q \sim M(S)}[|\mathbb{E}_{x \sim \mathcal{P}}[q(x)] - \frac{1}{n}\sum_{x \in S} q(x)| \geq \epsilon] \leq e^{-O(\epsilon^2 n)}$ [1]. Note that this bound does not refer to the *estimated answers* supplied to the data analyst: it says only that a differentially private data analyst is unlikely to be able to find a query whose average value on the dataset differs substantially from its expectation. Pairing this with a simultaneous high probability bound on the *in-sample accuracy* of a mechanism – that it supplies answers a such that with high probability the empirical error is small: $\Pr_{a \sim M(S)}[|a - \frac{1}{n}\sum_{x \in S} q(x)| \geq \alpha] \leq \beta$ – yields a bound on out-of-sample accuracy via the triangle inequality.

Dwork et al. [6] proved their high probability bound via a direct computation on the *moments* of empirical query values, but this technique was unable to achieve the optimal rate. Bassily et al. [1] proved a bound with the optimal rate by introducing the ingenious *monitor technique*. This important technique has subsequently found other uses [24, 19, 13], but is a heavy hammer that seems unavoidably to yield large constant overhead, even after numeric optimization [20].

Our Approach

We take a fundamentally different approach by directly providing high probability bounds on the out-of-sample accuracy $|a - \mathbb{E}_{x\sim\mathcal{P}}[q(x)]|$ of mechanisms that are both differentially private and accurate in-sample. Our elementary approach is motivated by the following thought experiment: in actuality, the dataset S is fixed before any interaction with M begins. However, imagine that after the entire interaction with M is complete, the dataset S is *resampled* from the conditional distribution $\mathcal{Q}$ on datasets *conditioned* on the output of M. This thought experiment doesn't alter the joint distribution on datasets and outputs, and so any in-sample accuracy guarantees that M has continue to hold under this hypothetical re-sampling experiment. But because the empirical value of the queries on the re-sampled dataset are likely to be close to their expected value over the conditional distribution $\mathcal{Q}$, the only way the mechanism can promise to be sample-accurate with high probability is if it provides answers that are *close to their expected value over the conditional distribution with high probability.*

This focuses attention on the conditional distribution on datasets induced by differentially private transcripts. But it is not hard to show that a consequence of differential privacy is that the conditional expectation of any query must be close to its expectation over the data distribution with high probability. In contrast to prior work, this argument directly leverages high-probability in-sample accuracy guarantees of a private mechanism to derive high-probability out-of-sample guarantees, without the need for additional machinery like the monitor argument of Bassily et al. [1].

1.2 Further Related Work

The study of "adaptive data analysis" was initiated by Dwork et al. [6, 5] who provided upper bounds via a connection to differential privacy, and Hardt and Ullman [15] who provided lower bounds via a connection to fingerprinting codes. The upper bounds were subsequently strengthened by Bassily et al. [1], and the lower bounds by Steinke and Ullman [23] to be (essentially) matching, asymptotically. The upper bounds were optimized by Rogers et al. [20], which we use in our comparisons. Subsequent work proved transfer theorems related to other quantities like description length bounds [4] and compression schemes [3], and expanded the types of analyses whose generalization properties we could reason about via a connection to a quantity called approximate max information [4, 21]. Feldman and Steinke [11, 12] give improved methods that could guarantee out-of-sample accuracy bounds that depended on query variance. Neel and Roth [17] extend the transfer theorems from this literature to the related problem of adaptive data gathering, which was identified by Nie et al. [18]. Ligett and Shenfeld [16] give an algorithmic stability notion they call *local statistical stability* (also defined with respect to a conditional data distribution) that they show asymptotically characterizes the ability of mechanisms to offer high probability out-of-sample generalization guarantees for linear queries. A related line of work initiated by Russo and Zou [22] and extended by Xu and Raginsky [25] starts with weaker assumptions on the mechanism (mutual information bounds), and derives weaker conclusions (bounds on bias, rather than high probability generalization guarantees).

A more recent line of work aims at mitigating the fact that the worst-case bounds deriving from transfer theorems do not give non-trivial guarantees on reasonably sized datasets. Zrnic and Hardt [26] show that better bounds can be derived under the assumption that the data analyst is restricted in various ways to not be fully adaptive. Feldman et al. [10] show that overfitting by a classifier because of test-set re-use is mitigated in multi-label prediction

problems, compared to binary prediction problems. Rogers et al. [20] give a method for certifying the correctness of heuristically guessed confidence intervals, which they show often out-perform the theoretical guarantees by orders of magnitude.

Finally, Elder [9, 8] proposes a Bayesian reformulation of the adaptive data analysis problem. In the model of [9], the data distribution $\mathcal{P}$ is assumed to itself be drawn from a prior that is commonly known to the data analyst and mechanism. In contrast, we work in the standard adversarial setting originally introduced by Dwork et al. [6] in which the mechanism must offer guarantees for worst case data distributions and analysts, and focus our attention on conditional distributions purely as a proof technique.

2 Preliminaries

Let $\mathcal{X}$ be an abstract data domain, and let $\mathcal{P}$ be an arbitrary distribution over $\mathcal{X}$. A dataset of size n is a collection of n data records: $S = \{S_i\}_{i=1}^n \in \mathcal{X}^n$. We study datasets sampled *i.i.d.* from $\mathcal{P}$: $S \sim \mathcal{P}^n$. We will write S to denote the random variable and $\boldsymbol{x}$ for realizations of this random variable. A linear query is a function $q : \mathcal{X}^* \to [0, 1]$ that takes the following empirical average form when acting on a data set $S \in \mathcal{X}^n$:

$$q(S) = \frac{1}{n}\sum_{i=1}^{n} q(S_i).$$

We will be interested in estimating the expectations of linear queries over $\mathcal{P}$. Abusing notation, given a distribution $\mathcal{D}$ over datasets, we write $q(\mathcal{D})$ to denote the expectation of q over datasets drawn from $\mathcal{D}$, and write $S_i \sim S$ to denote a datapoint sampled uniformly at random from a dataset S. Note that for linear queries we have:

$$q(\mathcal{D}) = \mathop{\mathbb{E}}_{S\sim\mathcal{D}}[q(S)] = \mathop{\mathbb{E}}_{S\sim\mathcal{D},S_i\sim S}[q(S_i)]$$

We note that for linear queries, when the dataset distribution $\mathcal{D} = \mathcal{P}^n$, we have $q(\mathcal{P}^n) = \mathbb{E}_{x\sim\mathcal{P}}[q(x)]$, which we write as $q(\mathcal{P})$ when the notation is clear from context. However, the more general definition will be useful because we will need to evaluate the expectation of q over other (non-product) distributions over datasets in our arguments, and we will generalize beyond linear queries in Appendices A.1 and A.2.

Given a family of queries Q, a statistical estimator is a (possibly stateful) randomized algorithm $M : \mathcal{X}^n \times Q^* \to \mathbb{R}^*$ parameterized by a dataset S that interactively takes as input a stream of queries $q_i \in Q$, and provides answers $a_i \in \mathbb{R}$. An analyst is an arbitrary randomized algorithm $\mathcal{A} : \mathbb{R}^* \to Q^*$ that generates a stream of queries and receives a stream of answers (which can inform the next queries it generates). When an analyst interacts with a statistical estimator, they generate a transcript of their interaction $\pi \in \mathbf{\Pi}$ where $\mathbf{\Pi} = (Q \times \mathbb{R})^*$ is the space of all transcripts. Throughout we write Π to denote the transcript's random variable and π for its realizations.

The interaction is summarized in Algorithm 1, and we write $\text{Interact}(M, \mathcal{A}; S)$ to refer to it. When M and $\mathcal{A}$ are clear from context, we will abbreviate this notation and write simply $I(S)$. When we refer to an indexed query q_j, this is implicitly a function of the transcript π. Given a transcript $\pi \in \mathbf{\Pi}$, write $\mathcal{Q}_\pi$ to denote the conditional distribution on datasets conditional on $\Pi = \pi$: $\mathcal{Q}_\pi = (\mathcal{P}^n)|\text{Interact}(M, \mathcal{A}; S) = \pi$. Note that $\mathcal{Q}_\pi$ will no longer generally be a product distribution. We will be interested in evaluating uniform accuracy bounds, which control the worst-case error over all queries:

Algorithm 1 Interact$(M, \mathcal{A}; S)$: An Analyst Interacting with a Statistical Estimator to Generate a Transcript.

Input: A statistical estimator M, an analyst $\mathcal{A}$, and a dataset $S \in \mathcal{X}^n$.
for $t = 1$ *to* k **do**
  The analyst generates a query $q_t \leftarrow \mathcal{A}(a_1, \ldots, a_{t-1})$ and sends it to the statistical estimator;
  The statistical estimator generates an answer $a_t \leftarrow M(S; q_t)$;
return $\Pi = ((q_1, a_1), \ldots, (q_k, a_k))$.

▶ **Definition 1** (Accuracy). *M satisfies (α, β)-sample accuracy if for every data analyst $\mathcal{A}$ and every data distribution $\mathcal{P}$,*

$$\Pr_{S\sim\mathcal{P}^n, \Pi\sim Interact(M,\mathcal{A};S)}[\max_j |q_j(S) - a_j| \geq \alpha] \leq \beta.$$

We say M satisfies (α, β)-distributional accuracy if for every data analyst $\mathcal{A}$ and every data distribution $\mathcal{P}$,

$$\Pr_{S\sim\mathcal{P}^n, \Pi\sim Interact(M,\mathcal{A};S)}[\max_j |q_j(\mathcal{P}^n) - a_j| \geq \alpha] \leq \beta.$$

We will be interested in interactions I that satisfy *differential privacy.*

▶ **Definition 2** (Differential Privacy [7]). *Two datasets $S, S' \in \mathcal{X}^n$ are neighbors if they differ in at most one coordinate. An interaction Interact$(M, \cdot; \cdot)$ satisfies (ϵ, δ)-differential privacy if for all data analysts $\mathcal{A}$, pairs of neighboring datasets $S, S' \in \mathcal{X}^n$, and for all events $E \subseteq \mathbf{\Pi}$:*

$$\Pr_{\Pi\sim Interact(M,\mathcal{A};S)}[\Pi \in E] \leq e^\epsilon \cdot \Pr_{\Pi\sim Interact(M,\mathcal{A};S')}[\Pi \in E] + \delta.$$

If Interact$(M, \cdot; \cdot)$ satisfies (ϵ, δ)-differential privacy, we will also say that M satisfies (ϵ, δ)-differential privacy.

We introduce a novel quantity that will be crucial to our argument: it captures the effect of the transcript on the change in the expectation of a query contained in the transcript.

▶ **Definition 3.** *An interaction Interact$(M, \mathcal{A}; \cdot)$ is called (ϵ, δ)-posterior stable if for every data distribution $\mathcal{P}$:*

$$\Pr_{S\sim\mathcal{P}^n, \Pi\sim Interact(M,\mathcal{A};S)}[\max_j |q_j(\mathcal{P}^n) - q_j(\mathcal{Q}_\Pi)| \geq \epsilon] \leq \delta.$$

3 An Elementary Proof of the Transfer Theorem

3.1 A General Transfer Theorem

In this section we prove a general transfer theorem for sample accurate mechanisms with low posterior stability. In Section 3.2 we prove that differentially private mechanisms have low posterior stability.

▶ **Theorem 4** (General Transfer Theorem). *Suppose that Interact$(M, \mathcal{A}; \cdot)$ is an (α, β)-sample accurate, (ϵ, δ)-posterior stable interaction. Then for every $c > 0$ it also satisfies:*

$$\Pr_{S\sim\mathcal{P}^n, \Pi\sim Interact(M,\mathcal{A};S)}[\max_j |a_j - q_j(\mathcal{P})| > \alpha + c + \epsilon] \leq \frac{\beta}{c} + \delta$$

i.e. it is (α', β')-distributionally accurate for $\alpha' = \alpha + c + \epsilon$ and $\beta' = \frac{\beta}{c} + \delta$.

The theorem follows easily from a change in perspective driven by an elementary observation. Imagine that *after* the interaction is run and results in a transcript π, the dataset S is *resampled* from its conditional distribution $\mathcal{Q}_\pi$. This does not change the joint distribution on datasets and transcripts. This simple claim is formalized below: its elementary proof appears in Appendix B.

▶ **Lemma 5** (Resampling Lemma). *Let $E \subseteq \mathcal{X}^n \times \mathbf{\Pi}$ be any event. Then:*

$$\Pr_{S\sim\mathcal{P}^n,\Pi\sim I(S)}[(S,\Pi)\in E] = \Pr_{S\sim\mathcal{P}^n,\Pi\sim I(S),S'\sim\mathcal{Q}_\Pi}[(S',\Pi)\in E]$$

The change in perspective suggested by the resampling lemma makes it easy to see why the following must be true: any sample-accurate mechanism must in fact be accurate with respect to the conditional distribution it induces. This is because if it can first commit to answers, and guarantee that they are sample-accurate *after* the dataset is resampled from the conditional, the answers it committed to must have been close to the conditional means, because it is likely that the empirical answers on the resampled dataset will be. This argument is generic and does not use differential privacy.

▶ **Lemma 6.** *Suppose that M is (α,β)-sample accurate. Then for every $c>0$ it also satisfies:*

$$\Pr_{S\sim\mathcal{P}^n,\,\Pi\sim Interact(M,\mathcal{A};S)}[\max_j |a_j - q_j(\mathcal{Q}_\Pi)| > \alpha + c] \le \frac{\beta}{c}$$

Proof. Denote by $j^*(\pi) = \arg\max_j |a_j - q_j(\mathcal{Q}_\pi)|$. Given $\alpha \ge 0$ and $c>0$, and expanding the definition of $q_{j^*(\Pi)}(\mathcal{Q}_\Pi)$ we get:

$$\begin{aligned}
&\Pr_{S\sim\mathcal{P}^n,\Pi\sim I(S)}\left[a_{j^*(\Pi)} - q_{j^*(\Pi)}(\mathcal{Q}_\Pi) > \alpha + c\right] \\
&\quad= \Pr_{S\sim\mathcal{P}^n,\Pi\sim I(S)}\left[\mathop{\mathbb{E}}_{S'\sim\mathcal{Q}_\Pi}\left[a_{j^*(\Pi)} - q_{j^*(\Pi)}(S') - \alpha\right] > c\right] \\
&\quad\le \Pr_{S\sim\mathcal{P}^n,\Pi\sim I(S)}\left[\mathop{\mathbb{E}}_{S'\sim\mathcal{Q}_\Pi}\left[\max\left\{a_{j^*(\Pi)} - q_{j^*(\Pi)}(S') - \alpha, 0\right\}\right] > c\right] \\
&\quad\overset{(1)}{\le} \frac{1}{c}\mathop{\mathbb{E}}_{S\sim\mathcal{P}^n,\Pi\sim I(S)}\left[\mathop{\mathbb{E}}_{S'\sim\mathcal{Q}_\Pi}\left[\max\left\{a_{j^*(\Pi)} - q_{j^*(\Pi)}(S') - \alpha, 0\right\}\right]\right] \\
&\quad\overset{(2)}{\le} \frac{1}{c}\mathop{\mathbb{E}}_{S\sim\mathcal{P}^n,\Pi\sim I(S)}\left[\Pr_{S'\sim\mathcal{Q}_\Pi}\left[a_{j^*(\Pi)} - q_{j^*(\Pi)}(S') - \alpha > 0\right]\right] \\
&\quad= \frac{1}{c}\Pr_{S\sim\mathcal{P}^n,\Pi\sim I(S),S'\sim\mathcal{Q}_\Pi}\left[a_{j^*(\Pi)} - q_{j^*(\Pi)}(S') > \alpha\right] \\
&\quad\overset{(3)}{=} \frac{1}{c}\Pr_{S\sim\mathcal{P}^n,\Pi\sim I(S)}\left[a_{j^*(\Pi)} - q_{j^*(\Pi)}(S) > \alpha\right]
\end{aligned}$$

Here, inequality (1) follows from Markov's inequality, inequality (2) follows from the fact that $a_{j^*(\Pi)} - q_{j^*(\Pi)}(S') - \alpha \le 1$, and equality 3 follows from the Resampling Lemma (Lemma 5). Repeating this argument for $q_{j^*(\Pi)}(Q_\Pi) - a_{j^*(\Pi)}$ yields a symmetric bound, so by combining the two with the guarantee of (α,β)-sample accuracy we get,

$$\begin{aligned}
\Pr_{S\sim\mathcal{P}^n,\Pi\sim I(S)}\left[\left|a_{j^*(\Pi)} - q_{j^*(\Pi)}(Q_\Pi)\right| > \alpha + c\right] &\le \frac{1}{c}\Pr_{S\sim\mathcal{P}^n,\Pi\sim I(S)}\left[\left|a_{j^*(\Pi)} - q_{j^*(\Pi)}(S)\right| > \alpha\right] \\
&\le \frac{\beta}{c}
\end{aligned}$$

◀

Because sample accuracy implies accuracy with respect to the conditional distribution, together with a bound on posterior stability, the transfer theorem follows immediately:

Proof of Theorem 4. By the triangle inequality:

$$\max_j |a_j - q_j(\mathcal{P})| \leq \max_i |a_i - q_i(\mathcal{Q}_\Pi)| + \max_l |q_l(\mathcal{Q}_\Pi) - q_l(\mathcal{P})|.$$

Lemma 6 bounds the first term by $\alpha + c$ with probability $1 - \frac{\beta}{c}$ over Π, and the definition of posterior stability bounds the second term by ϵ with probability $1 - \delta$ over Π, which concludes the proof. ◀

3.2 A Transfer Theorem for Differential Privacy

In this section we prove a transfer theorem for differentially private mechanisms by demonstrating that they have low posterior stability and applying our general transfer theorem.

We here show that differentially private mechanisms have low posterior stability for linear queries. In the Appendix we extend this argument to low-sensitivity and optimization queries.

▶ **Lemma 7.** *If M is (ϵ, δ)-differentially private, then for any data distribution $\mathcal{P}$, any analyst $\mathcal{A}$, and any constant $c > 0$:*

$$\Pr_{S \sim \mathcal{P}^n, \Pi \sim Interact(M, \mathcal{A}; S)} \left[\max_j |q_j(\mathcal{Q}_\Pi) - q_j(\mathcal{P})| > (e^\epsilon - 1) + 2c \right] \leq \frac{\delta}{c}$$

i.e. it is (ϵ', δ')-posterior stable for every data analyst $\mathcal{A}$, where $\epsilon' = e^\epsilon - 1 + 2c$ and $\delta' = \frac{\delta}{c}$.

Proof. Given a transcript $\pi \in \mathbf{\Pi}$, let $j^*(\pi) \in \arg\max_j |q_j(\mathcal{Q}_\pi) - q_j(\mathcal{P})|$. Define for an $\alpha > 0$:

$$\mathbf{\Pi}_\alpha = \left\{ \pi \in \mathbf{\Pi} | \, q_{j^*(\pi)}(\mathcal{Q}_\pi) - q_{j^*(\pi)}(\mathcal{P}) > \alpha \right\}$$

$$\mathcal{X}^+(\pi) = \left\{ x \in \mathcal{X} | \, \Pr_{S \sim \mathcal{Q}_\pi, S_i \sim S} [S_i = x] > \Pr_{S_i \sim \mathcal{P}} [S_i = x] \right\}$$

$$B^+_\alpha = \bigcup_{\pi \in \mathbf{\Pi}_\alpha} \left(\mathcal{X}^+(\pi) \times \{\pi\} \right)$$

$$\mathbf{\Pi}^+_\alpha(x) = \left\{ \pi \in \mathbf{\Pi} | \, (x, \pi) \in B^+_\alpha \right\}$$

Fix any α. Suppose that $\Pr\left[\left| q_{j^*(\Pi)}(\mathcal{Q}_\Pi) - q_{j^*(\Pi)}(\mathcal{P}) \right| > \alpha \right] > \frac{\delta}{c}$. We must have that either $\Pr\left[q_{j^*(\Pi)}(\mathcal{Q}_\Pi) - q_{j^*(\Pi)}(\mathcal{P}) > \alpha \right] > \frac{\delta}{2c}$ or $\Pr\left[q_{j^*(\Pi)}(\mathcal{P}) - q_{j^*(\Pi)}(\mathcal{Q}_\Pi) > \alpha \right] > \frac{\delta}{2c}$. Without loss of generality, assume

$$\Pr\left[q_{j^*(\Pi)}(\mathcal{Q}_\Pi) - q_{j^*(\Pi)}(\mathcal{P}) > \alpha \right] = \Pr\left[\Pi \in \mathbf{\Pi}_\alpha \right] > \frac{\delta}{2c} \tag{1}$$

Let S_i be the random variable obtained by first sampling $S \sim \mathcal{P}^n$ and then sampling $S_i \in S$ uniformly at random. We compare the probability measure of B^+_α under the joint distribution on S_i and Π with its corresponding measure under the product distribution of S_i and Π:

$$
\begin{aligned}
&\Pr_{(S_i,\Pi)}\left[(S_i,\Pi)\in B_\alpha^+\right]-\Pr_{S_i\otimes\Pi}\left[(S_i,\Pi)\in B_\alpha^+\right]\\
&\quad=\sum_{\pi\in\mathbf{\Pi}_\alpha}\Pr[\Pi=\pi]\sum_{x\in\mathcal{X}^+(\pi)}\left(\Pr[S_i=x|\Pi=\pi]-\Pr[S_i=x]\right)\\
&\quad\geq\sum_{\pi\in\mathbf{\Pi}_\alpha}\Pr[\Pi=\pi]\sum_{x\in\mathcal{X}^+(\pi)}q_{j^*(\pi)}(x)\left(\Pr[S_i=x|\Pi=\pi]-\Pr[S_i=x]\right)\\
&\quad\geq\sum_{\pi\in\mathbf{\Pi}_\alpha}\Pr[\Pi=\pi]\sum_{x\in\mathcal{X}}q_{j^*(\pi)}(x)\left(\Pr[S_i=x|\Pi=\pi]-\Pr[S_i=x]\right)\\
&\quad=\sum_{\pi\in\mathbf{\Pi}_\alpha}\Pr[\Pi=\pi]\left(q_{j^*(\pi)}(\mathcal{Q}_\pi)-q_{j^*(\pi)}(\mathcal{P})\right)\\
&\quad>\alpha\cdot\Pr\left[\Pi\in\mathbf{\Pi}_\alpha\right]
\end{aligned}
$$

On the other hand, using the definition of (ϵ,δ)-differential privacy (See Lemma 21 for the elementary derivation of the first inequality):

$$
\begin{aligned}
&\Pr_{(S_i,\Pi)}\left[(S_i,\Pi)\in B_\alpha^+\right]-\Pr_{S_i\otimes\Pi}\left[(S_i,\Pi)\in B_\alpha^+\right]\\
&\quad=\sum_{x\in\mathcal{X}}\Pr[S_i=x]\left(\Pr\left[\Pi\in\mathbf{\Pi}_\alpha^+(x)|S_i=x\right]-\Pr\left[\Pi\in\mathbf{\Pi}_\alpha^+(x)\right]\right)\\
&\quad\leq\sum_{x\in\mathcal{X}}\Pr[S_i=x]\left((e^\epsilon-1)\Pr\left[\Pi\in\mathbf{\Pi}_\alpha^+(x)\right]+\delta\right)\\
&\quad=(e^\epsilon-1)\Pr_{S_i\otimes\Pi}\left[(S_i,\Pi)\in B_\alpha^+\right]+\delta\\
&\quad\leq(e^\epsilon-1)\Pr\left[\Pi\in\mathbf{\Pi}_\alpha\right]+\delta\\
&\quad<(e^\epsilon-1)\Pr\left[\Pi\in\mathbf{\Pi}_\alpha\right]+2c\Pr\left[\Pi\in\mathbf{\Pi}_\alpha\right]\quad\text{(by Equation (1))}\\
&\quad=\left((e^\epsilon-1)+2c\right)\cdot\Pr\left[\Pi\in\mathbf{\Pi}_\alpha\right]
\end{aligned}
$$

This is a contradiction for $\alpha\geq(e^\epsilon-1)+2c$. ◀

▶ Remark 8. Note

1. Since differential privacy is closed under post processing, this claim can be generalized beyond queries contained in the transcript to any query generated as function of the transcript.
2. In the case of $(\epsilon,0)$-differential privacy, choosing $c=0$, the claim holds for every query with probability 1.

Combined with our general transfer theorem (Theorem 4), this directly yields a transfer theorem for differential privacy:

▶ **Theorem 9** (Transfer Theorem for (ϵ,δ)-Differential Privacy)**.** *Suppose that M is (ϵ,δ)-differentially private and (α,β)-sample accurate for linear queries. Then for every analyst $\mathcal{A}$ and $c,d>0$ it also satisfies:*

$$
\Pr_{S\sim\mathcal{P}^n,\,\Pi\sim Interact(M,\mathcal{A};S)}[\max_j|a_j-q_j(\mathcal{P})|>\alpha+(e^\epsilon-1)+c+2d]\leq\frac{\beta}{c}+\frac{\delta}{d}
$$

i.e. it is (α',β')-distributionally accurate for $\alpha'=\alpha+(e^\epsilon-1)+c+2d$ and $\beta'=\frac{\beta}{c}+\frac{\delta}{d}$.

▶ Remark 10. As we will see in Section 4, the Gaussian mechanism (and many other differentially private mechanisms) has a sample accuracy bound that depends only on the square root of the log of both $1/\beta$ and $1/\delta$. Thus, despite the Markov-like term $\beta' = \frac{\beta}{c} + \frac{\delta}{d}$ in the above transfer theorem, together with the sample accuracy bounds of the Gaussian mechanism, it yields Chernoff-like concentration.

Our technique extends easily to reason about arbitrary low sensitivity queries and minimization queries. See Appendix A.1 and A.2 for more details.

4 Applications: The Gaussian Mechanism

We now apply our new transfer theorem to derive the concrete bounds that we plotted in Figure 1. The Gaussian mechanism is extremely simple and has only a single parameter σ: for each query q_i that arrives, the Gaussian mechanism returns the answer $a_i \sim \mathcal{N}(q_i(S), \sigma^2)$ where $\mathcal{N}(q_i(S), \sigma^2)$ denotes the Gaussian distribution with mean $q_i(S)$ and standard deviation σ. First, we recall the differential privacy properties of the Gaussian mechanism.

▶ **Theorem 11** ([2]). *When used to answer k linear queries, for every $0 < \delta < 1$, the Gaussian mechanism with parameter σ satisfies (ϵ, δ)-differential privacy for:*

$$\epsilon = \frac{k}{2n^2\sigma^2} + \sqrt{2\frac{k}{n^2\sigma^2}\log\left(\sqrt{\pi \cdot \frac{k}{2n^2\sigma^2}}/\delta\right)}$$

It is also easy to see that the sample-accuracy of the Gaussian mechanism is characterized by the CDF of the Gaussian distribution:

▶ **Lemma 12.** *For any $0 < \beta < 1$, the Gaussian mechanism with parameter σ is (α_G, β)-sample accurate for:*

$$\alpha_G = \sqrt{2}\sigma \cdot \mathrm{erfc}^{-1}\left(2 - 2\left(1 - \frac{\beta}{2}\right)^{1/k}\right) < \sqrt{2}\sigma \cdot \mathrm{erfc}^{-1}\left(\frac{\beta}{k}\right) < \sqrt{2}\sigma\sqrt{\log\left(\frac{\sqrt{2}k}{\pi\beta}\right)}.$$

Above, $\mathrm{erfc}(x) = 1 - \mathrm{erf}(x)$ is the complementary error function.

Proof. For a query q_j, write $a_j = q_j(S) + Z_j$ where $Z_j \sim \mathcal{N}(0, \sigma^2)$. The sample error is $\max_j |a_j - q_j(S)| = \max_j |Z_j|$. We have that $\Pr[\max_j |Z_j| \geq \alpha] \leq \Pr[\max_j Z_j \geq \alpha] + \Pr[\min_j Z_j \leq -\alpha]$. α_G is the value that solves the equation $\Pr[\max_j Z_j \geq \alpha] = \Pr[\min_j Z_j \leq -\alpha] = \beta/2$ ◀

With these quantities in hand, we can now apply Theorem 9 to derive distributional accuracy bounds for the Gaussian mechanism:

▶ **Theorem 13.** *Fix a desired confidence parameter $0 < \beta < 1$. When σ is set optimally, the Gaussian mechanism can be used to answer k linear queries while satisfying (α, β)-distributional accuracy, where α is the solution to the following unconstrained minimization problem:*

$$\alpha = \min_{\sigma,\delta>0}\left\{\sqrt{2}\sigma \cdot \mathrm{erfc}^{-1}\left(\frac{\delta}{k}\right) + e^{\frac{k}{2n^2\sigma^2} + \sqrt{2\frac{k}{n^2\sigma^2}\log\left(\sqrt{\pi\cdot\frac{k}{2n^2\sigma^2}}/\delta\right)}} - 1 + 6\left(\frac{\delta}{\beta}\right)\right\}$$

Proof. Using Theorem 9 and fixing $\beta' = \delta$ and $c = d$, we have that an (α', β')-sample accurate, (ϵ, δ)-differentially private mechanism is (α, β)-distributionally accurate for $\alpha = \alpha' + (e^\epsilon - 1) + 3c$ and $\beta = \frac{2\delta}{c}$ where c can be an arbitrary parameter. For any fixed value of β, we can take $c = \frac{2\delta}{\beta}$, and see that we obtain (α, β)-distributional accuracy where $\alpha = \alpha' + (e^\epsilon - 1) + 6\,(\delta/\beta)$. The theorem then follows from plugging in the privacy bound from Theorem 11, the sample accuracy bound from Theorem 12, and optimizing over the free variables σ and δ. ◀

5 Discussion

We have given a new proof of the transfer theorem for differential privacy that has several appealing properties. Besides being simpler than previous arguments, it achieves substantially better concrete bounds than previous transfer theorems, and uncovers new structural insights about the role of differential privacy and sample accuracy. In particular, sample accuracy serves to guarantee that the reported answers are close to their conditional means, and differential privacy serves to guarantee that the conditional means are close to their true answers. This focuses attention on the conditional data distribution as a key quantity of interest, which we expect will be fruitful in future work. In particular, it may shed light on what makes certain data analysts overfit less than worst-case bounds would suggest: because they choose queries whose conditional means are closer to the prior than the worst-case query.

There seems to be one remaining place to look for improvement in our transfer theorem: Lemmas 6 and 7 both exhibit a Markov-like tradeoff between a parameter c and β and δ respectively. Although the dependence on β and δ in our ultimate bounds is only root-logarithmic, it would still yield an improvement if this Markov-like dependence could be replaced with a Chernoff-like dependence. It *is* possible to do this for the β parameter: we give an alternative (and even simpler) proof of the transfer theorem for $(\epsilon, 0)$-differential privacy which shows that conditional distributions induced by private mechanisms exhibit Chernoff-like concentration, in Appendix D. But the only way we know to extend this argument to (ϵ, δ)-differential privacy requires dividing δ by a factor of n, which yields a final theorem that is inferior to Theorem 9.

References

1. Raef Bassily, Kobbi Nissim, Adam Smith, Thomas Steinke, Uri Stemmer, and Jonathan Ullman. Algorithmic stability for adaptive data analysis. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 1046–1059. ACM, 2016.
2. Mark Bun and Thomas Steinke. Concentrated differential privacy: Simplifications, extensions, and lower bounds. In *Theory of Cryptography Conference*, pages 635–658. Springer, 2016.
3. Rachel Cummings, Katrina Ligett, Kobbi Nissim, Aaron Roth, and Zhiwei Steven Wu. Adaptive learning with robust generalization guarantees. In *Conference on Learning Theory*, pages 772–814, 2016.
4. Cynthia Dwork, Vitaly Feldman, Moritz Hardt, Toni Pitassi, Omer Reingold, and Aaron Roth. Generalization in adaptive data analysis and holdout reuse. In *Advances in Neural Information Processing Systems*, pages 2350–2358, 2015.
5. Cynthia Dwork, Vitaly Feldman, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Aaron Roth. The reusable holdout: Preserving validity in adaptive data analysis. *Science*, 349(6248):636–638, 2015.
6. Cynthia Dwork, Vitaly Feldman, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Aaron Leon Roth. Preserving statistical validity in adaptive data analysis. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 117–126. ACM, 2015.

7 Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284. Springer, 2006.

8 Sam Elder. Bayesian adaptive data analysis guarantees from subgaussianity. *arXiv preprint*, 2016. arXiv:1611.00065.

9 Sam Elder. Challenges in bayesian adaptive data analysis. *arXiv preprint*, 2016. arXiv:1604.02492.

10 Vitaly Feldman, Roy Frostig, and Moritz Hardt. The advantages of multiple classes for reducing overfitting from test set reuse. In *International Conference on Machine Learning*, pages 1892–1900, 2019.

11 Vitaly Feldman and Thomas Steinke. Generalization for Adaptively-chosen Estimators via Stable Median. In *Conference on Learning Theory*, pages 728–757, 2017.

12 Vitaly Feldman and Thomas Steinke. Calibrating Noise to Variance in Adaptive Data Analysis. In *Conference On Learning Theory*, pages 535–544, 2018.

13 Vitaly Feldman and Jan Vondrak. Generalization bounds for uniformly stable algorithms. In *Advances in Neural Information Processing Systems*, pages 9747–9757, 2018.

14 Andrew Gelman and Eric Loken. The Statistical Crisis in Science. *American Scientist*, 102(6):460, 2014.

15 Moritz Hardt and Jonathan Ullman. Preventing false discovery in interactive data analysis is hard. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, pages 454–463. IEEE, 2014.

16 Katrina Ligett and Moshe Shenfeld. A necessary and sufficient stability notion for adaptive generalization. *arXiv preprint*, 2019. arXiv:1906.00930.

17 Seth Neel and Aaron Roth. Mitigating Bias in Adaptive Data Gathering via Differential Privacy. In *International Conference on Machine Learning (ICML)*, 2018.

18 Xinkun Nie, Xiaoying Tian, Jonathan Taylor, and James Zou. Why Adaptively Collected Data Have Negative Bias and How to Correct for It. In *International Conference on Artificial Intelligence and Statistics*, pages 1261–1269, 2018.

19 Kobbi Nissim and Uri Stemmer. Concentration Bounds for High Sensitivity Functions Through Differential Privacy. *Journal of Privacy and Confidentiality*, 9(1), 2019.

20 Ryan Rogers, Aaron Roth, Adam Smith, Nathan Srebro, Om Thakkar, and Blake Woodworth. Guaranteed Validity for Empirical Approaches to Adaptive Data Analysis. *arXiv preprint*, 2019. arXiv:1906.09231.

21 Ryan Rogers, Aaron Roth, Adam Smith, and Om Thakkar. Max-information, differential privacy, and post-selection hypothesis testing. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 487–494. IEEE, 2016.

22 Daniel Russo and James Zou. Controlling bias in adaptive data analysis using information theory. In *Artificial Intelligence and Statistics*, pages 1232–1240, 2016.

23 Thomas Steinke and Jonathan Ullman. Interactive fingerprinting codes and the hardness of preventing false discovery. In *Conference on Learning Theory*, pages 1588–1628, 2015.

24 Thomas Steinke and Jonathan Ullman. Subgaussian tail bounds via stability arguments. *arXiv preprint*, 2017. arXiv:1701.03493.

25 Aolin Xu and Maxim Raginsky. Information-theoretic analysis of generalization capability of learning algorithms. In *Advances in Neural Information Processing Systems*, pages 2524–2533, 2017.

26 Tijana Zrnic and Moritz Hardt. Natural Analysts in Adaptive Data Analysis. In *International Conference on Machine Learning*, pages 7703–7711, 2019.

A Extensions

A.1 Low Sensitivity Queries

Our technique extends easily to reason about arbitrary *low sensitivity* queries. We only need to generalize our lemma about posterior stability.

▶ **Definition 14.** *A query $q : \mathcal{X}^n \to \mathbb{R}$ is called Δ-sensitive if for all pairs of neighbouring datasets $S, S' \in \mathcal{X}^n$: $|q(S) - q(S')| \leq \Delta$. Note that linear queries are $(1/n)$-sensitive.*

▶ **Lemma 15.** *If M is an (ϵ, δ)-differentially private mechanism for answering Δ-sensitive queries, then for any data distribution $\mathcal{P}$, analyst $\mathcal{A}$, and any constant $c > 0$:*

$$\Pr_{S\sim\mathcal{P}^n, \Pi\sim Interact(M,\mathcal{A};S))}\left[\max_j |q_j(\mathcal{Q}_\Pi) - q_j(\mathcal{P}^n)| > (e^\epsilon - 1 + 4c)n\Delta\right] \leq \frac{\delta}{c}$$

i.e. it is $(\epsilon', \frac{\delta}{c})$-posterior stable for every $\mathcal{A}$, where $\epsilon' = (e^\epsilon - 1 + 4c)n\Delta$.

Proof. We introduce a useful bit of notation: $\bar{q}\left(\boldsymbol{x}_{\leq i}\right) = \mathop{\mathbb{E}}_{S'\sim\mathcal{P}^{n-i}}\left[q\left(\left(\boldsymbol{x}_{\leq i}, S'\right)\right)\right]$. Notice that $\bar{q}\left(\boldsymbol{x}_{\leq 0}\right) = q\left(\mathcal{P}^n\right)$ and $\bar{q}\left(\boldsymbol{x}_{\leq n}\right) = q\left(\boldsymbol{x}\right)$. Given a transcript $\pi \in \mathbf{\Pi}$, let $j^*(\pi) \in \arg\max_j |q_j(\mathcal{Q}_\pi) - q_j(\mathcal{P}^n)|$. Denote for any $\alpha \geq 0$

$$\mathbf{\Pi}_\alpha = \left\{\pi \in \mathbf{\Pi} \,|\, q_{j^*(\pi)}\left(\mathcal{Q}_\pi\right) - q_{j^*(\pi)}\left(\mathcal{P}^n\right) > \alpha\right\}$$

and for any $z \in [0, 2\Delta]$ denote

$$\mathbf{\Pi}_{\alpha,z}\left(\boldsymbol{x}_{\leq i}\right) = \left\{\pi \in \mathbf{\Pi}_\alpha \,|\, \bar{q}_{j^*(\pi)}\left(\boldsymbol{x}_{\leq i}\right) - \bar{q}_{j^*(\pi)}\left(\boldsymbol{x}_{\leq i-1}\right) > z - \Delta\right\}$$

From the definition of differential privacy:

$$\begin{aligned}
& \mathop{\mathbb{E}}_{S\sim\mathcal{P}^n}\left[\sum_{\pi\in\mathbf{\Pi}_\alpha} \Pr_{\Pi\sim I(S)}\left[\Pi = \pi\right]\left(\bar{q}_{j^*(\pi)}\left(S_{\leq i}\right) - \bar{q}_{j^*(\pi)}\left(S_{\leq i-1}\right) + \Delta\right)\right] \\
= & \mathop{\mathbb{E}}_{S\sim\mathcal{P}^n}\left[\int_0^{2\Delta} \Pr_{\Pi\sim I(S)}\left[\Pi \in \mathbf{\Pi}_{\alpha,z}\left(S_{\leq i}\right)\right] dz\right] \\
\leq & \mathop{\mathbb{E}}_{S\sim\mathcal{P}^n, Y\sim\mathcal{P}}\left[\int_0^{2\Delta}\left(e^\epsilon \Pr_{\Pi\sim I(S^{i\leftarrow Y})}\left[\Pi \in \mathbf{\Pi}_{\alpha,z}\left(S_{\leq i}\right)\right] + \delta\right) dz\right] \\
= & \mathop{\mathbb{E}}_{S\sim\mathcal{P}^n, Y\sim\mathcal{P}}\left[e^\epsilon \sum_{\pi\in\mathbf{\Pi}_\alpha} \Pr_{\Pi\sim I(S^{i\leftarrow Y})}\left[\Pi = \pi\right]\left(\bar{q}_{j^*(\pi)}\left(S_{\leq i}\right) - \bar{q}_{j^*(\pi)}\left(S_{\leq i-1}\right) + \Delta\right) + 2\Delta\delta\right] \\
= & \mathop{\mathbb{E}}_{S\sim\mathcal{P}^n, Y\sim\mathcal{P}}\left[e^\epsilon \sum_{\pi\in\mathbf{\Pi}_\alpha} \Pr_{\Pi\sim I(S)}\left[\Pi = \pi\right]\left(\bar{q}_{j^*(\pi)}\left(S^{i\leftarrow Y}_{\leq i}\right) - \bar{q}_{j^*(\pi)}\left(S_{\leq i-1}\right) + \Delta\right) + 2\Delta\delta\right]
\end{aligned}$$

where $S^{i\leftarrow Y} = (S_1, \ldots, S_{i-1}, Y, S_{i+1}, \ldots, S_n)$, and the last equality follows from the observation that (S, Y) and $(S^{i\leftarrow Y}, S_i)$ are identically distributed. Since $Y \sim \mathcal{P}$, independently from Π, we get that $\mathbb{E}_{Y\sim\mathcal{P}}\left[\bar{q}_{j^*(\pi)}\left(S^{i\leftarrow Y}_{\leq i}\right)\right] = \bar{q}_{j^*(\pi)}\left(S_{\leq i-1}\right)$, so

$$\begin{aligned}
& \mathop{\mathbb{E}}_{S\sim\mathcal{P}^n}\left[\sum_{\pi\in\mathbf{\Pi}_\alpha} \Pr_{\Pi\sim I(S)}\left[\Pi = \pi\right]\left(\bar{q}_{j^*(\pi)}\left(S_{\leq i}\right) - \bar{q}_{j^*(\pi)}\left(S_{\leq i-1}\right) + \Delta\right)\right] \\
\leq & \mathop{\mathbb{E}}_{S\sim\mathcal{P}^n}\left[\left(e^\epsilon \Pr_{\Pi\sim I(S)}\left[\Pi \in \mathbf{\Pi}_\alpha\right] + 2\delta\right)\Delta\right] \\
= & \left(e^\epsilon \Pr\left[\Pi \in \mathbf{\Pi}_\alpha\right] + 2\delta\right)\Delta
\end{aligned}$$

Subtracting $\Delta \Pr[\Pi \in \Pi_\alpha]$ from both sides we get

$$\mathop{\mathbb{E}}_{S\sim\mathcal{P}^n}\left[\sum_{\pi\in\mathbf{\Pi}_\alpha}\Pr_{\Pi\sim I(S)}[\Pi=\pi]\left(\bar{q}_{j^*(\pi)}(S_{\le i})-\bar{q}_{j^*(\pi)}(S_{\le i-1})\right)\right] \le ((e^\epsilon-1)\Pr[\Pi\in\mathbf{\Pi}_\alpha]+2\delta)\Delta \tag{2}$$

We now chooose $\alpha = (e^\epsilon - 1 + 4c)\, n\Delta$. Suppose that $\Pr\left[\left|q_{j^*(\Pi)}(\mathcal{Q}_\Pi) - q_{j^*(\Pi)}(\mathcal{P}^n)\right| > \alpha\right] > \frac{\delta}{c}$. We must have that either $\Pr\left[q_{j^*(\Pi)}(\mathcal{Q}_\Pi) - q_{j^*(\Pi)}(\mathcal{P}^n) > \alpha\right] > \frac{\delta}{2c}$ or $\Pr\left[q_{j^*(\Pi)}(\mathcal{P}^n) - q_{j^*(\Pi)}(\mathcal{Q}_\Pi) > \alpha\right] > \frac{\delta}{2c}$. Without loss of generality, assume

$$\Pr\left[q_{j^*(\Pi)}(\mathcal{Q}_\Pi) - q_{j^*(\Pi)}(\mathcal{P}^n) > \alpha\right] = \Pr[\Pi\in\mathbf{\Pi}_\alpha] > \frac{\delta}{2c} \tag{3}$$

But this leads to a contradiction, since

$$\begin{aligned}
&\Pr[\Pi\in\mathbf{\Pi}_\alpha]\,(e^\epsilon-1+4c)\,n\Delta \\
&< \sum_{\pi\in\mathbf{\Pi}_\alpha}\Pr[\Pi=\pi]\left(q_{j^*(\pi)}(\mathcal{Q}_\pi)-q_{j^*(\pi)}(\mathcal{P}^n)\right) \\
&= \mathop{\mathbb{E}}_{S\sim\mathcal{P}^n}\left[\sum_{\pi\in\mathbf{\Pi}_\alpha}\Pr_{\Pi\sim I(S)}[\Pi=\pi]\left(q_{j^*(\pi)}(S)-q_{j^*(\pi)}(\mathcal{P}^n)\right)\right] \\
&= \sum_{i=1}^{n}\mathop{\mathbb{E}}_{S\sim\mathcal{P}^n}\left[\sum_{\pi\in\mathbf{\Pi}_\alpha}\Pr_{\Pi\sim I(S)}[\Pi=\pi]\left(\bar{q}_{j^*(\pi)}(S_{\le i})-\bar{q}_{j^*(\pi)}(S_{\le i-1})\right)\right] \\
&\le ((e^\epsilon-1)\Pr[\Pi\in\mathbf{\Pi}_\alpha]+2\delta)\,n\Delta \quad \text{(by Equation (2))} \\
&< \Pr[\Pi\in\mathbf{\Pi}_\alpha]\,(e^\epsilon-1+4c)\,n\Delta \quad \text{(by Equation (3))}
\end{aligned}$$

◀

We can combine this Lemma with Lemma 6 (which holds for any query type) to get our transfer theorem:

▶ **Theorem 16** (Transfer Theorem for Low Sensitivity Queries). *Suppose that M is (ϵ,δ)-differentially private and (α,β)-sample accurate for Δ-sensitive queries. Then for every analyst $\mathcal{A}$, $c,d>0$ it also satisfies:*

$$\Pr_{S\sim\mathcal{P}^n,\,\Pi\sim Interact(M,\mathcal{A};S)}\left[\max_j |a_j - q_j(\mathcal{P}^n)| > \alpha + c + (e^\epsilon-1+4d)n\Delta\right] \le \frac{\beta}{c}+\frac{\delta}{d}$$

i.e. it is (α',β')-distributionally accurate for $\alpha' = \alpha + c + (e^\epsilon-1+4d)n\Delta$ and $\beta' = \frac{\beta}{c}+\frac{\delta}{d}$.

A.2 Minimization Queries

▶ **Definition 17.** *Minimization queries are specified by a loss function $L : \mathcal{X}^n \times \Theta \to [0,1]$ where Θ is generally known as the "parameter space". An answer to a minimization query L is a parameter $\theta\in\Theta$. We work with Δ-sensitive minimization queries: for all pairs of neighbouring datasets $S, S'\in\mathcal{X}^n$ and all $\theta\in\Theta$, $|L(S,\theta) - L(S',\theta)| \le \Delta$.*

A mechanism M is (α,β)-sample accurate for minimization queries if for every data analyst $\mathcal{A}$ and every dataset $S\in\mathcal{X}^n$:

$$\Pr_{\Pi\sim Interact(M,\mathcal{A};S)}\left[\max_j\left|L_j(S,\theta_j) - \min_{\theta\in\Theta}L_j(S,\theta)\right| \ge \alpha\right] \le \beta$$

We say that M satisfies (α,β)-distributional accuracy for minimization queries if for every data analyst $\mathcal{A}$ and every data distribution $\mathcal{P}$:

$$\Pr_{S\sim\mathcal{P}^n,\Pi\sim Interact(M,A;S)}\left[\max_j\left|\mathop{\mathbb{E}}_{S'\sim\mathcal{P}^n}\left[L_j(S',\theta_j) - \min_{\theta\in\Theta}L_j(S',\theta)\right]\right| \ge \alpha\right] \le \beta$$

▶ Remark 18. Note that

$$\mathop{\mathbb{E}}_{S'\sim\mathcal{P}^n}[L_j(S',\theta_j)] - \min_{\theta\in\Theta}\mathop{\mathbb{E}}_{S'\sim\mathcal{P}^n}[L_j(S',\theta)] \leq \mathop{\mathbb{E}}_{S'\sim\mathcal{P}^n}\left[L_j(S',\theta_j) - \min_{\theta\in\Theta} L_j(S',\theta)\right]$$

So as long as the RHS is bounded, the LHS is bounded too.

▶ Remark 19. For a given Δ-sensitive minimization query L_j and an answer θ_j, define:

$$q_j(S) := L_j(S,\theta_j) - \min_{\theta\in\Theta} L_j(S,\theta) \quad \text{and} \quad a_j := 0$$

Note several things:

1. If L_j is Δ-sensitive, then q_j is 2Δ-sensitive.
2. The mapping from a minimization query transcript $\pi = ((L_1,\theta_1),\ldots,(L_k,\theta_k))$ to the 2Δ-sensitive query transcript $\pi' = ((q_1,a_1),\ldots,(q_k,a_k))$ as defined above is a dataset-independent post-processing $\pi' = f(\pi)$.
3. π satisfies an (α,β)-accuracy guarantee if and only if π' does.

With the above observation, the transfer theorem for minimization queries immediately follows by Lemma 15 and Lemma 6.

▶ **Theorem 20** (Transfer Theorem for Minimization Queries). *Suppose that M is (ϵ,δ)-differentially private and (α,β)-sample accurate for Δ-sensitive minimization queries. Then for every analyst $\mathcal{A}$ and $c,d>0$ it also satisfies:*

$$\Pr\left[\max_j \left|\mathop{\mathbb{E}}_{S'\sim\mathcal{P}^n}\left[L_j(S',\theta_j) - \min_{\theta\in\Theta} L_j(S',\theta)\right]\right| > \alpha + c + 2(e^\epsilon - 1 + 4d)n\Delta\right] \leq \frac{\beta}{c} + \frac{\delta}{d}$$

i.e. it is (α',β')-distributionally accurate for $\alpha' = \alpha + c + 2(e^\epsilon - 1 + 4d)n\Delta$ and $\beta' = \frac{\beta}{c} + \frac{\delta}{d}$.

B Details from Section 3.1

Proof of Lemma 5. This follows from the expansion of the definition, and an application of Bayes Rule.

$$\begin{aligned}
&\Pr_{S\sim\mathcal{P}^n,\Pi\sim I(S),S'\sim\mathcal{Q}_\Pi}[(S',\Pi)\in E]\\
&= \sum_{\boldsymbol{x}}\sum_{\pi}\sum_{\boldsymbol{x}'} \Pr[S=\boldsymbol{x}]\Pr[\Pi=\pi|S=\boldsymbol{x}]\Pr_{S'\sim\mathcal{Q}_\pi}[S'=\boldsymbol{x}']\mathbb{1}[(\boldsymbol{x}',\pi)\in E]\\
&= \sum_{\pi}\sum_{\boldsymbol{x}'} \Pr[\Pi=\pi]\Pr_{S'\sim\mathcal{Q}_\pi}[S'=\boldsymbol{x}']\mathbb{1}[(\boldsymbol{x}',\pi)\in E]\\
&= \sum_{\pi}\sum_{\boldsymbol{x}'} \Pr[\Pi=\pi]\Pr[S=\boldsymbol{x}'|\Pi=\pi]\mathbb{1}[(\boldsymbol{x}',\pi)\in E]\\
&= \sum_{\pi}\sum_{\boldsymbol{x}'} \Pr[\Pi=\pi]\frac{\Pr[\Pi=\pi|S=\boldsymbol{x}']\cdot\Pr[S=\boldsymbol{x}']}{\Pr[\Pi=\pi]}\mathbb{1}[(\boldsymbol{x}',\pi)\in E]\\
&= \Pr_{S\sim\mathcal{P}^n,\Pi\sim I(S)}[(S,\Pi)\in E]
\end{aligned}$$

◀

C Details from Section 3.2

▶ **Lemma 21.** *If M is (ϵ,δ)-differentially private, then for any event E and datapoint x:*

$$\Pr_{S\sim\mathcal{P}^n,S_i\sim S,\Pi\sim I(S)}[\Pi\in E|S_i=x] \leq e^\epsilon \Pr_{S\sim\mathcal{P}^n,\Pi\sim I(S)}[\Pi\in E] + \delta$$

Proof. This follows from expanding the definitions.

$$\begin{aligned}
\Pr_{S\sim\mathcal{P}^n, S_i\sim S, \Pi\sim I(S)}[\Pi \in E | S_i = x] &= \frac{1}{n}\sum_{i=1}^{n} \Pr_{S\sim\mathcal{P}^n, \Pi\sim I(S)}[\Pi \in E | S_i = x] \\
&= \frac{1}{n}\sum_{i=1}^{n}\sum_{\boldsymbol{x}\in\mathcal{X}^n} \Pr_{S\sim\mathcal{P}^n}[S = \boldsymbol{x}] \cdot \Pr[\Pi \in E | S = (\boldsymbol{x}_{-i}, x)] \\
&\leq \frac{1}{n}\sum_{i=1}^{n}\sum_{\boldsymbol{x}\in\mathcal{X}^n} \Pr_{S\sim\mathcal{P}^n}[S = \boldsymbol{x}] \cdot (e^{\epsilon}\Pr[\Pi \in E | S = \boldsymbol{x}] + \delta) \\
&= e^{\epsilon} \Pr_{S\sim\mathcal{P}^n, \Pi\sim I(S)}[\Pi \in E] + \delta
\end{aligned}$$

where the inequality follows from the definition of differential privacy. ◀

D An (even) Simpler and Better Proof for ϵ-Differential Privacy

In this section we give an *even simpler* proof of an *even better* transfer theorem for $(\epsilon, 0)$-differential privacy. Rather than using Markov's inequality as we did in the proof of Lemma 6, we can directly show that conditional distributions induced by differentially private mechanisms exhibit Chernoff-like concentration.

▶ **Lemma 22.** *If M is $(\epsilon, 0)$-differentially private, then for any data distribution $\mathcal{P}$, any transcript $\pi \in \mathbf{\Pi}$, any linear query q, and any $\eta > 0$:*

$$\Pr_{S\sim\mathcal{Q}_\pi}\left[|q(S) - q(\mathcal{P})| \geq (e^{\epsilon} - 1) + \sqrt{\frac{2\ln(2/\eta)}{n}}\right] \leq \eta$$

Proof. Define the random variables $V_i = q(S_i) - \mathbb{E}[q(S_i)|S_{<i}]$, and let $X_i = \frac{1}{n}\sum_{j=1}^{i} V_j$. Then the sequence $0 = X_0, X_1, \ldots, X_n$ forms a martingale and $|X_i - X_{i-1}| = \frac{1}{n}|V_i| \leq \frac{1}{n}$. We can therefore apply Azuma's inequality to conclude that:

$$\Pr\left[\left|\frac{1}{n}\sum_{i=1}^{n} q(S_i) - \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}[q(S_i)|S_{<i}]\right| \geq t\right] \leq 2\exp\left(\frac{-t^2 n}{2}\right) \tag{4}$$

Now fix any realization $\boldsymbol{x}$, and consider each term: $\mathbb{E}[q(S_i)|S_{<i} = \boldsymbol{x}_{<i}]$. We have:

$$\begin{aligned}
\mathop{\mathbb{E}}_{S\sim\mathcal{Q}_\pi}[q(S_i)|S_{<i} = \boldsymbol{x}_{<i}] &= \sum_x q(x) \cdot \Pr_{S\sim\mathcal{P}^n}[S_i = x | \Pi = \pi, S_{<i} = \boldsymbol{x}_{<i}] \\
&= \sum_x q(x) \cdot \frac{\Pr_{S\sim\mathcal{P}^n}[\Pi = \pi | S_i = x, S_{<i} = \boldsymbol{x}_{<i}] \cdot \Pr_{S\sim\mathcal{P}^n}[S_i = x]}{\Pr[\Pi = \pi | S_{<i} = \boldsymbol{x}_{<i}]} \\
&\leq e^{\epsilon} \cdot \sum_x q(x) \cdot \Pr_{S\sim\mathcal{P}^n}[S_i = x] \\
&= e^{\epsilon} q(\mathcal{P})
\end{aligned}$$

where the inequality follows from the definition of $(\epsilon, 0)$-differential privacy. Symmetrically, we can show that $\mathbb{E}_{S\sim\mathcal{Q}_\pi}[q(S_i)|S_{<i} = \boldsymbol{x}_{<i}] \geq e^{-\epsilon} q(\mathcal{P})$. Therefore we have that:

$$e^{-\epsilon} q(\mathcal{P}) \leq \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}[q(S_i)|S_{<i}] \leq e^{\epsilon} q(\mathcal{P}).$$

Combining this with Equation 4 gives us that for any $\eta > 0$, with probability $1 - \eta$ when $S \sim \mathcal{Q}_\pi$:

$$q(S) \leq e^{\epsilon} q(\mathcal{P}) + \sqrt{\frac{2\ln(2/\eta)}{n}} \quad \text{and} \quad q(S) \geq e^{-\epsilon} q(\mathcal{P}) - \sqrt{\frac{2\ln(2/\eta)}{n}}$$ ◀

A transfer theorem follows immediately from lemma 22.

▶ **Theorem 23.** *Suppose that M is $(\epsilon, 0)$-differentially private and (α, β)-sample accurate. Then for any $\eta > 0$ it is (α', β')-distributionally accurate for $\alpha' = \alpha + (e^\epsilon - 1) + \sqrt{\frac{2\ln(2/\eta)}{n}}$ and $\beta' = \beta + \eta$.*

Proof. For a given π, let $j^*(\pi) = \arg\max_j |a_j - q_j(\mathcal{P})|$. By the triangle inequality we have:

$$\begin{aligned}|a_{j^*(\Pi)} - q_{j^*(\Pi)}(\mathcal{P})| &\leq |a_{j^*(\Pi)} - q_{j^*(\Pi)}(S)| + |q_{j^*(\Pi)}(S) - q_{j^*(\Pi)}(\mathcal{P})| \\ &\leq \max_j |a_j - q_j(S)| + |q_{j^*(\Pi)}(S) - q_{j^*(\Pi)}(\mathcal{P})|\end{aligned}$$

By the definition of (α, β)-sample accuracy, we have that with probability $1 - \beta$, $\max_j |a_j - q_j(S)| \leq \alpha$. The Resampling Lemma (Lemma 5) gives us that:

$$\begin{aligned}&\Pr_{S\sim\mathcal{P}^n,\, \Pi\sim I(S)}\left[|q_{j^*(\Pi)}(S) - q_{j^*(\Pi)}(\mathcal{P})| \geq (e^\epsilon - 1) + \sqrt{\frac{2\ln(2/\eta)}{n}}\right] \\ &= \Pr_{S\sim\mathcal{P}^n,\, \Pi\sim I(S),\, S'\sim\mathcal{Q}_\Pi}\left[|q_{j^*(\Pi)}(S') - q_{j^*(\Pi)}(\mathcal{P})| \geq (e^\epsilon - 1) + \sqrt{\frac{2\ln(2/\eta)}{n}}\right] \\ &= \mathop{\mathbb{E}}_{S\sim\mathcal{P}^n,\, \Pi\sim I(S)}\left[\Pr_{S'\sim\mathcal{Q}_\Pi}\left[|q_{j^*(\Pi)}(S') - q_{j^*(\Pi)}(\mathcal{P})| \geq (e^\epsilon - 1) + \sqrt{\frac{2\ln(2/\eta)}{n}}\right]\right] \\ &\leq \eta\end{aligned}$$

Because Lemma 22 guarantees us that for *every* π,

$$\Pr_{S'\sim\mathcal{Q}_\pi}\left[|q_{j^*(\pi)}(S') - q_{j^*(\pi)}(\mathcal{P})| \geq (e^\epsilon - 1) + \sqrt{\frac{2\ln(2/\eta)}{n}}\right] \leq \eta.$$

The theorem then follows from a union bound. ◀

Robust Algorithms for the Secretary Problem

Domagoj Bradac
ETH Zurich, Switzerland
University of Zagreb, Croatia
dbradac@student.ethz.ch

Anupam Gupta
Carnegie Mellon University, Pittsburgh, PA, USA
anupamg@cs.cmu.edu

Sahil Singla
Princeton University, Princeton, NJ, USA
Institute for Advanced Study, Princeton, NJ, USA
singla@cs.princeton.edu

Goran Zuzic
Carnegie Mellon University, Pittsburgh, PA, USA
gzuzic@cs.cmu.edu

Abstract

In classical secretary problems, a sequence of n elements arrive in a uniformly random order, and we want to choose a single item, or a set of size K. The random order model allows us to escape from the strong lower bounds for the adversarial order setting, and excellent algorithms are known in this setting. However, one worrying aspect of these results is that the algorithms overfit to the model: they are not very robust. Indeed, if a few "outlier" arrivals are adversarially placed in the arrival sequence, the algorithms perform poorly. E.g., Dynkin's popular $1/e$-secretary algorithm is sensitive to even a single adversarial arrival: if the adversary gives one large bid at the beginning of the stream, the algorithm does not select any element at all.

We investigate a robust version of the secretary problem. In the *Byzantine Secretary* model, we have two kinds of elements: green (good) and red (rogue). The values of all elements are chosen by the adversary. The green elements arrive at times uniformly randomly drawn from $[0, 1]$. The red elements, however, arrive at adversarially chosen times. Naturally, the algorithm does not see these colors: how well can it solve secretary problems?

We show that selecting the highest value red set, or the single largest green element is not possible with even a small fraction of red items. However, on the positive side, we show that these are the only bad cases, by giving algorithms which get value comparable to the value of the optimal green set *minus the largest green item*. (This benchmark reminds us of regret minimization and digital auctions, where we subtract an additive term depending on the "scale" of the problem.) Specifically, we give an algorithm to pick K elements, which gets within $(1 - \varepsilon)$ factor of the above benchmark, as long as $K \geq \text{poly}(\varepsilon^{-1} \log n)$. We extend this to the knapsack secretary problem, for large knapsack size K.

For the single-item case, an analogous benchmark is the value of the second-largest green item. For value-maximization, we give a $\text{poly} \log^* n$-competitive algorithm, using a multi-layered bucketing scheme that adaptively refines our estimates of second-max over time. For probability-maximization, we show the *existence* of a good randomized algorithm, using the minimax principle.

We hope that this work will spur further research on robust algorithms for the secretary problem, and for other problems in sequential decision-making, where the existing algorithms are not robust and often tend to overfit to the model.

2012 ACM Subject Classification Theory of computation → Adversary models; Theory of computation → Streaming models; Theory of computation → Design and analysis of algorithms; Theory of computation → Packing and covering problems; Theory of computation → Algorithmic game theory and mechanism design

Keywords and phrases stochastic optimization, robust optimization, secretary problem, matroid secretary, robust secretary

11th Innovations in Theoretical Computer Science Conference (ITCS 2020).
Editor: Thomas Vidick; Article No. 32; pp. 32:1–32:26

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.32

Funding *Domagoj Bradac*: Part of this work was done when visiting the Computer Science Department at Carnegie Mellon University.
Anupam Gupta: Supported in part by NSF award CCF-1907820.
Sahil Singla: Supported in part by the Schmidt Foundation.
Goran Zuzic: Supported in part by NSF grants CCF-1527110, CCF-1618280, CCF-1814603, CCF-1910588, NSF CAREER award CCF-1750808, Sloan Research Fellowship and the DFINITY 2018 award.

Acknowledgements We thank Thomas Kesselheim and Marco Molinaro for sharing their model and thoughts on robust secretary problems with us; these have directly inspired our model.

1 Introduction

In sequential decision-making, we have to serve a sequence of requests *online*, i.e., we must serve each request before seeing the next one. E.g., in online auctions and advertising, given a sequence of arriving buyers, we want to choose a high bidder. Equivalently, given a sequence of n numbers, we want to choose the highest of these. The worst-case bounds for this problem are bleak: choosing a random buyer is the best we can do. So we make (hopefully reasonable) stochastic assumptions about the input stream, and give algorithms that work well under those assumptions.

A popular assumption is that the values/bids are chosen by an adversary, but presented to the algorithm in a uniformly random order. This gives the *secretary* or the *random-order model*, under which we can get much better results. E.g., Dynkin's secretary algorithm that selects the first prefix-maximum bidder after discarding the first $1/e$-fraction of the bids, selects the highest bid with probability $1/e$ [13]. The underlying idea – of fixing one or more thresholds after seeing some prefix of the elements – can be generalized to solve classes of packing linear programs near-optimally [8, 9, 25, 21], and to get $O(\log \log n)$-competitive algorithms for matroids [28, 16] in the random-order model.

However, the assumption that we see the elements in a uniformly random order is quite strong, and most current algorithms are not robust to small perturbations to the model. E.g., Dynkin's algorithm is sensitive to even a single adversarial corruption: if the adversary gives one large bid at the beginning of the stream, the algorithm does not select any buyer at all, even if the rest of the stream is perfectly random! Many other algorithms in the secretary model suffer from similar deficiencies, which suggests that we may be over-fitting to the assumptions of the model.

We propose the *Byzantine secretary model*, where the goal is to design algorithms robust to outliers and adversarial changes. The use of the term "Byzantine" parallels its use in distributed systems, where some of the input is well-behaved while the rest is arbitrarily corrupted by an adversary. Alternatively, our model can be called *semi-random* or *robust*: these other terms are used in the literature which inspires our work. Indeed, there is much interest currently in designing stochastic algorithms that are robust to adversarial noise (see [10, 34, 29, 3, 34, 11, 12, 14, 30] and references therein). Our work seeks to extend robustness to online problems. Our work is also related in spirit to investigations into how much randomness in the stream is necessary and sufficient to get competitive algorithms [5, 23].

1.1 Our Model

In the secretary problem, n elements arrive one-by-one. Each item has a value that is revealed upon its arrival, which happens at a time chosen independently and *uniformly at random* in $[0, 1]$. (We choose the continuous time model, instead of the *uniformly random* arrival order model, since the independence allows us to get clean proofs.) When we see an item, we must either *select* it or discard it before we see the next item. Our decisions are *irrevocable*. We can select at most K elements, where $K = 1$ for the classical version of the problem. We typically want to maximize the expected total value of the selected elements where the value of a set is simply the sum of values of individual elements. (For the single-item case we may also want to maximize the probability of selecting the highest-value item, which is called the *ordinal case*.) Given its generality and wide applicability, this model and its extensions are widely studied; see §1.3.

The difference between the classical and Byzantine secretary models is in how the sequence is generated. In both models, the adversary chooses the values of all n elements. In the classical model, these are then permuted in a random order (say by choosing the arrival times independently and uniformly at random (u.a.r.) from $[0, 1]$). In the Byzantine model, the elements are divided into two groups: the *green (or good)* elements/items G, and the *red (or rogue/bad)* elements/items R. This partition and the colors are not visible to the algorithm. Now elements in G arrive at independently chosen u.a.r. times between $[0, 1]$, but those in R arrive at times chosen by the adversary. Faced with this sequence, the algorithm must select some subset of elements (say, having size at most K, or more generally belonging to some down-closed family). The precise order of actions is important:

- First, the adversary chooses values of elements in $R \cup G$, and the arrival times of elements in R.
- Then each element $e \in G$ is independently assigned a uniformly random arrival time $t_e \sim U[0, 1]$.

Hence the adversary is powerful and strategic, and can "stuff" the sequence with values in an order that fools our algorithms the most. The green elements are non-strategic (hence are in random order) and beyond the adversary's control. When an element is presented, the algorithm does not see the color (green vs. red): it just sees the value and the time of arrival. We assume that the algorithm knows $n := |R| + |G|$, but not $|R|$ or $|G|$; see Appendix B on how to relax this assumption. The green elements are denoted $G = \{g_{\max} = g_1, g_2, \ldots, g_{|G|}\}$ in non-increasing order of values.

What results can we hope to get in this model? Here are two cautionary examples:

- Since the red elements behave completely arbitrarily, the adversary can give non-zero values to only the reds, and plant a bad example for the adversarial order using them. Hence, we cannot hope to get the value of the optimal red set in general, and should aim to get value only from the greens.
- Moreover, suppose essentially all the value among the greens is concentrated in a single item $g_{\max}$. Here's a bad example: the adversary gives a sequence of increasing reds, all having value much smaller than $g_{\max}$, but values which are very far from each other. When the algorithm does see the green item, it will not be able to distinguish it from the next red, and hence will fail. This is formalized in Observation 19. Hence, to succeed, the green value must be spread among more than one item.

Given these examples, here is the "leave-one-out" benchmark we propose:

$$\boxed{V^* := \text{value of the best feasible green set from } G \setminus g_{\max}.} \tag{1}$$

This benchmark is at least as strong as the following guarantee:

$$(\text{value of best feasible green set from } G) \; - v(g_{\max}). \tag{2}$$

The advantage of (1) over (2) is that V^* is interesting even when we want to select a single item, since it asks for value v_{g_2} or higher.

We draw two parallels to other commonly used benchmarks. Firstly, the perspective (2) suggests the regret-type guarantees, where we seek the best solution in hindsight, minus the "scale of the problem instance". The value of $g_{\max}$ is the scale of the instance here. Secondly, think of the benchmark (1) as assuming the existence of at least two high bids, then the second-largest element is almost as good a benchmark as the top element. This is a commonly used assumption, e.g., in digital goods auctions [4].

Finally, if we really care about a benchmark that includes $g_{\max}$, our main results for selecting multiple items (Theorem 1 and Theorem 2) continue to hold, under the (mild?) assumption that the algorithm starts with a polynomial approximation to $v(g_{\max})$.

1.2 Our Results

We first consider the setting where we want to select at most K elements to maximize the expected total value. In order to get within $(1+\varepsilon)$ factor of the benchmark V^* defined in (1), we need to assume that we have a "large budget", i.e., we are selecting a sufficiently large number of elements. Indeed, having a larger budget K allows us to make some mistakes and yet get a good expected value.

▶ **Theorem 1** (Uniform Matroids). *There is an algorithm for Byzantine secretary on uniform matroids of rank $K \geq \text{poly}(\varepsilon^{-1} \log n)$ that is $(1+\varepsilon)$-competitive with the benchmark V^*.*

For the standard version of the problem, i.e. without any red elements, [26] gave an algorithm that achieves the same competitiveness when $K \geq \Omega(1/\varepsilon^2)$. The algorithm from [26] uses a single threshold, that it updates dynamically; we extend this idea to having several thresholds/budgets that "cascade down" over time; we sketch the main ideas in §2.2. In fact, we give a more general result – an algorithm for the *knapsack* setting where each element has a size in $[0, 1]$, and the total size of elements we can select is at most K. (The uniform-matroids case corresponds to all sizes being one.) Whereas the main intuition remain unchanged, the presence of non-uniform sizes requires a little more care.

▶ **Theorem 2** (Knapsack). *There is an algorithm for Byzantine secretary on knapsacks with size at least $K \geq \text{poly}(\varepsilon^{-1} \log n)$ (and elements of at most unit size) that is $(1+\varepsilon)$-competitive with the benchmark V^*.*

As mentioned earlier, under mild assumptions the guarantee in Theorem 2 can be extended against the stronger benchmark that includes $g_{\max}$. Formally, assuming the algorithm starts with a $\text{poly}(m)$-approximation to the value of $g_{\max}$, we get a $(1+\varepsilon)$-competitive algorithm for $K \geq \text{poly}(\varepsilon^{-1} \log(mn))$ against the stronger benchmark.

Selecting a Single Item. What if we want to select a single item, to maximize its expected value? Note that the benchmark V^* is now the value of g_2, the second-largest green item. Our main result for this setting is the following, where $\log^* n$ denotes the iterated logarithm:

▶ **Theorem 3** (Value Maximization Single-Item). *There is a randomized algorithm for the value-maximization (single-item) Byzantine secretary problem which gets an expected value at least $(\log^* n)^{-2} \cdot V^*$.*

Interestingly, our result is unaffected by the corruption level, and works even if just two elements $g_{\max}, g_2$ are green, and every other item is red. This is in contrast to many other robustness models where the algorithm's performance decays with the fraction of bad elements [14, 3, 12, 30]. Moreover, our algorithms do not depend on the fraction of bad items. Intuitively, we obtain such strong guarantees because the adversary has no incentive to present too many bad elements with large values, as otherwise an algorithm that selects a random element would have a good performance.

In the classical setting, the proofs for the value-maximization proceed via showing that the best item itself is chosen with constant probability. Indeed, in that setting, the competitiveness of value-maximization and probability-maximization versions is the same. We do not know of such a result in the Byzantine model. However, we can show a non-trivial performance for the probability-maximization (ordinal) problem:

▶ **Theorem 4** (Ordinal Single-item Algorithm). *There is a randomized algorithm for the ordinal Byzantine secretary which selects an element of value at least the second-largest green item with probability* $\Omega(1/\log^2 n)$.

Other Settings. Finally, we consider some other constraint sets given by matroids. In (simple) partition matroids, the universe U is partitioned into r groups, and the goal is to select one item from each group to maximize the total value. If we were to set the benchmark to be the sum of second-largest green items from each group, we can just run the single-item algorithm from Theorem 1 on each group independently. But our benchmark V^* is much higher: among the items $g_2, \ldots, g_{|G|}$, the set V^* selects the largest one from each group. Hence, we need to get the largest green item from $r-1$ groups! Still, we do much better than random guessing.

▶ **Theorem 5** (Partition Matroids). *There is an algorithm for Byzantine secretary on partition matroids that is* $O(\log\log n)^2$*-competitive with the benchmark* V^*.

Finally, we record a simple but useful logarithmic competitive ratio for arbitrary matroids (proof in §6.2), showing how to extend the corresponding result from [1] for the non-robust case.

▶ **Observation 6** (General Matroids). *There is an algorithm for Byzantine secretary on general matroids that is* $O(\log n)$*-competitive with the benchmark* V^*.

Our results show how to get robust algorithms for the widely-studied secretary problems, and we hope it will generate futher interest in robust algorithm design. Interesting next directions include improving the quantitative bounds in our results (which are almost certainly not optimal), and understanding tradeoffs between competitiveness and robustness.

1.3 Related Work

The secretary problem has a long history, see [17] for a discussion. The papers [1, 28, 16] studied generalizations of the secretary problem to matroids, [20, 27, 24, 22] studied extensions to matchings, and [38, 39] studied extensions to arbitrary packing constraints. More generally, the random-order model has been considered, both as a tool to speed up algorithms (see [6, 41]), and to go beyond the worst-case in online algorithms (see [32, 18, 19]). E.g., we can solve *linear programs* online if the columns of the constraint matrix arrive in a random order [8, 9, 25, 21], and its entries are small compared to the bounds. In online algorithms, the random-order model provides one way of modeling benign nature, as opposed to an adversary hand-crafting a worst-case input sequence; this model is at least as general as i.i.d. draws from an unknown distribution.

Both the random-order model and the Byzantine model are *semi-random* models [2, 15], with different levels of power to the adversary. Other restrictions of the random-order model have been studied, via lower-bounding the entropy of the input stream, to ensure the permutations are "random enough" [5, 23]. These papers give sufficient conditions for the classical algorithms to perform well, whereas we take the dual approach of permitting outliers and then asking for new robust algorithms. There are works (e.g., [31, 33, 35]) that give algorithms which have a worst-case adversarial bound, and which work better when the input is purely stochastic; most of these do not study the performance on mixed arrival sequences. One exception is the work [14] who study online matching for mixed arrivals, under the assumption that the "magnitude of noise" is bounded. Another exception is a recent (unpublished) work of Kesselheim and Molinaro, who define a robust K-secretary problem similar to ours. They assume the corruptions have a bursty pattern, and get $1 - f(K)$-competitive algorithms. Our model is directly inspired by theirs.

2 Preliminaries and Techniques

By $[a \dots b]$ we denote the set of integers $\{a, a+1, \dots, b-1, b\}$. The *universe* $U := R \cup G$ consists of *red/corrupted* elements R and *greed/good* elements G. Let $v(e)$ denote the value of element e: in the *ordinal* case $v(e)$ merely defines a total ordering on the elements, whereas in the *value-maximization* case $v(e) \in \mathbb{R}_{\geq 0}$. Similarly, let $v(\mathcal{A})$ be the random variable denoting the value of the elements selected by algorithm $\mathcal{A}$. Let $t_e \in [0, 1]$ be the arrival time of element e. Let $R = \{r_1, r_2, \dots, r_{|R|}\}$ and $G = \{g_{\max}, g_2, g_3, \dots, g_{|G|}\}$; the elements in each set are ordered in non-increasing values. Let V^* be the benchmark to which we compare our algorithm. Note that V^* is some function of $G \setminus \{g_{\max}\}$, depending on the setting. We sometimes refer to elements $\{e \in U \mid v(e) \geq v(g_2)\}$ as *big*.

2.1 Two Useful Subroutines

Here are two useful subroutines.

Select a Random Element. The subroutine is simple: *select an element uniformly at random*. The algorithm can implement this in an online fashion since it knows the total number of elements n. An important property of this subroutine is that, in the value case, if any element in U has value at least nV^*, this subroutine gets at least V^* in expectation since this highest value element is selected with probability $1/n$.

Two-Checkpoints Secretary. The subroutine is defined on two checkpoints $T_1, T_2 \in [0, 1]$, and let $\mathbb{I} := [T_1, T_2]$ be the interval between them. The subroutine ignores the input up to time T_1, observes it during $\mathbb{I}$ by setting threshold τ to be the highest value seen in the interval $\mathbb{I}$, i.e., $\tau \leftarrow \max\{v(e) \mid t_e \in \mathbb{I}\}$. Finally, during $\langle T_2, 1]$ the subroutine selects the first element e with value $v(e) \geq \tau$.

We use the subroutine in the single-item setting where the goal is to find a "big" element, i.e., an element with value at least $v(g_2)$. Suppose that there are no big red elements in $\mathbb{I}$. Now, if g_2 lands in $\mathbb{I}$, and also $g_{\max}$ lands in $\langle T_2, 1]$, we surely select some element with value at least g_2. Indeed, if there are no big items, threshold $\tau \leftarrow g_2$, and because $g_{\max}$ lands after $\mathbb{I}$, it or some other element will be selected. Hence, with probability $\Pr[t_{g_2} \in \mathbb{I}] \Pr[t_{g_{\max}} \in \langle T_2, 1]] = (T_2 - T_1) \cdot (1 - T_2)$, we select an element of value at least $v(g_2)$.

2.2 Our Techniques

A common theme of our approaches is to prove a "good-or-learnable" lemma for each problem. Our algorithms begin by putting down a small number of *checkpoints* $\{T_i\}_i$ to partition the time horizon $[0, 1]$ – and the arriving items – into disjoint *intervals* $\{\mathbb{I}_i\}_i$. We maintain *thresholds* in each interval to decide whether to select the next element. Now a "good-or-learnable" lemma says that either the setting of the thresholds in the current interval $\mathbb{I}_i$ will give us a "good" performance, or we can "learn" that this is not the case and update the thresholds for the next interval $\mathbb{I}_{i+1}$. Next we give details for each of our problems.

Uniform Matroid Value Maximization (§3). Recall that here we want to pick K elements (in particular, all elements have size 1, unlike the knapsack case where sizes are in the range $[0, 1]$). For simplicity, suppose the algorithm knows that the benchmark V^* lies in $[1, n]$; we remove this assumption later. We define $O(\varepsilon^{-1} \log n)$ levels, where level $\ell \geq 0$ corresponds to values in the range $[n/(1+\varepsilon)^{\ell+1}, n/(1+\varepsilon)^{\ell})$. For each interval $\mathbb{I}_i$ and level ℓ, we maintain a budget $B_{\ell,i}$. Within this interval $\mathbb{I}_i$, we select the next arriving element having a value in some level ℓ only if the budget $B_{\ell,i}$ has not been used up. How should we set these budgets? If there are $1/\delta$ intervals of equal size, we expect to select δK elements in this interval. So we have a total of δK budget to distribute among the various levels. We start off optimistically, giving all the budget to the highest-value level. Now this budget gradually *cascades* from a level ℓ to the next (lower-value) level $\ell + 1$, if level ℓ is not selecting elements at a "good enough" rate. The intuition is that for the "heavy" levels (i.e., those that contain many elements from the benchmark-achieving set S^*), we will roughly see the right number of them arriving in each interval. This allows us to prove a good-or-learnable lemma, that either we select elements at a "good enough" rate in the current interval, or this is not the case and we "learn" that the budgets should cascade to lower value levels. There are many details to be handled: e.g., this reasoning is only possible for levels with many benchmark elements, and so we need to define a dedicated budget to handle the "light" levels.

Single-Item Value-Maximization (§5). We want to maximize the expected value of the selected element, compared to $V^* := v(g_2)$, the value of the second-max green. With some small constant probability our algorithm selects a uniformly random element. This allows us to assume that every element has value less than nV^*, as otherwise the expected value of a random guess is $\Omega(V^*)$. We now describe how applying the above "good-or-learnable" paradigm in a natural way guarantees an expected value of $\Omega(V^*/\log n)$. Running the two-checkpoint secretary (with constant probability) during $T_1 = 0, T_2 = 1/2$ we know that it gets value $\Omega(V^*)$ and we are done, or failing that, there exist a red element of value at least V^* in $[0, 1/2]$. But then we can use this red element (highest value in the first half) to get a factor n estimate on the value of V^*. So by grouping elements into buckets if their values are within a factor 2, and randomly guessing the bucket that contains $v(g_2)$, gives us an expected value of $\Omega(V^*/\log n)$. To obtain the stronger factor of $\operatorname{poly}\log^* n$ in Theorem 3, we now define $\log^* n$ checkpoints. We prove a "good-or-learnable" lemma that either selecting a random element from one of the current buckets has a good value, or we can learn a tighter estimate on V^* and reduce the number of buckets.

Ordinal Single-Item Secretary (§4). We now want to maximize the probability of selecting an element whose value is as large as the green second-max; this is more challenging than value-maximization since there is no notion of values for bucketing. Our approach is crucially different. Indeed, we use the *minimax principle* in the "hard" direction: we give an algorithm

that does well when the input distribution is *known to the algorithm* (i.e., where the algorithm can adapt to the distribution), and hence infer the existence of a (randomized) algorithm that does well on worst-case inputs.

The known-distribution algorithm uses $O(\log n)$ intervals. Again, we can guarantee there is a "big" (larger than g_2) red element within each interval, as otherwise running Dynkin's algorithm on a random interval with a small probability already gives a "good" approximation. This implies that even if the algorithm "learns" a good estimate of the second-max just before the last interval, it will win. This is because the algorithm can set this estimate of second-max as a threshold, and it wins by selecting the big red element of the last interval. Finally, to learn a good estimate on the second-max, we again prove a "good-or-learnable" lemma. Its proof crucially relies on the algorithm knowing the arrival distribution, since that allows us to set "median" of the conditional distribution as a threshold.

Other Results (§6). We also give $O(\log \log n)^2$-competitive algorithms for Partition matroids, where the difficulty is that we cannot afford to lose the max-green element in every part. Our idea is to only lose one element globally to get a very rough scale of the problem, and then exploit this scale in every part. We also show why other potential benchmarks are too optimistic in §A, and how to relax the assumption that n is known in §B. See those sections for details.

3 Knapsack Byzantine Secretary

Consider a knapsack of size K; for all the results in this section we assume that $K \geq \text{poly}(\varepsilon^{-1} \log n)$. Each arriving element e has a size $s(e) \in [0, 1]$ and a value $v(e) \geq 0$. Let $g_{\max}, g_2, g_3, \ldots,$ denote the green elements G with decreasing values and let

$$V^* := \max \left\{ \textstyle\sum_{e \in S} v(e) \mid S \subseteq G \setminus g_{\max} \text{ and } \sum_{e \in S} s(e) \leq K \right\} \tag{3}$$

be the value of the benchmark solution, i.e., the optimal solution obtained after discarding the top green element $g_{\max}$. Let S^* be the set of green elements corresponding to this benchmark.

In §3.1 we give a $(1 + \varepsilon)$-competitive algorithm assuming we have a factor $\text{poly}(n)$-approximation to the benchmark value V^*. (In fact, given this $\text{poly}(n)$-approximation, we can even get within a $(1 + \varepsilon)$-factor of the optimal set *including* $g_{\max}$.) Then in §3.2 we remove the assumption, but now our value is only comparable to V^* (i.e., excluding $g_{\max}$).

Intuition. The main idea of the regular (non-robust) multiple-secretary problem (where we pick at most K items) is to observe a small ε fraction of the input, estimate the value of the K^{th} largest element, and then select elements with value exceeding this estimate. (A better algorithm revises these estimates over time, but let us ignore this optimization for now.) In the Byzantine case, there may be an arbitrary number of red items, so strategies that try to estimate some statistics (like the K^{th} largest) to use for the rest of the algorithm are susceptible to adversarial attacks.

For now, suppose we know that all items of S^* have values in $[1, n^c]$ for some constant c. The density of an item to be its value divided by its size. We define $O(\log n)$ *density levels*, where elements in the same level have roughly the same density, so our algorithm does not distinguish between them. The main idea of our algorithm is to use *cascading budgets*. At the beginning we allocate all our budget to picking only the highest-density level items. If we find that we are not picking items at a rate that is "good enough", we re-allocate parts of

our budget to lower-density levels. The hope is that if the benchmark solution S^* selects many elements from a certain density level, we can get a good estimate of the "right" rate at which to pick up items from this level. Moreover, since our budgets trickle from higher to lower densities, the only way the adversary can confuse us is by giving dense red elements, in which case we will select them.

Such an idea works only for the value levels that contain many elements of S^*. For the remaining value levels, we allocate *dedicated budgets* whose sole purpose is to pick a "few" elements from that level, irrespective of whether they are from S^*. By making the total number of levels logarithmic, we argue that the total amount of dedicated budgets is only $o(K)$, so it does not affect the analysis for the cascading budget.

3.1 An Algorithm Assuming a Polynomial Approximation

Suppose we know the benchmark V^* to within a polynomial factor: by rescaling, assume that V^* lies in the range $[1 \dots n^c]$ for some constant c. This allows us to simplify the instance structure as follows: Firstly, we can pick all elements of size at most $1/n$, since the total space usage is at most $n \cdot 1/n = 1$ (recall, $K \geq \text{poly}(\varepsilon^{-1} \log n)$). Next, we can ignore all elements with value less than $1/n^2$ because their total value is at most $1/n \ll 1 \leq V^*$. If the *density* of an element is defined to be the ratio $v(e)/s(e)$, then all remaining elements have density between n^{c+1} and n^{-2}. The main result of this section is the following:

▶ **Lemma 7.** *If V^* lies between 1 and n^c for some constant c, each element has size at least $1/n$ and value at least $1/n^2$, and $K \geq \text{poly}(\varepsilon^{-1} \log n)$, then there exists a $(1 + O(\varepsilon))$-competitive algorithm.*

The idea of our algorithm is to partition the input into $1/\delta$ disjoint pieces (δ is a small parameter that will be chosen later) and try to solve $1/\delta$ "similar-looking" instances of the knapsack problem, each with a knapsack of size δK.

The Algorithm. Define *checkpoints* $T_i := \delta i$ and corresponding intervals $\mathbb{I}_i := \langle T_{i-1}, T_i]$ for all $i \in [1 \dots 1/\delta]$. Define $L := (1 + \frac{c+3}{\varepsilon} \log n)$ *density levels* as follows: for each integer $\ell \in [0 \dots L)$, density value $\rho_\ell := n^{c+1}/(1+\varepsilon)^\ell$. Now density level ℓ corresponds to all densities lying in the range $(\rho_{\ell+1}, \rho_\ell]$. Note that densities decrease as ℓ increases. We later show that the setting of parameters $K \geq \Omega\big(\frac{L^2 \log L/\varepsilon}{\varepsilon^4}\big)$ and $1/\delta = \Omega(L/\varepsilon)$ suffices.

We maintain two kinds of *budgets*:

- *Cascading budgets*: We maintain a budget $B_{\ell,i}$ for each density level ℓ and each interval $\mathbb{I}_i$. For the first interval $\mathbb{I}_1$, define $B_{0,1} := \delta K$, and $B_{\ell,1} := 0$ for $\ell > 0$. For the subsequent intervals, we will set $B_{\ell,i}$ in an online way as described later.
- *Dedicated budgets*: We maintain a dedicated budget $\widetilde{B}_\ell := H$ for each density level ℓ; we will later show that setting $H := \Omega\big(\frac{L \log L/\varepsilon}{\varepsilon^3}\big)$ suffices.

Suppose we are in the interval $\mathbb{I}_i$, and the arriving element e has density $v(e)/s(e)$ in level ℓ.

1. If the remaining cascading budget $B_{\ell',i}$ of one of the density levels $\ell' \geq \ell$ is positive then select e. For the smallest $\ell' \geq \ell$ satisfying this condition, update $B_{\ell',i} \leftarrow B_{\ell',i} - s(e)$.
2. Else, if the remaining dedicated budget $\widetilde{B}_\ell$ for level ℓ is positive, select e and update $\widetilde{B}_\ell \leftarrow \widetilde{B}_\ell - s(e)$.

Finally, for $i > 1$, we define the cascading budgets $B_{\ell,i}$ for this interval $\mathbb{I}_i$ based on how much of the budgets at levels ℓ and $\ell - 1$ are consumed in the previous interval $\mathbb{I}_{i-1}$ as follows. The amount of budget $B_{\ell-1,i-1}$ at level $\ell - 1$ that is not consumed in interval $\mathbb{I}_{i-1}$ is moved to level ℓ (which has lower density), and the budget that gets consumed in $\mathbb{I}_{i-1}$ is restored

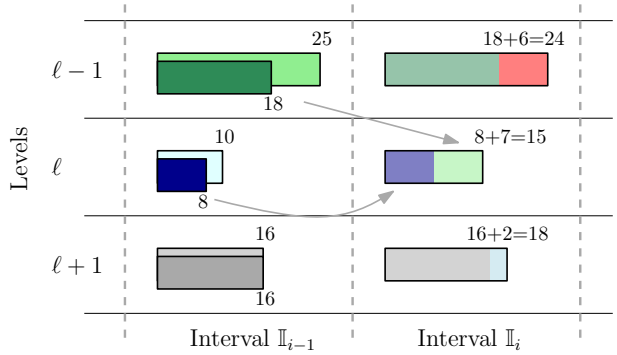


Figure 1 The bars for $\mathbb{I}_{i-1}$ show the budget, and (in darker colors) the amount consumed. The consumed budget (in dark blue) at level ℓ in interval $\mathbb{I}_{i-1}$ is restored at level ℓ in $\mathbb{I}_i$; the unconsumed budget at level $\ell-1$ in interval $\mathbb{I}_{i-1}$ is then added to it.

at level ℓ (see Figure 1). Formally, if $C_{\ell,i-1}$ is the amount of *consumed* cascading budget for level ℓ in interval $\mathbb{I}_{i-1}$ and $R_{\ell,i-1}$ is the amount of *remaining* budget at level ℓ at the end of interval $i-1$ (i.e., the value of $B_{\ell,i-1}$ at the time corresponding to the end of $\mathbb{I}_{i-1}$), then we define the initial budget for level ℓ at the start of interval i to be

$$B_{\ell,i} := C_{\ell,i-1} + R_{\ell-1,i-1}.$$

It is easy to see that we can compute these cascading budgets online.

A Note about Budget Violations. The total budget, summed over both categories and over all the intervals for the cascading budgets, is $K' := ((1/\delta)\cdot\delta K) + L\,H > K$. If we use up all this budget, we would violate the knapsack capacity. Moreover, we select an element as long as the corresponding budget is positive, and hence may exceed each budget by the size of the last element. However, since $K' \le (1+\varepsilon)K$ and K is much larger than individual element sizes, the violations is a small fraction of K, so we can *reject* each element originally selected by the algorithm with some small probability (e.g., $p = 2\varepsilon$) to guarantee that the non-rejected selected elements have size at most K with high probability (i.e., at least $1 - 1/n^c$, for an arbitrary constant $c > 0$). Henceforth, we will not worry about violating the budget.

The Analysis. Recall the benchmark V^* from (3), and let S^* be a set that achieves this value. All the elements have value in $[1/n^2, n^c]$ and size at least $[1/n, 1]$, so each element $e \in S^*$ has a corresponding density level $\ell(e) \in [0\ldots L)$ based on its density $v(e)/s(e)$. We need the notion of "heavy" and "light" levels. For any level $\ell \in [0\ldots L)$, define s^*_ℓ to be the total size of elements in S^* with density level ℓ:

$$s^*_\ell := \textstyle\sum_{e\in S^*:\ell(e)=\ell} s(e). \tag{4}$$

We say a level ℓ is *heavy* if $s^*_\ell \ge H$, else level ℓ is *light*. We refer to (green) elements of S^* at a heavy (resp., light) level as heavy-green (resp., light-green) elements. Note that elements not in S^* (some are red and others green) are left unclassified. If H is sufficiently large, a concentration-of-measure argument using the uniformly random arrival times for green items shows that each heavy level receives $(1-\varepsilon)\delta H$ size during each interval with high probability. The idea of the proof is to argue that the cascading budget never "skips" a heavy level, and hence we get almost all the value of the heavy levels.

To avoid double-counting the values of the light levels, we separately account for the algorithm's value attained (a) on light levels using the dedicated budget or on light-green elements using the cascading budget, and (b) for elements that are not light-green (incl.

red elements) using the cascading budget. Note that (a) and (b) are disjoint, hence their contributions can be added up. We show that (a) exceeds the value of S^* restricted to the light levels, while (b) exceeds $(1-\varepsilon)$ times the value of of S^* on the heavy levels. This is sufficient to prove our result. We start by arguing the former claim.

▷ Claim 8 (Light-Green Elements). The sum of values of elements selected using the dedicated budget at light levels, and of light-green elements selected using the cascading budget, is at least $\sum_{\ell \text{ light}} s^*_\ell \cdot \rho_{\ell+1}$.

Proof. Our algorithm attempts to select each light-green element in S^* using the cascading budget, or failing that, by the dedicated budget at its density level. The only case in which a light-green element $e \in S^*$ is dropped is if all the dedicated budget at its level $\ell(e)$ has been exhausted. But this means the algorithm has already collected at least $s^*_\ell \cdot \rho_{\ell+1}$ from the dedicated budget at this light level ℓ. ◁

Next, to prove that (b) exceeds the value on heavy levels (up to $1-\varepsilon$), we need the following property of the cascading budget on the heavy levels.

▷ Claim 9. For all intervals $\mathbb{I}_i$ and levels ℓ, w.h.p. we have that if $B_{\ell,i} > 0$ then every heavy level $\ell' < \ell$ satisfies $B_{\ell',i} \geq \delta s^*_{\ell'} \cdot (1-\varepsilon)$.

Proof. For a heavy level ℓ', the expected size of heavy-green elements from S^* falling in any interval is $\delta s^*_{\ell'} \geq \delta H$. If $\delta H \geq \frac{\Omega(\log(L/(\delta\varepsilon)))}{\varepsilon^2}$ then with probability $1-\varepsilon$ we get that for each interval i and each heavy level ℓ', the total size of elements from S^* lying at level ℓ' and arriving in interval $\mathbb{I}_i$ is at least $\delta s^*_{\ell'} \cdot (1-\varepsilon)$, by a concentration bound. Henceforth, let us condition on this event happening for all heavy levels ℓ'.

Now if the cascading budget $B_{\ell,i} > 0$, this budget must have gradually come from levels $\ell' < \ell$ of higher densities. But this means $B_{\ell',i} \geq \delta s^*_{\ell'} \cdot (1-\varepsilon)$ because otherwise the cascading budget would never move to level $\ell'+1$, since level ℓ' receives at least $\delta s^*_{\ell'} \cdot (1-\varepsilon)$ size of elements in every interval. ◁

For a level τ let $h^*_{[0,\tau\rangle} := \sum_{\ell' \text{heavy},\ \ell'<\tau} s^*_{\ell'}$ denote the total size of items in S^* restricted to heavy levels from $[0,\tau\rangle$. Similarly, let $h^{\mathcal{A}}_{[0,\tau\rangle}$ be the total size of non-light-green items collected by the algorithm in levels $[0,\tau\rangle$ and charged against the cascading budget.

▷ Claim 10. For all levels τ we have that $h^{\mathcal{A}}_{[0,\tau\rangle} \geq (1-O(\varepsilon))h^*_{[0,\tau\rangle}$.

Proof. Let t be the smallest index of an interval where $B_{\tau,t+1} > 0$. We partition the intervals into two groups: $\mathbb{I}_1, \mathbb{I}_2, \ldots, \mathbb{I}_t$ and $\mathbb{I}_{t+1}, \ldots, \mathbb{I}_{1/\delta}$. From Claim 9 we can conclude that for *each interval* in the latter group, the algorithm collects a total size of at least $\sum_{\ell \text{ heavy},\ \ell<\tau}(1-\varepsilon)\,\delta\, s^*_\ell = (1-\varepsilon)\,\delta\, h^*_{[0,\tau\rangle}$ from levels $[0,\tau\rangle$. Hence the total contribution over all the intervals of the latter group is $(\frac{1}{\delta} - t)(1-\varepsilon)\,\delta\, h^*_{[0,\tau\rangle} = (1-t\delta)(1-\varepsilon)\,h^*_{[0,\tau\rangle}$.

We now consider the group $\mathbb{I}_1, \ldots, \mathbb{I}_t$. Let C_i, R_i and Q_i be the total size of the consumed non-light-green, remaining budget, and consumed light-green elements charged to the cascading budget in interval i with levels $[0,\tau\rangle$. By definition, the total size of all light-green elements is at most LH, giving $\sum_{i=1}^t Q_i \leq LH$. Furthermore, since the full cascading budget is contained in $[0,\tau\rangle$, the algorithm construction guarantees $C_i + R_i + Q_i = \delta K$. Finally, we argue that $\sum_{i=1}^t R_i \leq \delta KL$: consider an infinitesimally small part dB of the budget. At the end of each interval, dB is either used to consume an element or it "moves" from level ℓ to $\ell+1$, which can happen at most L times. Since the total amount of budget per interval is $\int dB = \delta K$, the total sum is at most δKL.

This lower-bounds the total size contribution of the group $\mathbb{I}_{t+1}, \ldots, \mathbb{I}_{1/\delta}$.

$$\begin{aligned}\sum_{i=1}^{t} C_i &= t\delta K - \sum_{i=1}^{t} R_i - \sum_{i=1}^{t} Q_i \quad \geq \quad t\delta K - \delta K L - LH \quad \geq \quad (t\delta - \varepsilon - \varepsilon)K \\ &\geq (t\delta - O(\varepsilon))\, h^*_{[0,\tau\rangle},\end{aligned}$$

where we use $K \geq h^*_{[0,\tau\rangle}$ (since the total size of elements in S^* is at most K), $\delta L \leq \varepsilon$, and $LH \leq \varepsilon K$. Combining contributions from both groups we get:

$$\begin{aligned}(1-t\delta)(1-\varepsilon)\, h^*_{[0,\tau\rangle} + (t\delta - O(\varepsilon))\, h^*_{[0,\tau\rangle} &= [(1-\varepsilon) - t\delta(1-\varepsilon) + t\delta - O(\varepsilon)]\ h^*_{[0,\tau\rangle} \\ &= (1 - O(\varepsilon))\ h^*_{[0,\tau\rangle}.\end{aligned}$$

Hence, we conclude that $h^{\mathcal{A}}_{[0,\tau\rangle} \geq (1 - O(\varepsilon))\, h^*_{[0,\tau\rangle}$. ◁

Using the above claims we now prove Lemma 7.

Proof of Lemma 7. Our fine-grained discretization of densities gives us that

$$V^* \leq (1+\varepsilon)\Bigg(\sum_{\ell \text{ light}} s^*_\ell \rho_{\ell+1} + \sum_{\ell \text{ heavy}} s^*_\ell \rho_{\ell+1}\Bigg). \tag{5}$$

From Claim 8, our algorithm accrues value at least $\sum_{\ell \text{ light}} s^*_\ell \cdot \rho_{\ell+1}$ due to the elements from light levels that were charged to the dedicated budget and light-green elements charged to the cascading budget. It is therefore sufficient to prove a similar bound on the value accrued on non-light-green elements charged to the cascading budget with respect to $\sum_{\ell \text{ heavy}} s^*_\ell \cdot \rho_{\ell+1}$, which we deduce from Claim 10.

Let $\ell'(x)$ be defined as the largest level ℓ' where $\rho_{\ell'} \geq x$, then

$$\begin{aligned}\sum_{\ell \text{ heavy}} s^*_\ell \rho_{\ell+1} &= \int_0^\infty \sum_{\ell \text{ heavy},\ \rho_{\ell+1} \geq x} s^*_\ell \, dx \quad = \quad \int_0^\infty h^*_{[0,\ell'(x)\rangle}\, dx \\ &\leq (1 + O(\varepsilon)) \int_0^\infty h^{\mathcal{A}}_{[0,\ell'(x)\rangle}\, dx,\end{aligned}$$

where the last inequality uses Claim 10. Notice the right-hand side is the value of non-light-green elements charged against the cascading budget. Thus, this part of the algorithm's value exceeds (up to $1 - O(\varepsilon)$) the value of heavy levels of S^*, finalizing our proof. ◀

3.2 An Algorithm for the General Case

To remove the assumption that we know a polynomial approximation to V^*, the idea is to ignore the first ε fraction of the arrivals, and use the maximum value in this interval to get a poly(n) approximation to the benchmark. This strategy is easily seen to work if there are $\Omega(1/\varepsilon)$ elements with a non-negligible contribution to V^*. For the other case where most of the value in V^* comes from a small number of elements, we separately reserve some of the budget, and run a collection of algorithms to catch these elements when they arrive.

Formally, we define $(1/\varepsilon)$ *checkpoints* $T_i := i\varepsilon$ and corresponding intervals $\mathbb{I}_i := \langle T_{i-1}, T_i]$ for all $i \in [1 \ldots (1/\varepsilon)]$. We run the following three algorithms in parallel, and select the union of elements selected by them.

(i) Select one of the n elements uniformly at random; i.e., run Select-Random-Element from §2.1.

(ii) Ignore elements that arrive in times $[0, \varepsilon)$, and let $\hat{v}$ denote the highest of their values. Run the algorithm from §3.1 during time $[1/\varepsilon, 1]$, assuming that $V^* \in [\hat{v}/n^2, \hat{v}\, n^2]$.

(iii) At every checkpoint T_i, consider the largest value $\hat{v}_i$ seen until then. Define $L := \frac{10}{\varepsilon} \log n$ *value levels* as follows: for $\ell \in (-L/2 \dots L/2)$ and $\tau_\ell := \hat{v}_i/(1+\varepsilon)^\ell$, define level $\ell(i)$ as corresponding to values in $(\tau_{\ell(i)+1}, \tau_{\ell(i)}]$. For each of these levels ℓ, keep $D := \frac{10}{\varepsilon} \log \frac{1}{\varepsilon}$ dedicated slots, and select any element having this value level and arriving after T_i, as long as there is an empty slot in its level.

The total space used by the there algorithms is at most

$$1 + K + (1/\varepsilon) \cdot L \cdot D \quad = \quad K + O\Big(\frac{\log n \log {}^1\!/_\varepsilon}{\varepsilon^3}\Big) \quad \leq \quad (1+\varepsilon)K,$$

where the last inequality holds because $K \geq \Omega\big(\frac{L^2 \log(L/\varepsilon)}{\varepsilon^4}\big)$ from the size condition from §3.1. We can now fit this into our knapsack of size K w.h.p. by sub-sampling each selected element with probability $(1 - O(\varepsilon))$. To complete the proof of Theorem 2, we need to show that we get expected value $(1 - O(\varepsilon))V^*$.

Proof of Theorem 2. The proof considers several cases. Firstly, if there is any single element with value more than $n \cdot V^*$, then the algorithm in Step (i) will select it with probability $1/n$, proving the claim. Hence, all elements have value at most nV^*.

Now suppose at least $D = \frac{10}{\varepsilon} \log \frac{1}{\varepsilon}$ elements in S^* (recall S^* has total value V^*) have individual values at least V^*/n^2. In this case, at least one of these D elements arrives in the interval $[0, \varepsilon)$ with probability $1 - \varepsilon$, and that element gives us the desired n^2-approximation to V^*. Moreover, the expected value of elements in S^* arriving in times $[\varepsilon, 1]$ is at least $(1 - O(\varepsilon))\, V^*$, even conditioning on one of them arriving in $[0, \varepsilon)$.

Finally, consider the case where $D' \leq D$ elements of S^* have value more than V^*/n^2. The idea of the algorithm in Step (iii) is to use the earliest arriving of these D' elements, or the element $g_{\max}$, to get a rough estimate of V^*, and from thereon use the dedicated slots to select the remaining elements. Indeed, if the first of these elements arrive in interval $\mathbb{I}_i$, the threshold $\hat{v}_i$ lies in $[V^*/n^2, nV^*]$ (since we did not satisfy the first case above). Now the value levels and dedicated budgets set up at the end of this interval would pick the rest of these D' elements – except those that fall in this same interval $\mathbb{I}_i$. We argue that each of the remaining D' elements has at least $(1 - \varepsilon)$ probability of not being in $\mathbb{I}_i$, which gives us an expected value of $(1 - O(\varepsilon))V^*$ in this case as well. This is true because the expected number of these $1 + D'$ elements (including $g_{\max}$) that land in any interval that contains at least one of them is at most $1 + \varepsilon D'$ (even after we condition on the first arrival, each remaining element has ε chance of falling in this interval). Since any such interval has the same chance of being the first interval $\mathbb{I}_i$, and these $1 + D'$ elements have the same distribution, the expected number of additional elements in $\mathbb{I}_i$ is $\varepsilon D'$. ◀

This completes the proof of Theorem 2 for the knapsack case, where the size K of the knapsack is large enough compared to the largest size of any element. This generalizes the multiple-secretary problem, where all items have unit size. We have not optimized the value of K that suffices, opting for modularity and simplicity. It can certainly be improved further, though getting an algorithm that works under the assumption that $K \geq O(1/\varepsilon^2)$, like in the non-robust case, may require new ideas.

4 Single-Item Ordinal Case

In this section we give a proof of Theorem 4, showing that there exists an algorithm which selects an element with value no smaller than g_2, with probability at least $\Omega(1/\log^2 n)$. Our proof for this theorem is non-constructive and uses (the hard direction of) the Minimax

Theorem; hence we can currently only show the *existence* of this algorithm, and not give a compact description for it. Our main technical lemma furnishes an algorithm which, given a known (general) probability distribution $\mathcal{B}$ over input instances, selects a big element with probability at least $\Omega(1/\log^2 n)$. Consequently, we use the Minimax Lemma to deduce that the known-distribution case is equivalent to the worst-case input setting and recover the analogous result.

Since our algorithms crucially argue about the input distribution $\mathcal{B}$ and rely on the Minimax, we need to formally define these terms and establish notation connecting the Byzantine secretary problem with two-player zero-sum games. Suppose we want to maximize the probability of selecting a big element and to this end we choose an algorithm $\mathcal{A}$, while the adversary chooses a distribution $\mathcal{B}$ over the input instances and there is an (infinite) payoff matrix K prescribing the outcomes. Its rows are indexed by different algorithms, and columns by input instances. Formally, a *"pure" input instance* is represented as an $|R|$-tuple of numbers in $[0,1]$, representing the arrival times t_e of the red elements; and a permutation $\pi \in S_n$ over U representing the total ordering of all values in $U = R \cup G$. Recall that the green elements G choose their arrival times independently and uniformly at random in $[0,1]$, hence their t_e's are not part of the input. A *"mixed" input instance* is a probability distribution $\mathcal{B}$ over pure instances $[0,1]^{|R|} \times S_n$.

While we do not need the full formal specifications of algorithms, we will mention that a "mixed" algorithm $\mathcal{A}$ is a distribution over deterministic algorithms. An algorithm $\mathcal{A}$ on an input instance I gets a payoff of $K(\mathcal{A}, I) := \Pr[v(\mathcal{A}) \geq v(g_2) \mid I]$ where the probability is taken over the assignment of random arrival times to elements in G and the distribution of deterministic algorithms $\mathcal{A}$. The following Lemma states that for each $\mathcal{B}$ there is an algorithm (that depends on $\mathcal{B}$) that selects a big elements with probability $\Omega(1/\log^2 n)$. We prove the result in §4.1 and §4.2.

▶ **Lemma 11** (Known Distribution Ordinal Single-Item Algorithm). *Given a distribution over input instances $\mathcal{B}$, there exists an algorithm $\mathcal{A}$ that has an expected payoff of $\Omega(1/\log^2 n)$.*

To deduce the general case from the known distribution setting, we use a minimax lemma for two-player games. We postpone the details to Appendix C and simply state the final result here.

▶ **Theorem 4** (Ordinal Single-item Algorithm). *There is a randomized algorithm for the ordinal Byzantine secretary which selects an element of value at least the second-largest green item with probability $\Omega(1/\log^2 n)$.*

4.1 The Algorithm when $\mathcal{B}$ is Known

In this section we give the algorithm for Lemma 11. We start with some preliminary notation. For each element e, let t_e denote the time at which it appears. Furthermore, for $t \in [0,1]$, let $\mathcal{K}(t)$ denote the information seen by the algorithm up to and including time t, consisting of arrival times and relative values of elements appearing before t.

We define $\log n + 1$ time *checkpoints* as follows: set the initial checkpoint $T_0 := \frac{1}{4}$, and then subsequent checkpoints $T_i := \frac{1}{4} + \frac{i}{2 \cdot \log n}$ for all $i \in [1 \dots \log n]$. Note that the last checkpoint is $T_{\log n} = \frac{3}{4}$. Now the corresponding intervals are

$$\mathbb{I}_0 := [0, 1/4] \quad , \quad \mathbb{I}_i := \langle T_{i-1}, T_i] \ \forall\, i \in [1 \dots \log n], \quad \text{and} \quad \mathbb{I}_{\log n+1} := \langle 3/4, 1]. \tag{6}$$

Let $m_i := \max\{v(e) \mid e \in R \text{ and } t_e \in \mathbb{I}_i\}$ be the maximum value among the red elements that land in interval $\mathbb{I}_i$, and let $\mathcal{H} := \{m_i > v(g_2) \text{ for all } i \in [1 \dots \log n]\}$ be the event where the maximum value red item in all intervals is larger than the target g_2, i.e., is "big". We

call this event $\mathcal{H}$ the *hard cases* and $\mathcal{H}^c$ the *easy cases*; we will show the Two Checkpoints Secretary (from §2.1) achieves $\Omega(1/\log^2 n)$ winning probability for all input instances in $\mathcal{H}^c$. Finally, define

$$p_e^i := \Pr_{\mathcal{B}}[e = g_2 \mid \mathcal{H} \text{ and } \mathcal{K}(T_i)],$$

i.e., p_e^i is the probability that e is the second-highest green element conditioned on the information seen until checkpoint T_i and the current instance being hard. Importantly, the algorithm can compute p_e^i at T_i.

Now to solve the hard cases, at each checkpoint T_i the algorithm computes sets S_i satisfying $S_{i+1} \subseteq S_i$. These sets represent elements which are candidates for the second-max. In other words, at time T_i there is reasonable probability that second-max is in S_i. We start with defining $S_0 \leftarrow \{e \in U \mid t_e \in \mathbb{I}_0\}$, the elements the algorithm saw before T_0. For $i \geq 0$, let c_i denote the *center* of S_i, i.e., the element of S_i such that there are exactly $\lfloor |S_i|/2 \rfloor$ elements smaller than it. Define $p^i(X) := \sum_{e \in X} p_e^i$ for a set X and index $i \geq 0$. Given S_{i-1}, we determine S_i as follows:

- Define $\mathsf{bot}_{i-1} \leftarrow \{e \in S_{i-1} \mid v(e) \leq v(c_{i-1})\}$, and note that $\mathsf{bot}_{i-1} \subseteq S_{i-1}$.
- If $p^i(\mathsf{bot}_{i-1}) = p^{i-1}(S_{i-1})$ then $S_i \leftarrow \mathsf{bot}_{i-1}$, else $S_i \leftarrow S_{i-1} \setminus \mathsf{bot}_{i-1}$.

Our algorithm runs one of the following three algorithms uniformly at random:

(i) Select a random $i \in [1 \dots \log n]$, define $\tau \leftarrow \max\{v(e) \mid t_e \in \mathbb{I}_i\}$ and select the first element larger than τ. I.e., run Two Checkpoints Secretary (from §2.1) with the checkpoints being the ends of interval $\mathbb{I}_i$.

(ii) Select a random $i \in [0 \dots \log n]$, read input until checkpoint T_i, define $\tau \leftarrow v(c_i)$ and select the first element larger than it.

(iii) Compute the sets S_i until $|S_k| \leq 10$ for some k: then define τ to be the value of a random element in S_k, and select the first element larger than it.

4.2 The Analysis

In this section we prove Lemma 11. Let us give some intuition. We can assume we have a hard case, else the first algorithm achieves $\Omega(1/\log^2 n)$ winning probability. For the other two algorithms, let us condition on g_2 falling in the first interval $\mathbb{I}_0$, and then exploit the fact that there is a big red element in every interval $\mathbb{I}_i$. It may be useful to imagine that we are trying to guess, at each checkpoint, which of the elements in the past were actually g_2. If we could do this, we would set a threshold at its value, and select the first subsequent element bigger than the threshold – and since there is a $1/4$ chance that $g_{\max}$ would fall in $\mathbb{I}_{\log n+1}$, we'd succeed! Of course, since there are red elements all around, guessing g_2 is not straightforward.

So suppose we are at checkpoint T_k, and suppose there is a reasonable probability that $v(g_2) \leq v(c_{k-1})$, but also still some nonzero probability that $v(g_2) > v(c_{k-1})$. In such a scenario, we claim that trying to choose an element in the interval $\mathbb{I}_k$ larger than c_{k-1} will give us a reasonable probability of success. Indeed, we claim there would have been at least one red element in $\mathbb{I}_k$ bigger than c_{k-1} (since there is still a non-zero probability that $v(g_2) > v(c_{k-1})$ even at the end of the interval $\mathbb{I}_k$, and since the case is hard), and $v(g_2) \leq v(c_{k-1})$ with reasonable probability. Of course, we only know this at the end of the interval, but the algorithm can randomly guess k with $\Omega(1/\log n)$ probability. Finally, if there is no such checkpoint, then in every interval we reduce the size of set $|S_i|$ by half while suffering a small loss in $p(S_i)$. In this case, both $|S_{\log n}| = O(1)$ and $p(S_{\log n}) = \Omega(1)$, so the third algorithm can guess g_2 with constant probability and select an element larger than it in the last interval.

Formal Analysis. Let ALG be 1 if $v(\mathcal{A}) \geq v(g_2)$ and 0 otherwise, where $\mathcal{A}$ is the algorithm from the last section. Suppose we're in an easy case, i.e., there is an interval $\mathbb{I}_s$ such that all red elements in this interval are smaller than g_2. Now if the first algorithm is chosen, suppose it selects the interval $\mathbb{I}_s$, suppose g_2 lands in $\mathbb{I}_s$, and $g_{\max}$ lands in $\mathbb{I}_{\log n+1}$. Then the algorithm surely selects an element greater than g_2, and it has expected value:

$$\mathbb{E}[\text{ALG}] \geq \tfrac{1}{3} \cdot \tfrac{1}{\log n} \cdot \tfrac{1}{2 \log n} \cdot \tfrac{1}{4} = \Omega\left(\tfrac{1}{\log^2 n}\right).$$

Henceforth we can assume the case is hard, and hence each interval I_i contains a red element bigger than g_2. We condition on the event that g_2 appears in $\mathbb{I}_0$, which happens with constant probability. Define

$$k^* := \min\left\{i \in [1 \dots \log n] \mid \tfrac{1}{\log n} \leq \tfrac{p^i(\mathsf{bot}_{i-1})}{p^{i-1}(S_{i-1})} < 1\right\},$$

and set $k^* = \log n + 1$ if the above set is empty.

▷ Claim 12. For all $i < k^*$, the probability $p^i(S_i) = \Omega(1)$.

Proof. By definition, $p^0(S_0) = 1$. By our definition of the sets S_i, we know that if $p^i(\mathsf{bot}_{i-1}) = p^{i-1}(S_{i-1})$ then $p^i(S_i) = p^{i-1}(S_{i-1})$. Else since $i < k^*$, we have

$$p^i(S_i) = p^{i-1}(S_{i-1}) - p^i(\mathsf{bot}_{i-1}) \leq p^{i-1}(S_{i-1})(1 - \tfrac{1}{\log n}).$$

Hence, $p^i(S_i) \geq (1 - \frac{1}{\log n})^{\log n} = \Omega(1)$, proving the claim. ◁

Now there are two cases, depending on the value of k^*. Suppose $k^* \leq \log n$. Condition on the event that the second algorithm is chosen, that it chooses the $i^{th} = (k^* - 1)^{th}$ checkpoint, and that $v(g_2) \leq v(c_i)$. By our choice of k^*, we get that $v(g_2) \leq \tau = v(c_i)$ with probability at least $p^i(S_i) \cdot \frac{1}{\log n}$, and by Claim 12 this is $\Omega(\frac{1}{\log n})$. Since the case we are considering is hard and $\Pr[v(g_2) > v(c_i) \mid \mathcal{H} \text{ and } \mathcal{K}(T_{i+1})] > 0$, there is a red element larger than $v(c_i) = \tau$ appearing in $\mathbb{I}_i$. Thus the algorithm will always select an element in this interval. The correct interval is chosen with probability $\frac{1}{\log n}$, so the algorithm's value is

$$\mathbb{E}[\text{ALG}] = \tfrac{1}{3} \cdot \tfrac{1}{\log n} \cdot \Omega\big(\tfrac{1}{\log n}\big) = \Omega\big(\tfrac{1}{\log^2 n}\big).$$

The other case is when $k^* = \log n + 1$. By definition $|S_0| \leq n$ and $|S_i| \leq \lceil |S_{i-1}|/2 \rceil$. Therefore $|S_{\log n}| \leq 10$. Let us condition on the event that the third algorithm is chosen, that $g_{\max}$ appears in $\mathbb{I}_{\log n+1}$, and that the algorithm guesses g_2 correctly. The probability of this event is at least

$$\tfrac{1}{3} \cdot \tfrac{1}{4} \cdot p^{\log n}(S_{\log n}) \cdot \tfrac{1}{10} = \Omega(1).$$

where we use Claim 12 to bound the probability $p^{\log n}(S_{\log n})$. In this event, the algorithm selects an element larger than g_2 and has expected value $\mathbb{E}[\text{ALG}] = \Omega(1)$.

Putting all these cases together, we get that our algorithm selects an element with value at least $v(g_2)$ with probability at least $\Omega((\log n)^{-2})$. This finishes the proof of Lemma 11, and hence of Theorem 4. It remains an intriguing open question to get a direct algorithm that achieves similar guarantees.

5 Single-Item Value-Maximization

In this section, we give an algorithm for the problem of selecting an item to maximize the expected *value*, instead of maximizing the probability of selecting the second-largest green item (the ordinal problem considered in §4). In the classical secretary problem, both problems are well known to be equivalent, with Dynkin's algorithm giving a tight $1/e$ bound for both. But in the Byzantine case the problems thus far appear to have different levels of complexity: in §6.2 we present a simple $O(\log n)$-competitive algorithm for the value-maximization byzantine secretary problem, which is already better than the $\operatorname{poly}\log n$-competitive of §4. We now substantially improve it to give a $\operatorname{poly}\log^* n$-competitive ratio.

▶ **Theorem 3** (Value Maximization Single-Item). *There is a randomized algorithm for the value-maximization (single-item) Byzantine secretary problem which gets an expected value at least* $(\log^* n)^{-2} \cdot V^*$.

In the rest of this section, let $V^* := v(g_2)$ denote the benchmark, the value of the second-largest green element. The high level idea of our algorithm is to partition the input into $O(\log^* n)$ intervals and argue that every interval contains a red element of value $v_i > V^*$, as otherwise Dynkin's algorithm will be successful. Moreover, this v_i cannot be much larger than V^*, as otherwise we can just select a random element. This implies we can use the largest value in each interval to find a good estimate of V^*, and eventually set it as a threshold in the last interval to select a large value element.

5.1 The Algorithm

Define $\log^{(i)} n$ to be the *iterated logarithm* function: $\log^{(0)} n = n$ and $\log^{(i+1)} n = \log(\log^{(i)} n)$. We define $\log^* n + 1$ time *checkpoints* as follows: the initial checkpoint $T_0 = \frac{1}{2}$, and then subsequent checkpoints $T_i = \frac{1}{2} + \frac{i}{4 \cdot \log^* n}$ for all $i \in [1, \ldots, \log^* n]$. Note that the last checkpoint is $T_{\log^* n} = \frac{3}{4}$. Now the intervals are

$$\mathbb{I}_0 := [0, T_0] \quad , \quad \mathbb{I}_i := \langle T_{i-1}, T_i] \; \forall \, i \in [1 \ldots \log^* n], \quad \text{and} \quad \mathbb{I}_{\log^* n + 1} := \langle T_{\log^* n}, 1]. \tag{7}$$

Our algorithm runs one of the following three algorithms chosen uniformly at random.

- **(i)** Select one of the n elements uniformly at random; i.e., run Select-Random-Element from §2.1.
- **(ii)** Select a random interval $i \in [1 \ldots \log^* n]$ and run Dynkin's secretary algorithm on $\mathbb{I}_i$. Formally, run Two-Checkpoints-Secretary (from §2.1) with the interval being $[T_{i-1}, \frac{1}{2}(T_{i-1} + T_i)]$.
- **(iii)** Select a random index $i \in [0 \ldots \log^* n]$ and observe the maximum value during the interval $\mathbb{I}_i$; let this maximum value be v_i. Choose a uniformly random $s \in [0 \ldots 2 \log^{(i)} n]$. Select the first element arriving after T_i that has value at least $\tau := (v_i \log^{(i)} n)/2^s$.

5.2 The Analysis

To prove Theorem 3, assume WLOG that there are only two green elements $g_{\max}$ and g_2, and every other element is red (otherwise, we can condition on the arrival times of all other green elements). Let v_i be the value of the highest red element in $\mathbb{I}_i$, i.e., excluding $g_{\max}$ and g_2.

Proof of Theorem 3. We assume $\log^{(i)} n$ is an integer for all i; this is true with a constant factor loss. For sake of a contradiction, assume that the algorithm in §5.1 does not get expected value $\Omega((\log^* n)^{-2}V^*)$. Under this assumption, we first show that every interval contains a red element of value at least V^*.

▷ Claim 13. For all $j \in [1 \dots \log^* n]$ we have $v_j \geq V^*$.

Proof. Suppose this is not the case. Let $\mathcal{E}_1$ be the event that the following three things happen simultaneously: that we select Algorithm ii in §5.1 with random variable $i = j$, that the second-highest green element g_2 falls in the interval $[T_{i-1}, \frac{1}{2}(T_{i-1}+T_i)\rangle$, and that the highest green element $g_{\max}$ falls in $\mathbb{I}_{\log^* n+1}$. Note that $\Pr[\mathcal{E}_1] = \frac{1}{3\log^* n} \cdot \frac{1}{4\log^* n} \cdot \frac{1}{4} = \Omega((\log^* n)^{-2})$. Conditioned on this event $\mathcal{E}_1$, our algorithm (or specifically, Algorithm ii on the interval $\mathbb{I}_j$) gets a value at least $v(g_2) = V^*$. Hence the algorithm has expected valuation $\Omega\big((\log^* n)^{-2}V^*\big)$, which is a contradiction to our assumption on its performance. ◁

We now prove that these red elements with large values cannot be much larger than V^*.

▶ **Lemma 14.** *For all $j \in [1 \dots \log^* n]$ we have $v_j \leq V^* \cdot \log^{(j-1)} n$.*

Proof. We prove this lemma by induction. The base case $j = 1$ says $v_1 \leq nV^*$, i.e., the highest observed value in $\mathbb{I}_1 = [T_0, T_1\rangle$ is at most nV^*. Suppose this is not the case – there exists a red element e in $\mathbb{I}_1$ with value at least nV^*. Let $\mathcal{E}_1$ be the event that we select Algorithm i in §5.1 (i.e., Select-Random-Element) and that it selects e. Since $\Pr[\mathcal{E}_1] = \Omega(\frac{1}{n})$, we have a contradiction that the expected valuation is $\Omega(V^*)$.

Now suppose the statement is true until $j \geq 1$. We prove the inductive step $j+1$. Suppose not, i.e., $v_{j+1} > V^* \log^{(j)} n$. Let $\mathcal{E}_2$ be the event that we select Algorithm iii in §5.1 with parameter $i = j$ and that the random $s \in [0 \dots 2\log^{(j)} n]$ is such that $\frac{v_j}{2^{s+1}} \leq V^* < \frac{v_j}{2^s}$ (it exists by induction hypothesis). This implies threshold $\tau := \frac{v_j \log^{(j)} n}{2^s}$ is between $\frac{1}{2}V^* \log^{(j)} n$ and $V^* \log^{(j)} n$. Note $\Pr[\mathcal{E}_2] \geq \frac{1}{3} \cdot \frac{1}{\log^* n} \cdot \frac{1}{2\log^{(j)} n}$. Since event $\mathcal{E}_2$ implies the algorithm gets value at least $\tau \geq \frac{1}{2}V^* \log^{(j)} n$ (because $v_{j+1} > \tau$), its expected value is $\Omega((\log^* n)^{-1}V^*)$, a contradiction. ◀

Now by Claim 13 and Lemma 14, we have $v_j \in [V^*, V^* \cdot \log^{(j)} n]$ for all $j \in [1 \dots \log^* n]$. We still get a contradiction. Let $\mathcal{E}_3$ be the event that the following three things happen simultaneously: that we select Algorithm iii in §5.1 with $i = \log^* n$, that the highest green element $g_{\max}$ is in interval $\mathbb{I}_{\log^* n+1}$, and that we select s in Algorithm iii such that $\tau := (v_{\log^* n} \log^{(j)} n)/(2^s)$ is between $\frac{1}{2}V^*$ and V^*. Note $\Pr[\mathcal{E}_3] \geq \frac{1}{3\log^* n} \cdot \frac{1}{4} \cdot \frac{1}{2\log^* n}$. Since the event $\mathcal{E}_2$ implies the algorithm gets value at least $\tau \geq \frac{1}{2}V^*$ (because $g_{\max}$ is in $\mathbb{I}_{\log^* n+1}$), its expected value is $\Omega((\log^* n)^{-2}V^*)$. Thus, we have a contradiction in every case, which means our assumption is incorrect and the algorithm has expected value $\Omega((\log^* n)^{-2}V^*)$. ◀

6 Value Maximization for Matroids

In this section we discuss multiple-choice Byzantine secretary algorithms in the matroid setting.

▶ **Definition 15** (Byzantine secretary problem on matroids). *Let $\mathcal{M}$ be a matroid over $U = R \cup G$, where elements in $G = \{g_{\max}, g_2, \dots, g_{|G|}\}$ arrive uniformly at random in $[0, 1]$. When an element $e \in U$ arrives, the algorithm must irrevocably select or ignore e, while ensuring that the set of selected elements forms an independent set in $\mathcal{M}$. The leave-one-out benchmark V^* is the highest-value independent subset of $G \setminus \{g_{\max}\}$.*

The knapsack results imply $(1-\varepsilon)$-competitiveness for uniform matroids as long as the rank r is large enough; we now consider other matroids.

6.1 $O(\log\log n)^2$-competitiveness for Partition Matroids

A partition matroid is where the elements of the universe are partitioned into parts $\{P_1, P_2, \ldots\}$. Given some integers $r_1, r_2, \ldots$, a subset of elements is independent if for every i it contains at most r_i element from part P_i.

▶ **Theorem 5** (Partition Matroids). *There is an algorithm for Byzantine secretary on partition matroids that is $O(\log\log n)^2$-competitive with the benchmark V^*.*

We prove Theorem 5 for simple partition matroids where all $r_i = 1$, i.e., we can select at most one element in each part. This is without loss of generality (up to $O(1)$ approximation) because we can randomly partition each part P_i further into r_i parts and run the simple partition matroid algorithm.

Recall that our single item $\operatorname{poly}\log^* n$ algorithm from §5 no longer works for partition matroids. This is because besides one part we want to get the highest green element in all the other parts. Formally, Claim 13 where we use Dynkin's secretary algorithm in the proof of Theorem 3 fails because it needs at least two green elements. So we need to overcome the lower bound to getting the highest-value green element $v(g_1)$ in Observation 19. We achieve this and design an $O(\log\log n)^2$-approximation algorithm by making an assumption that the algorithm starts with a polynomial approximation to $v(g_1)$. Although in general this is a strong assumption, it turns out that for partition matroids this assumption is w.l.o.g. because the algorithm may lose the highest green element in one of the parts.

6.1.1 The Algorithm

We define $\log\log n + 1$ time *checkpoints* as follows: the initial checkpoint $T_0 = \frac{1}{2}$, and then subsequent checkpoints $T_i = \frac{1}{2} + \frac{i}{2\cdot\log\log n}$ for all $i \in [1 \ldots \log\log n]$. Now the corresponding intervals are

$$\mathbb{I}_0 := [0, T_0] \quad \text{and} \quad \mathbb{I}_i = \langle T_{i-1}, T_i] \quad \forall\, i \in [1 \ldots \log\log n] \tag{8}$$

Let v_0 denote the value of the max element seen by the algorithm in $\mathbb{I}_0$.

Now for every part P of the partition matroid, we execute the following algorithm separately. Let v_i for $i \in [1 \ldots \log\log n]$ denote the value of the max element seen by the algorithm *in part* P during interval $\mathbb{I}_i$. Let V^* denote the element of our benchmark in P. Notice that $v_i \in P$ and V^* cannot be the overall highest green element as we exclude it. We define $4\log^{1/i} n$ levels for $\mathbb{I}_i$ where *level* j for $j \in [1 \ldots 4\log^{1/i} n]$ is given by elements with values in

$$\Big[\frac{v_{i-1}\cdot\log^{1/i} n}{2^j}, \frac{v_{i-1}\cdot\log^{1/i} n}{2^{j-1}}\Big].$$

We run one of the following algorithms uniformly at random.

(a) Select an element uniformly at random as discussed in §2.1.
(b) For every part P, select a random interval $i \in [1 \ldots \log\log n]$ and select a random level $j \in [4\log^{1/i} n]$. Select the first element above $\frac{v_{i-1}\cdot\log^{1/i} n}{2^j}$ in P.
(c) For every part P, select a random interval $i \in [1 \ldots \log\log n]$ and if there is an element with value more than $2^{\log^{1/i} n}$ times the max of all the already seen elements in $\mathbb{I}_i$, selects it with constant probability, say $1/100$.

6.1.2 The Analysis

Since with constant probability our algorithm selects one of the n elements uniformly at random (Algorithm a), we can assume that $v_0 \le n^2 \cdot V^*$. We always condition on the event that $g_{\max}$ arrives in the interval $\mathbb{I}_0$, which happens with constant probability and implies $v_0 \ge v(g_{\max})$. Moreover, we ignore parts P where V^* is below $v(g_{\max})/n^2$ because they do not contribute significantly to the benchmark. So from now assume

$$V^*/n^2 \le v_0 \le n^2 \cdot V^*.$$

We design an algorithm that gets value $\Omega(V^*/(\log\log n)^2)$ in each part P, which implies Theorem 5 by linearity of expectation over parts.

Let $v_i^{(\text{red})} \in P$ for $i \in [1 \dots \log\log n]$ denote the value of the max red element that the adversary presents in $\mathbb{I}_i$.

▷ Claim 16. If there exists an $i \in [1 \dots \log\log n]$ with $v_i^{(\text{red})} > V^* \cdot \log^{1/i} n$ then the expected value of the algorithm is $\Omega(V^*/\log\log n)$.

Proof. With constant probability, our algorithm selects a random interval i and selects a random level element in it (Algorithm b). Since w.p. $1/\log\log n$ it selects this i, and w.p. $\frac{1}{4}\log^{1/i} n$ it selects the random level of $v_i^{(\text{red})}$ in $\mathbb{I}_i$, the algorithm has expected value at least

$$\frac{1}{\log\log n} \cdot \frac{1}{4\log^{1/i} n} \cdot v_i^{(\text{red})} \ge \frac{1}{4\log\log n} \cdot V^*. \qquad \triangleleft$$

By the last claim we can assume for all $i \in [1 \dots \log\log n]$, we have $v_i^{(\text{red})} \le V^* \cdot \log^{1/i} n$.

▷ Claim 17. If there exists an $i \in [1 \dots \log\log n]$ with $v_i^{(\text{red})} < V^*/2^{\log^{1/i} n}$ then the expected value of the algorithm is $\Omega(V^*/(\log\log n)^2)$.

Proof. With constant probability the algorithm guesses one of the intervals i and if there is an element with value more than $2^{\log^{1/i} n}$ times the max of all the already seen elements in $\mathbb{I}_i$, selects it with constant probability (Algorithm c). With $1/\log\log n$ probability the algorithm selects this particular i and with $1/\log\log n$ probability V^* appears in this interval with value at least $2^{\log^{1/i} n}$ times the max seen element in this interval. Notice there can be at most $O\left(\frac{4\log^{1/i} n}{\log^{1/i} n}\right) = O(1)$ elements with such large jumps in value in this interval. In this case our algorithm selects V^* with constant probability. $\triangleleft$

Finally, we are only left with the case where for all $i \in [1 \dots \log\log n]$ value $\frac{V^*}{2^{\log^{1/i} n}} \le v_i^{(\text{red})} \le V^* \cdot \log^{1/i} n$, which we handle using Algorithm b.

▷ Claim 18. If for all $i \in [1 \dots \log\log n]$ we have

$$\frac{V^*}{2^{\log^{1/i} n}} \le v_i^{(\text{red})} \le V^* \cdot \log^{1/i} n$$

then the expected value of the algorithm is $\Omega(V^*/(\log\log n)^2)$.

Proof. Consider Algorithm b. It selects $i = \log\log n - 1$ w.p. $1/\log\log n$. Moreover, suppose V^* appears in $\mathbb{I}_{\log\log n - 1}$. Now since there are only a constant number of levels in this interval, our algorithm selects an element of value at least V^* with constant probability. $\triangleleft$

We have shown that in every case the algorithm has expected value $\Omega(V^*/(\log\log n)^2)$ for any fixed part P. This implies Theorem 5 by linearity of expectation over parts.

6.2 $O(\log n)$-approx for General Matroids

▶ **Observation 6** (General Matroids). *There is an algorithm for Byzantine secretary on general matroids that is $O(\log n)$-competitive with the benchmark V^*.*

Proof. Notice that no element can have weight more than nr times the second max-element because w.p. $1/n$ our algorithm selects one of the n elements uniformly at random. Given this, condition on the event that the max element with value v lands in the first half of the input. Define $2\log(nr)$ exponentially separated levels as follows:

$$[\frac{v}{2^{\log(nr)}}, \frac{v}{2^{\log(nr)-1}}\rangle, [\frac{v}{2^{\log(nr)-1}}, \frac{v}{2^{\log(nr)-2}}\rangle, \ldots, [\frac{v}{2}, v\rangle, \ldots, [v2^{\log(nr)-1}, v2^{\log(nr)}\rangle.$$

Since at least one of these intervals contains at least $2\log(nr)$ fraction of OPT, we can guess that interval and run a greedy algorithm, i.e., accept any element with value in that interval or above if it is independent. ◀

7 Conclusion

In this paper we defined a robust model for the secretary problem, one where some of the elements can arrive at adversarially chosen times, whereas the others arrive at random times. For this setting, we argue that a natural is the optimal solution on all but the highest-valued green item (or even simpler, the optimal solution on the green items, minus the single highest-value item). This benchmark reflects the fact that we cannot hope to compete with the red (adversarial) items, and also cannot do well if all the green value is concentrated in a single green item.

We show that for the case where we want to pick K items, or if we have a knapsack of size K, we can get within $(1-\varepsilon)$ of this benchmark, assuming K is large enough. We can also get non-trivial results for the single-item case, where our benchmark is now the second-highest valued green item. In the ordinal setting where we only see the relative order of arriving elements and the goal is to maximize the probability of getting an element whose value is above the benchmark, we use the minimax principle to show existence of an $O(\log^2 n)$-approximation algorithm in §4. In the value maximization setting, we give an $O(\log^* n)^2$-approximation algorithm in §5. We also show $O(\log\log n)$-competitiveness for partition matroids.

The results above suggest many question. Can we improve the lower bound on the size required for $(1-\varepsilon)$-competitiveness? Can we get a constant-competitive algorithm for the single-item case? For the probability-maximization problem, our proof only shows the existence of an algorithm; can we make this constructive? More generally, many of the algorithms for secretary problems seem to overfit to the model, at least in the presence of small adversarial changes: how can we make our algorithms robust?

References

1 Moshe Babaioff, Nicole Immorlica, and Robert Kleinberg. Matroids, secretary problems, and online mechanisms. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, 2007.

2 Avrim Blum and Joel Spencer. Coloring Random and Semi-Random k-Colorable Graphs. *J. Algorithms*, 19(2):204–234, 1995.

3 Moses Charikar, Jacob Steinhardt, and Gregory Valiant. Learning from untrusted data. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 47–60, 2017.

4 Ning Chen, Nick Gravin, and Pinyan Lu. Optimal competitive auctions. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pages 253–262. ACM, 2014.

5 Kai-Min Chung, Michael Mitzenmacher, and Salil P. Vadhan. Why Simple Hash Functions Work: Exploiting the Entropy in a Data Stream. *Theory of Computing*, 9:897–945, 2013.

6 Kenneth L. Clarkson and Peter W. Shor. Applications of random sampling in computational geometry. II. *Discrete Comput. Geom.*, 4(5):387–421, 1989.

7 José R. Correa, Paul Dütting, Felix A. Fischer, and Kevin Schewior. Prophet Inequalities for I.I.D. Random Variables from an Unknown Distribution. In *Proceedings of the 2019 ACM Conference on Economics and Computation*, pages 3–17, 2019.

8 Nikhil R. Devanur and Thomas P. Hayes. The adwords problem: online keyword matching with budgeted bidders under random permutations. In *ACM Conference on Electronic Commerce*, pages 71–78, 2009.

9 Nikhil R. Devanur, Kamal Jain, Balasubramanian Sivan, and Christopher A. Wilkens. Near optimal online algorithms and fast approximation algorithms for resource allocation problems. In *ACM Conference on Electronic Commerce*, pages 29–38, 2011.

10 Ilias Diakonikolas. Algorithmic High-Dimensional Robust Statistics. Webpage `http://www.iliasdiakonikolas.org/simons-tutorial-robust.html`, 2018. Tutorial at Foundations of Data Science bootcamp.

11 Ilias Diakonikolas, Gautam Kamath, Daniel M. Kane, Jerry Li, Ankur Moitra, and Alistair Stewart. Robustly Learning a Gaussian: Getting Optimal Error, Efficiently. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2683–2702, 2018.

12 Ilias Diakonikolas, Daniel M. Kane, and Alistair Stewart. Learning geometric concepts with nasty noise. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 1061–1073, 2018.

13 Eugene B Dynkin. The optimum choice of the instant for stopping a Markov process. In *Soviet Math. Dokl*, volume 4, pages 627–629, 1963.

14 Hossein Esfandiari, Nitish Korula, and Vahab Mirrokni. Allocation with Traffic Spikes: Mixing Adversarial and Stochastic Models. *ACM Transactions on Economics and Computation (TEAC)*, 6(3-4):14, 2018.

15 Uriel Feige and Joe Kilian. Heuristics for semirandom graph problems. *Journal of Computer and System Sciences*, 63(4):639–671, 2001.

16 Moran Feldman, Ola Svensson, and Rico Zenklusen. A Simple $O(\log \log(\text{rank}))$-Competitive Algorithm for the Matroid Secretary Problem. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015*, pages 1189–1201, 2015.

17 Thomas S Ferguson et al. Who solved the secretary problem? *Statistical science*, 4(3):282–289, 1989.

18 Naveen Garg, Anupam Gupta, Stefano Leonardi, and Piotr Sankowski. Stochastic analyses for online combinatorial optimization problems. In *ACM-SIAM symposium on Discrete algorithms*, pages 942–951, 2008.

19 Oliver Göbel, Martin Hoefer, Thomas Kesselheim, Thomas Schleiden, and Berthold Vöcking. Online independent set beyond the worst-case: Secretaries, prophets, and periods. In *International Colloquium on Automata, Languages, and Programming*, pages 508–519, 2014.

20 Gagan Goel and Aranyak Mehta. Online budgeted matching in random input models with applications to adwords. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, 2008.

21 Anupam Gupta and Marco Molinaro. How the Experts Algorithm Can Help Solve LPs Online. *Math. Oper. Res.*, 41(4):1404–1431, 2016.

22 Guru Prashanth Guruganesh and Sahil Singla. Online Matroid Intersection: Beating Half for Random Arrival. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 241–253, 2017.

23 Thomas Kesselheim, Robert D. Kleinberg, and Rad Niazadeh. Secretary Problems with Non-Uniform Arrival Order. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, Portland, OR, USA, June 14-17, 2015*, pages 879–888, 2015.

24 Thomas Kesselheim, Klaus Radke, Andreas Tönnis, and Berthold Vöcking. An optimal online algorithm for weighted bipartite matching and extensions to combinatorial auctions. In *European Symposium on Algorithms*, pages 589–600. Springer, 2013.

25 Thomas Kesselheim, Klaus Radke, Andreas Tönnis, and Berthold Vöcking. Primal beats dual on online packing LPs in the random-order model. In *Symposium on Theory of Computing, 2014*, pages 303–312, 2014.

26 Robert D. Kleinberg. A multiple-choice secretary algorithm with applications to online auctions. In *ACM-SIAM Symposium on Discrete Algorithms*, 2005.

27 Nitish Korula and Martin Pál. Algorithms for secretary problems on graphs and hypergraphs. In *International Colloquium on Automata, Languages and Programming*, pages 508–520. Springer, 2009.

28 Oded Lachish. O(log log Rank) Competitive Ratio for the Matroid Secretary Problem. In *55th IEEE Annual Symposium on Foundations of Computer Science, Philadelphia, PA, USA, October 18-21*, pages 326–335, 2014.

29 Kevin A. Lai, Anup B. Rao, and Santosh Vempala. Agnostic Estimation of Mean and Covariance. In *IEEE 57th Annual Symposium on Foundations of Computer Science*, 2016.

30 Thodoris Lykouris, Vahab S. Mirrokni, and Renato Paes Leme. Stochastic bandits robust to adversarial corruptions. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, June 25-29, 2018*, pages 114–122, 2018.

31 Mohammad Mahdian, Hamid Nazerzadeh, and Amin Saberi. Allocating online advertisement space with unreliable estimates. In *Proceedings of the 8th ACM conference on Electronic commerce*, pages 288–294. ACM, 2007.

32 Adam Meyerson. Online facility location. In *Foundations of Computer Science, 2001. Proceedings. 42nd IEEE Symposium on*, pages 426–431. IEEE, 2001.

33 Vahab S Mirrokni, Shayan Oveis Gharan, and Morteza Zadimoghaddam. Simultaneous approximations for adversarial and stochastic online budgeted allocation. In *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms*, pages 1690–1701, 2012.

34 Ankur Moitra. Robustness Meets Algorithms (Invited Talk). In *16th Scandinavian Symposium and Workshops on Algorithm Theory, SWAT 2018*, pages 3:1–3:1, 2018.

35 Marco Molinaro. Online and Random-order Load Balancing Simultaneously. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 1638–1650, 2017.

36 John von Neumann. Zur theorie der gesellschaftsspiele. *Mathematische annalen*, 100(1):295–320, 1928.

37 T Parthasarathy. On games over the unit square. *SIAM Journal on Applied Mathematics*, 19(2):473–476, 1970.

38 Aviad Rubinstein. Beyond matroids: secretary problem and prophet inequality with general constraints. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 324–332, 2016.

39 Aviad Rubinstein and Sahil Singla. Combinatorial prophet inequalities. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, 2017.

40 Walter Rudin. *Principles of mathematical analysis*, volume 3. McGraw-hill New York, 1976.

41 Raimund Seidel. Backwards analysis of randomized geometric algorithms. In *New trends in discrete and computational geometry*, volume 10 of *Algorithms Combin.*, pages 37–67. Springer, Berlin, 1993.

A Hard Benchmarks

We show that for the benchmark $V^* := v(g_{max})$, every algorithm has an approximation of at most $O(1/n)$.

▶ **Observation 19** (Lower Bound for $g_{\max}$)**.** *Any randomized algorithm for the single-item Byzantine secretary problem cannot select the highest-value good/green item with probability larger than* $1/(|R|+1)$.

Proof. We use Yao's minimax lemma, so it is enough to construct an input distribution $\mathcal{B}$ for which no deterministic algorithm can achieve an approximation better than $\frac{1}{|R|+1}$. The distribution is as follows. The red elements arrive at random times, that is $(t_{r_1}, t_{r_2}, ...t_{r_{|R|}}) \sim U[0,1]^{|R|}$. The linear ordering among the elements is set such that the red elements are strictly increasing according to their arrival time, or formally: $t_{r_i} > t_{r_j} \implies v(r_i) > v(r_j)$. The maximum element is green and all the other green elements are smaller than all red elements. Formally: $v(g_{max}) > v(r_1) > v(r_2)... > v(r_{|R|}) > v(g_2) > v(g_3) \ldots > v(g_{|G|})$. This fully defines the input distribution.

All the arrival times are distinct with probability 1. Let $\mathcal{K}(t)$ denote the information seen by the algorithm up to and including time t. Partition the probability space according to $S := \{t_{g_{max}}, t_{r_1}, t_{r_2}, ...t_{r_{|R|}}\}$ and $L := (t_{g_2}, t_{g_3}, ...t_{g_{|G|}})$. Let $s_1 < s_2 < ... < s_{|R|+1}$ be the elements of S. Let $M_i := \{t_{g_{max}} = s_i\}$. By definition, we have $Pr[M_i|S,L] = \frac{1}{|R|+1}$. Note that, since the red items arrive in increasing order of value and the green item has maximum value, we have $Pr[M_i|S,L,\mathcal{K}(t)] = Pr[M_j|S,L,\mathcal{K}(t)]$ for all $t \le s_i, s_j$. Therefore, $Pr[M_i|S,L,\mathcal{K}(s_i)] \le \frac{1}{|R|+2-i}$. In other words, there is no way to distinguish the maximum green element from the red elements before it is too late, that is at the time of the green element's arrival. Thus, by a simple inductive argument, the proof is finished. ◀

Using techniques presented in [7], we can extend this result to the value case as well.

B Relaxing the Assumption that n is Known

In this section we extend our results to some settings where n is unknown. Most importantly, observe that all of the results in this paper hold even if n is *known only up to a constant factor* with at most a constant factor degradation in the quality of the result. As a simple example, note that picking a uniformly random element from an n-element sequence when the assumed number of elements is $\tilde{n} \in [n, 2n]$ will select an element $x \in U$ with probability $p_x \in [\frac{1}{2n}, \frac{1}{n}]$, leading to a degradation in the result by a factor of at most 2, which we typically ignore in this paper.

This still leaves us open to the possibility that we do not even know the scale of n. Surprisingly, it is still possible to "guess" $\tilde{n}$ while only incurring a loss of $\widetilde{O}(\log n)$ in the quality, even if there is no prior known upper limit on n.[1] The following claim formalizes this result.

▷ Claim 20. There exists a distribution X over the integers such that for every $n \ge 1$ the probability that the sampled number $\tilde{n} \sim X$ is within a constant factor of n, is at least $1/\widetilde{O}(\log n)$.

[1] By $\widetilde{O}(f(n))$ we mean $f(n) \cdot \text{poly}(\log f(n))$.

Proof. Consider the sequence $a_k := \frac{1}{k(\log_2 k)(\log_2 \log_2 k)^2}$ defined for $k \geq 2$. It is well-known that this sequence converges, i.e., $\sum_{k=2}^{\infty} a_k = O(1)$. A simple way to see this is by noting that a non-negative decreasing sequence $(A_k)_k$ converges if and only if $(2^k A_{2^k})_k$ converges [40, Thm 3.27].

Let $\sum_k^\infty A_k \sim \sum_k^\infty B_k$ be the equivalence relation denoting that $(A_k)_k$ and $(B_k)_k$ either both converge or both diverge. Then the above fact implies that $\sum_k^\infty \frac{1}{k \log_2 k (\log_2 \log_2 k)^2} \sim \sum_k^\infty \frac{1}{k(\log_2 k)^2} \sim \sum_k^\infty \frac{1}{k^2}$, where the last sequence clearly converges.

We can assume without loss of generality that $n \geq 100$ by handling those cases separately. The strategy for guessing the estimate $\tilde{n}$ is now immediate: we sample $\tilde{n}$ from $\mathbb{Z}_{\geq 2}$ according to the distribution $\Pr[\tilde{n} = k] = a_k/Z$ where $Z := \sum_{k=100}^{\infty} a_k = O(1)$. We observe that $\Pr[\tilde{n} \in \langle 2^{k-1} \dots 2^k]] \geq \frac{1}{2} 2^k a_{2^k}/Z = \widetilde{\Omega}(1/k)$. Let k' be the unique index such that $n \in \langle 2^{k'-1} \dots 2^{k'}]$, hence $k' = \Theta(\log n)$. Then $\Pr[\tilde{n} \in \langle 2^{k'} \dots 2^{k'+1}]] = \widetilde{\Omega}(1/k') = \widetilde{\Omega}(1/\log n)$. But also in that case we have that $n \in [\tilde{n}/4, \tilde{n}]$ and we are done. ◁

Finally, consider an important case where the fraction of red elements is bounded away from $1 - \Omega(1)$. This is a reasonable assumption for most applications, e.g., online auctions, where we do not expect that most of the arrivals will be chosen by an adversary. By simply observing the first half of the sequence, i.e., $[0, 1/2\rangle$, we can typically estimate n up to a constant while degrading the expected output of our algorithms by at most a constant factor.

▷ Claim 21. If there is a constant $\varepsilon < 1$ such that the fraction of red elements $\frac{|R|}{|R|+|G|} \leq \varepsilon$ then we can estimate n up to a constant factor by time $t = 1/2$.

Proof. We run a simple preprocessing step to estimate n up to a constant factor by $t = 1/2$. Notice that the expected number of green elements to arrive in the interval $[0, 1/2\rangle$ is $0.5 \cdot |G| = 0.5 \cdot n(1 - \varepsilon) = \Omega(n)$. Since by simple Chernoff bounds this means that w.h.p. we see $\Omega(n)$ elements in the first half, we run a simple algorithm that does not select any element till $t = 1/2$, and then use the number of elements that arrive in $[0, 1/2\rangle$ as an estimate of n. ◁

C Minimax

In this section we argue that an α-payoff (i.e., the probability of selecting the second-max element or better is at least α) known distribution algorithm for the ordinal single-item Byzantine secretary implies an α-payoff algorithm for the general, worst-case input, setting. This can be directly modeled as a two-player game where player A chooses an algorithm $\mathcal{A}$ and player B chooses a distribution over the input instances $\mathcal{B}$. Our coveted result would go along the lines of

$$\sup_{\mathcal{A}} \inf_{\mathcal{B}} K(a, b) = \inf_{\mathcal{B}} \sup_{\mathcal{A}} K(\mathcal{A}, \mathcal{B}),$$

where $K(\mathcal{A}, \mathcal{B})$ denotes the payoff when we run algorithm $\mathcal{A}$ on the input distribution $\mathcal{B}$. The left-hand side denotes the worst-case input setting, while the right-hand side denotes the known distribution setting.

The main challenge in proving such a claim stems from the infiniteness of the set of algorithms and set of input distributions. Indeed, if one makes no finiteness assumption for either A or B, the Minimax property can fail even for relatively well-behaved two-player games [37]. On the other hand if both A and B would be finite, then the result would follow from the classic Von Neumann's Minimax [36].

▶ **Fact 22** (Von Neumann's Minimax). *Let A and B be finite sets. Denote by $\mathcal{D}(A)$ and $\mathcal{D}(B)$ distributions over A and B, respectively. Then for any matrix of values $K : A \times B \to \mathbb{R}$ it holds that*

$$\max_{a\in\mathcal{D}(A)} \min_{b\in\mathcal{D}(B)} K(a,b) = \min_{b\in\mathcal{D}(B)} \max_{a\in\mathcal{D}(A)} K(a,b). \tag{9}$$

The infiniteness of the sets stems from the arrival times being in the infinite set $[0,1]$. To solve this issue, we slightly modify our algorithm by discretizing $[0,1]$. Let $N = n^3, \mathcal{T} := \{\frac{0}{N}, \frac{1}{N}, \frac{2}{N}, \ldots, \frac{N}{N}\}$ and $\Phi : [0,1] \to \mathcal{T}, \Phi(t) := \lfloor N \cdot t \rfloor / N$ be the discretizing function. We modify our algorithm in the following way: apply Φ to the input distribution $\mathcal{B}$, as well as to every arrival time. Note that the elements are presented to the algorithm exactly as before, it just *pretends* they arrive in discrete time steps. We can assume $\Phi(t_e) \neq \Phi(t_g)$ for every $e \in U, g \in G, e \neq g$ (otherwise, we say the algorithm loses), since this happens with at most $n^2/N = o(1)$ probability. Using completely analoguous techniques as in Section 4 we can show this algorithm is $\Omega(1/\log^2 n)$-competitive.

We note that a randomized algorithm is simply a distribution over deterministic algorithms. Hence our second goal is to argue that the number of distinct deterministic algorithms is finite (i.e., bounded by a function of n). To this end we have to specify how we represent them with at least some formality. We identify a deterministic algorithm $\mathcal{A}$ with a function that gets evaluated each time a new element arrives; its parameter is the information history $\mathcal{K}(t)$ (t being the current time) represented in any appropriate format; its output is $\{\bot, \top\}$ representing whether to select the current element. For concreteness, the information history consists of (t_e, π_e) for every element e that arrived before the function call, where $t_e \in \mathcal{T}$ is the discretized arrival time (after applying Φ) and $\pi_e \in [0 \ldots n-1]$ is the relative value order of e with respect to prior arrived elements. The number of distinct histories is bounded by $((N+1)n)^n$, a function of n; therefore the set of deterministic algorithms, i.e., functions from the history to $\{\bot, \top\}$, is also bounded.

We remember that an input distribution is simply a distribution over "pure" inputs. Note that the payoff of a deterministic algorithm for a specific input depends only on the following: $\Phi(t_r)$ for every red element; $\pi \in S_n$, the permutation representing the total order among the elements; and $\pi_t \in S_{|R|}$, the permutation denoting the order in which the red elements arrive (since red elements can have the same discretized arrival time, but an arbitrary order in which they are presented to the algorithm). The above discretization makes the number of pure inputs at most $(n!)^2 \cdot (N+1)^n$, i.e., bounded by a function of n. The reader can refresh their memory about the representation of pure inputs by reviewing the introduction to Section 4.

Finally, for our discretized algorithm, we proved that the set of pure inputs with different payoffs, as well as the number of deterministic algorithms is bounded by a function of n. Therefore, for a fixed n, both numbers are finite. We invoke the Von Neumann's Minimax (Fact 22) to conclude that the best result in the known distribution setting and worst-case input setting are equivalent, recovering the following theorem.

▶ **Theorem 4** (Ordinal Single-item Algorithm). *There is a randomized algorithm for the ordinal Byzantine secretary which selects an element of value at least the second-largest green item with probability $\Omega(1/\log^2 n)$.*

Universal Communication, Universal Graphs, and Graph Labeling

Nathaniel Harms
University of Waterloo, Canada
https://cs.uwaterloo.ca/~nharms
nharms@uwaterloo.ca

Abstract

We introduce a communication model called *universal SMP*, in which Alice and Bob receive a function f belonging to a family $\mathcal{F}$, and inputs x and y. Alice and Bob use shared randomness to send a message to a third party who cannot see f, x, y, or the shared randomness, and must decide $f(x, y)$. Our main application of universal SMP is to relate communication complexity to graph labeling, where the goal is to give a short label to each vertex in a graph, so that adjacency or other functions of two vertices x and y can be determined from the labels $\ell(x), \ell(y)$. We give a universal SMP protocol using $O(k^2)$ bits of communication for deciding whether two vertices have distance at most k in distributive lattices (generalizing the k-Hamming Distance problem in communication complexity), and explain how this implies a $O(k^2 \log n)$ labeling scheme for deciding $\mathsf{dist}(x, y) \leq k$ on distributive lattices with size n; in contrast, we show that a universal SMP protocol for determining $\mathsf{dist}(x, y) \leq 2$ in modular lattices (a superset of distributive lattices) has super-constant $\Omega(n^{1/4})$ communication cost. On the other hand, we demonstrate that many graph families known to have efficient adjacency labeling schemes, such as trees, low-arboricity graphs, and planar graphs, admit constant-cost communication protocols for adjacency. Trees also have an $O(k)$ protocol for deciding $\mathsf{dist}(x, y) \leq k$ and planar graphs have an $O(1)$ protocol for $\mathsf{dist}(x, y) \leq 2$, which implies a new $O(\log n)$ labeling scheme for the same problem on planar graphs.

2012 ACM Subject Classification Mathematics of computing → Probabilistic algorithms; Mathematics of computing → Graph coloring; Mathematics of computing → Trees; Theory of computation → Models of computation; Theory of computation → Communication complexity; Theory of computation → Computational geometry; Theory of computation → Generating random combinatorial structures

Keywords and phrases Universal graphs, graph labeling, distance labeling, planar graphs, lattices, hamming distance

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.33

Related Version https://arxiv.org/abs/1911.03757

Funding *Nathaniel Harms*: This research partially supported by the David R. Cheriton and GO-Bell Graduate Scholarships.

Acknowledgements Thanks to Eric Blais for comments on the structure of this paper; Amit Levi for helpful discussions and comments on the presentation; Anna Lubiw for an introduction to planar graphs and graph labeling; Corwin Sinnamon for comments on distributive lattices; and Sajin Sasy for observing the possible applications to privacy.

1 Introduction

In the simultaneous message passing (SMP) model of communication, introduced by Yao [34], Alice and Bob separately receive inputs x and y to a function f. They send messages $a(x), b(y)$ to a third party, called the referee, who knows f and must output $f(x, y)$ (with high probability) using the messages $a(x), b(y)$. But what if the referee *doesn't* know f? Can they still compute $f(x, y)$? Yes: Alice can include in her message a description of f, and then

the referee knows it; however, if f is restricted, they can sometimes do much better. Here is a simple example: the players receive vertices $x, y \in \{1, \ldots, n\}$ in a graph G of maximum degree 2, and want to decide if (x, y) is an edge in G. Sharing a source of randomness, Alice and Bob randomly label each vertex of G with a number up to 200; Alice sends the label of both neighbors of x and Bob sends the label of y. The referee says *yes* if one of Alice's labels matches the label of y, *no* otherwise. They will be correct with probability at least 99/100, and the referee never needs to learn G. This is also an example where the referee can decide many problems using only one strategy. In this work we will see that more interesting families of graphs, such as trees, planar graphs, and distributive lattices, also exhibit these phenomena, even when we wish to compute *distances* instead of just adjacency.

To study this, we introduce the *universal SMP* model, which operates as follows. Fix some family $\mathcal{F}$ of functions. Alice and Bob receive a function $f \in \mathcal{F}$ and inputs x, y, and they use shared randomness to each send one message to the referee. The referee knows the family $\mathcal{F}$ and the size of the inputs, but doesn't know f, x, y or the shared randomness, and must compute $f(x, y)$ with high probability. By choosing the family $\mathcal{F}$ to be the singleton family, one sees that this model includes standard SMP. As in the earlier example, we will be studying communication problems on graphs, but this is not a significant restriction: every Boolean-valued communication problem f is equivalent to determining adjacency in some graph (use f as the adjacency matrix), so we will treat $\mathcal{F}$ as a family of graphs.

A surprising but intuitive application of universal SMP is that it connects two apparently disjoint areas of study: communication complexity and graph labeling. For a graph family $\mathcal{F}$, the graph labeling problem (introduced by Kannan, Naor, and Rudich [23]) asks how to assign the shortest possible labels $\ell(v)$ to each vertex v of a graph $G \in \mathcal{F}$, so that the adjacency (or some other function [29]) of vertices x, y can be computed from $\ell(x), \ell(y)$ by a decoder that knows $\mathcal{F}$. We observe the following principle (Theorem 1.1):

If there is a (randomized) universal SMP protocol for the graph family $\mathcal{F}$ with communication cost c, then there is a labeling scheme for graphs $G \in \mathcal{F}$ with labels of size $O(c \log n)$, where n is the number of vertices.

Common variants of graph labeling are *distance labeling* [15], where the goal is to compute $\mathsf{dist}(x, y)$ from the labels, and *small-distance* labeling, where the goal is to compute $\mathsf{dist}(x, y)$ if it is at most k and output "$> k$" otherwise [24, 2]. This is similar to the well-studied k-Hamming Distance problem in communication complexity, where the players must decide if their vertices x, y have distance at most k in the Boolean hypercube graph. A natural generalization of the Boolean hypercube is the family of distributive lattices (which also include, for example, the hypergrids). We demonstrate that techniques from communication complexity can be used to obtain new graph labelings, by adapting the k-Hamming Distance protocol of Huang et al. [21] to the universal SMP model, achieving an $O(k^2)$ protocol for computing $\mathsf{dist}(x, y) \leq k$ and the corresponding k-distance labeling scheme with label size $O(k^2 \log n)$. It is interesting to note that, in contrast to the standard application of communication complexity as a method for obtaining lower bounds, we are using it to obtain upper bounds.

Generalizing in another direction, we ask: for which graphs other than the Boolean hypercube can we obtain efficient communication protocols for k-distance? For constant k, k-Hamming Distance can be computed with communication cost $O(1)$; which other graphs admit a constant-cost protocol? To approach this question, we observe that many (but not all) graph families known to have efficient $O(\log n)$ adjacency labeling schemes also admit an $O(1)$ universal SMP protocol for adjacency. Commonly studied families in the adjacency and distance labeling literature are trees [23, 24, 2, 5, 3] and planar graphs [23, 15, 14, 16, 6]. We

study the k-distance problem on these families and find that trees admit an $O(k)$ protocol, while planar graphs admit an $O(1)$ protocol for 2-distance; this implies a new labeling scheme for planar graphs.

Further motivation for the universal SMP model comes from *universal graphs*. Introduced by Rado [30], an induced-universal graph U for a set $\mathcal{F}$ is one that contains each $G \in \mathcal{F}$ as an induced subgraph. An efficient adjacency labeling scheme for a set $\mathcal{F}$ implies a small induced-universal graph for that set [23]. Deterministic universal SMP protocols are equivalent to universal graphs (Theorem 1.7), and we introduce *probabilistic universal graphs* as the analogous objects for randomized univeral SMP protocols. We think probabilistic universal graphs are worthy of study alongside universal graphs, especially since many non-trivial families admit one of *constant-size*.

The universal SMP model is also related to a recent line of work studying communication between parties with imperfect knowledge of each other's "context". The most relevant incarnation of this idea is the recent work [18, 17], who study the 2-way communication model where Alice and Bob receive functions f and g respectively, with inputs x and y, and must compute $f(x, y)$ under the guarantee that f and g are close in some metric. In other words, one party does not have full knowledge of the function to be computed. The universal SMP model provides a framework for studying a similar problem in the SMP setting, where the players know the function but the referee does not; the similarity is especially clear when we define the family $\mathcal{F}$ to be all graphs of distance δ to a reference graph G in some metric (we discuss this situation in more detail at the end of the paper). This could model, for example, a situation where the clients of a service operate in a shared environment but the server does not; or, a situation in which the clients want to keep their shared environment secret from the server, and their inputs secret from each other. This suggests a possible application to privacy and security. A relevant example is private proximity testing (e.g. [27]), where two clients should be notified by the server when they are at distance at most k from each other, without revealing to each other or the server their exact locations.

The Discussion at the end of the paper highlights some interesting questions and open problems related to universal SMP.

1.1 Results

A universal SMP protocol *decides k-distance* for a family $\mathcal{F}$ if for all graphs $G \in \mathcal{F}$ and vertices x, y, the protocol will correctly decide if $\mathsf{dist}(x, y) \leq k$, with high probability. A labeling scheme decides k-distance if $\mathsf{dist}(x, y) \leq k$ can be decided from the labels of x, y. Below, the variable n always refers to the number of vertices in the input graph.

Implicit graph representations. The main principle connecting communication and graph labeling is:

▶ **Theorem 1.1.** *Any graph family $\mathcal{F}$ with universal SMP cost m has an adjacency labeling scheme with labels of size $O(m \log n)$. In particular, if the universal SMP cost for $\mathcal{F}$ is $O(1)$ then $\mathcal{F}$ has an $O(\log n)$ adjacency labeling scheme.*

Adjacency labeling schemes of size $O(\log n)$ are of special interest because $\log n$ is the minimum number of bits required to label each vertex uniquely, and they correspond to *implicit graph representations*, as defined by Kannan, Naor, and Rudich [23] (we omit their requirement that the encoding and decoding be computable in polynomial-time). Section 2.3 elaborates further. To obtain implicit representations, we can relax our requirements:

▶ **Corollary 1.2.** *For any constant c, any graph family $\mathcal{F}$ where each $G \in \mathcal{F}$ has a public-coin 2-way communication protocol computing adjacency with cost c has an implicit representation.*

Distributive & Modular Lattices. Distributive and modular lattices are generalizations of the Boolean hypercube and hypergrids (see Section 3 for definitions). We define a *weakly-universal* SMP protocol as one where the referee shares the randomness of Alice and Bob. For distributive lattices we get the following:

▶ **Theorem 1.3.** *The k-distance problem on the family of distributive lattices has: a weakly-universal SMP protocol with cost $O(k \log k)$; a universal SMP protocol with cost $O(k^2)$; and a size $O(k^2 \log n)$ labeling scheme.*

Modular lattices are a superset of distributive lattices, but they do not admit k-distance protocols with a cost independent of n; we show that any universal SMP protocol (and any labeling scheme) deciding 2-distance must have cost $\Omega(n^{1/4})$ (Theorem 3.14). To our knowledge, there are no known labeling schemes for distributive or modular lattices. Our adjacency labeling scheme (i.e. for $k = 1$) requires $O(n \log n)$ space to store the whole lattice; this can be compared to Munro and Sinnamon [26], who present a data structures of size $O(n \log n)$ for distributive lattices that supports *meet* and *join* operations (and therefore distance queries, due to our Lemma 3.5). However, these are not labelings, so the result is not directly comparable.

Planar graphs and other efficiently-labelable families. When they introduced graph labeling, Kannan, Naor, and Rudich [23] studied trees, low-arboricity graphs (whose edges can be partitioned into a small number of trees), and planar graphs, and interval graphs (whose vertices are intervals in $\mathbb{R}$, with an edge if the intervals intersect), among others. These families have $O(\log n)$ adjacency labeling schemes. Trees, low-arboricity graphs, and planar graphs have constant-cost universal SMP protocols for adjacency. Trees admit an efficient k-distance protocol:

▶ **Theorem 1.4.** *The family of trees has a universal SMP protocol deciding k-distance with cost $O(k)$ and a $O(k \log n)$ labeling scheme deciding k-distance.*

Planar graphs admit an efficient 2-distance protocol, which implies a new 2-distance labeling scheme:

▶ **Theorem 1.5.** *The 2-distance problem on the family of planar graphs has a universal SMP protocol with cost $O(1)$ and a labeling scheme of size $O(\log n)$.*

On the other hand, a universal SMP protocol deciding 2-distance on the family of graphs with arboricity 2 has cost at least $\Omega(\sqrt{n})$ (Proposition 4.4), and a universal SMP protocol deciding adjacency in interval graphs has cost $\Theta(\log n)$ (Proposition 4.5).

Gavoille et al. [15] showed that trees have an $O(\log^2 n)$ labeling allowing $\mathsf{dist}(x, y)$ to be computed exactly from labels of x, y, and gave a matching lower bound; Kaplan and Milo [24] and Alstrup *et al* [2] studied k-distance for trees, with the latter achieveing a $\log n + O(k^2(\log \log n + \log k))$ labeling scheme. For planar graphs, [15] gives a lower bound of $\Omega(n^{1/3})$ for computing distances exactly, and an upper bound of $O(\sqrt{n} \log n)$, which was later improved to $O(\sqrt{n})$ in [16].

Communication Complexity. Our lower bounds are achieved by reduction from the family of all graphs, which has complexity $\Theta(n)$, in contrast to the upper bound of $\lceil \log n \rceil$ for the standard SMP cost of computing adjacency in any graph (since Alice and Bob can send $\lceil \log n \rceil$ bits to identify their vertices).

▶ **Theorem 1.6.** *For the family $\mathcal{G}$ of all graphs, the universal SMP cost of computing adjacency in $\mathcal{G}$ is $\Theta(n)$.*

The basic relationships between universal SMP, standard SMP, and universal graphs are as follows. Below, we use $D^{\|}(\textsc{Adj}(G))$ and $R^{\|}(\textsc{Adj}(G))$ for the deterministic and randomized (standard) SMP cost of computing adjacency on G, and $D^{\mathsf{univ}}(\mathcal{F}), R^{\mathsf{univ}}(\mathcal{F})$ for the deterministic and randomized universal SMP cost for computing adjacency in the family $\mathcal{F}$. We use the term "$\sqsubset$-universal graph" as opposed to "induced-universal" to denote a slightly different object that allows non-injective embeddings (see Section 2 for definitions).

▶ **Theorem 1.7.** *For a set $\mathcal{F}$, the following relationships hold. Let U range over the set of all $\sqsubset$-universal graphs:*

$$\max_{G\in\mathcal{F}} D^{\|}(\textsc{Adj}(G)) \leq D^{\mathsf{univ}}(\mathcal{F}_i) = \min_{U} D^{\|}(\textsc{Adj}(U)) = \min_{U} \lceil \log |U| \rceil \,,$$

with equality on the left iff $\exists H \in \mathcal{F}$ such that $\forall G \in \mathcal{F}$, G can be embedded *in H. For $\widetilde{U}$ ranging over the set of all* probabilistic universal graphs*:*

$$\max_{G\in\mathcal{F}} R^{\|}(\textsc{Adj}(G)) \leq R^{\mathsf{univ}}(\mathcal{F}) \leq \min_{\widetilde{U}} D^{\|}(\textsc{Adj}(\widetilde{U})) \leq O\left(R^{\mathsf{univ}}(\mathcal{F})\right) \,.$$

Randomized and deterministic universal SMP satisfy

$$\Omega\left(\frac{D^{\mathsf{univ}}(\mathcal{F})}{\log n}\right) \leq R^{\mathsf{univ}}(\mathcal{F}) \leq D^{\mathsf{univ}}(\mathcal{F}) \,.$$

The above results on graph labeling are proved through the relationship between randomized and deterministic universal SMP. We obtain this relationship by adapting Newman's Theorem [28], a standard derandomization result in communication complexity. Finally, we note the interesting fact that universal SMP characterizes the gap between standard SMP models where the referee does or does not share the randomness with Alice and Bob:

▶ **Proposition 1.8** (Informal). *Let $\mathcal{F}$ be a family of graphs and let Π be a weakly-universal SMP protocol for $\mathcal{F}$, which defines a distribution over the referee's decision functions F, which we interpret as the adjacency matrices of graphs. Let $\mathcal{U}_\Pi$ be the family on which this distribution is supported. Then, taking the minimum over all such protocols Π,*

$$R^{\mathsf{univ}}_{\epsilon}(\mathcal{F}) = \min_{\Pi} D^{\mathsf{univ}}(\mathcal{U}_\Pi) \,.$$

1.2 Other Related Work

Graph labeling. Randomized labeling schemes for trees have been studied by Fraigniaud and Korman [12], who give a randomized adjacency labeling scheme of $O(1)$ bits per label that has one-sided error (i.e. it can erroneously report that x, y are adjacent when they are not), and they show that achieving one-sided error in the opposite direction requires a randomized labeling with $\Omega(\log n)$ bits. They also give randomized schemes for determining if x is an ancestor of y, but they do not address distance problems. Spinrad's book [33] has a chapter on implicit graphs and Alstrup et al. [6] for a recent survey on adjacency labeling

schemes and induced-universal graphs. We know of no labeling schemes for lattices, but Fraigniaud and Korman [13] recently studied adjacency labeling schemes for posets of low "tree-dimension".

Distance-preserving labeling studies an opposite problem to k-distance labeling, where distances must be accurately reported when they are *above* some threshold D. Recent work includes Alstrup et al. [4].

To our knowledge, k-distance or even 2-distance has not been studied for planar graphs, but there are many results on other types of planar graph labelings with restrictions at distance 2. An example is the *frequency assignment problem* or $L(p,q)$*-labeling* problem, which asks how to construct a labeling ℓ assigning integers $[k]$ to vertices of a planar graph so that $\mathsf{dist}(x,y) \le 1 \implies |\ell(x) - \ell(y)| \ge p$ and $\mathsf{dist}(x,y) \le 2 \implies |\ell(x) - \ell(y)| \ge q$, with various optimization goals. See [7] for a survey.

Uncertain communication. There are several works studying communication problems where the parties do not agree on the function to be computed, starting with Goldreich, Juba, and Sudan [19] who studied communication where parties have different "goals". Canonne et al. [8] study communication in the shared randomness setting where the randomness is shared imperfectly. Haramarty and Sudan [20] study compression (á la Shannon) in situations where the parties do not agree on a common distribution. As mentioned earlier, Ghazi et al. [17] and Ghazi and Sudan [18] study 2-way communication where the parties do not agree on the function to be computed.

1.3 Notation

$[k]$ means $\{1,\dots,k\}$. The letter n always denotes the number of vertices in a graph. We use the notation $\mathbb{1}[E] = 1$ iff the statement E holds, and $\mathbb{1}[E] = 0$ otherwise. For a graph G, $V(G)$ is the set of vertices and $E(G)$ is the set of edges. For vertices x, y, we write $G(x,y) = \mathbb{1}[x, y \text{ are adjacent in } G]$ for the entry in the adjacency matrix of G. For an undirected, unweighted graph G and vertices u, v, $\mathsf{dist}(u,v)$ is the length of the shortest path from u to v.

For any graph G and integer k, we denote by G^k the k*-closure* of G, where two vertices u, v are adjacent iff $\mathsf{dist}(u,v) \le k$ in G; it is convenient to require that each vertex is adjacent to itself in G^k. For a set of graphs $\mathcal{F}$, $\mathcal{F}^k = \{G^k : G \in \mathcal{F}\}$.

$D^{\|}(f)$ is the deterministic SMP cost of the function f and $R^{\|}(f)$ is the randomized SMP cost of the function f, in the model where Alice and Bob share randomness but the deterministic referee does not.

2 Universal Communication and Universal Graphs

In this paper we focus on deciding adjacency. Every Boolean communication problem $f : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$ on finite domains $\mathcal{X}, \mathcal{Y}$ is equivalent to the adjacency problem on the graph G with vertex set $\mathcal{X} \cup \mathcal{Y}$ and $G(u,v) = f(u,v)$. We may either allow self-loops in G if $\mathcal{X} = \mathcal{Y}$ or take G to be bipartite. We will generally permit graphs to have self-loops.

▶ **Definition 2.1.** *A* family of graphs $\mathcal{F} = (\mathcal{F}_i)$ *is a sequence of sets* $\mathcal{F}_i$ *indexed by integers* i*, along with a strictly increasing size function* $n(i)$*, so that* $\mathcal{F}_i$ *is a set of graphs with vertex set* $[n(i)]$*. If* $\mathcal{F}_i$ *has size* $n(i) = i$ *then we write* $\mathcal{F}_n$*.*

▶ **Definition 2.2** (Universal SMP and Variations). *Let $\mathcal{F}$ be a family of graphs with size function n and let Φ be an operation taking size $n(i)$ graphs to size $n(i)$ graphs. Let $c : \mathbb{N} \to \mathbb{N}$ and let $\epsilon > 0$ be a constant. An ϵ-error, cost c sequence of* universal SMP *communication protocols for $\mathcal{F}$ is as follows. For any $i \in \mathbb{N}$, a protocol Π_i for $\mathcal{F}_i$ is a triple (a_i, b_i, F_i) where:*

- *Alice and Bob receive $(G, x), (G, y)$ respectively, where $G \in \mathcal{F}_i$ and $x, y \in V(G) = [n(i)]$;*
- *Alice and Bob share a random string r and compute messages $a_i(r, G, x), b_i(r, G, y) \in \{0,1\}^{c(i)}$, respectively;*
- *For each i, the (deterministic) referee has a function $F_i : \{0,1\}^{c(i)} \times \{0,1\}^{c(i)} \to \{0,1\}$, called the* decision function. *$F_i(a_i(r, G, x), b_i(r, G, y))$ must satisfy:*
  1. *If x, y are adjacent in $\Phi(G)$ then $\mathbb{P}_r[F_i(a_i(r, G, x), b_i(r, G, y)) = 1] > 1 - \epsilon$; and*
  2. *If x, y are not adjacent in $\Phi(G)$ then $\mathbb{P}_r[F_i(a_i(r, G, x), b_i(r, G, y))] < \epsilon$.*

A universal SMP protocol is symmetric *when the functions a_i, b_i computed by Alice and Bob are identical and the function F_i satisfies $F_i(a, b) = F_i(b, a)$ for all messages $a, b \in \{0,1\}^c$. We write $R^{\mathsf{univ}}_\epsilon(\Phi(\mathcal{F}))$ for the communication complexity in the universal SMP model of computing adjacency in graphs $\Phi(\mathcal{F}) = \{\Phi(G) : G \in \mathcal{F}\}$, where ϵ is the allowed probability of error. We write $R^{\mathsf{univ}}(\Phi(\mathcal{F}))$ for $R^{\mathsf{univ}}_{1/3}(\Phi(\mathcal{F}))$. If no operation Φ is specified, it is assumed to be the identity.*

It is also convenient to define a weakly-universal SMP *protocol as a universal SMP protocol where the referee can see the shared randomness, so the choice function is of the form $F_i(r, a(r, G, x), b(r, G, y))$ for random seed r, graph $G \in \mathcal{F}$, and $x, y \in V(G)$. We denote the ϵ-error complexity in this model with $R^{\mathsf{weak}}_\epsilon(\Phi(\mathcal{F}))$.*

Finally, we write $D^{\mathsf{univ}}(\Phi(\mathcal{F}))$ for the deterministic *universal SMP complexity.*

▶ Remark 2.3. We include the operator Φ in the definition to emphasize that the players are given the original graph G, not the graph $\Phi(G)$; for example, the players are not given G^k (from which it may be difficult to compute G), but are instead given G.

2.1 Deterministic Universal Communication and Universal Graphs

We will show that a deterministic universal SMP protocol is equivalent to an *embedding* into a $\sqsubset$-universal graph, which we we define using the following notion of embedding (following the terminology of Rado [30]):

▶ **Definition 2.4.** *For graphs G, H, a mapping $\phi : V(G) \to V(H)$ is an* embedding *iff $\forall u, v \in V(G)$, $G(u, v) = H(\phi(u), \phi(v))$. If such a mapping exists we write $G \sqsubset H$.*

For a set of graphs $\mathcal{F}_i$, a graph U is $\sqsubset$-universal *if $\forall G \in \mathcal{F}_i, G \sqsubset U$; i.e. $\forall G \in \mathcal{F}_i$ there exists an embedding $\phi_G : V(G) \to V(U)$. For a family of graphs $\mathcal{F} = (\mathcal{F}_i)$, a sequence $U = (U_i)$ is a* $\sqsubset$-universal graph sequence *if for each i, U_i is $\sqsubset$-universal for $\mathcal{F}_i$.*

Define an equivalence relation on $V(G)$ by $u \equiv v$ iff $\forall w \in V(G), G(u, w) = G(v, w)$, i.e. u, v have identical rows in the adjacency matrix. For a graph G, define the $\equiv$-reduction $G^{\equiv}$ as a graph on the equivalence classes $\mathcal{C}$ of $V(G)$ with $U, W \in \mathcal{C}$ adjacent iff $\exists u \in U, w \in W$ such that u, w are adjacent.

An embedding is not the same as a homomorphism since we must map non-edges to non-edges, and $G \sqsubset H$ is not the same as G being an induced subgraph of H since the mapping is not necessarily injective. Therefore a universal graph by our definition is not the same as an induced-universal graph, where G must exist as an induced subgraph. We could for example map the path $a - b - c \mapsto a' - b' - a'$. This difference between definitions is captured by the $\equiv$ relation between vertices. It is necessary to allow self-loops, otherwise the $\sqsubset$ relation is not transitive. The important properties of $\sqsubset, \equiv$, and $\equiv$-reductions are stated in the next proposition; the proofs are routine and for completeness are included in the appendix. The relation $\simeq$ is the isomorphism relation on graphs.

▶ **Proposition 2.5.** *The following properties are satisfied by the $\sqsubset$ relation, the $\equiv$ relation, and $\equiv$-reductions:*

1. *$\sqsubset$ is transitive.*
2. *For any graph G and $u, v \in V(G)$, $u \equiv v$ iff there exists H and an embedding $\phi : G \to H$ such that $\phi(u) = \phi(v)$.*
3. *For any graph G, $(G^{\equiv})^{\equiv} \simeq G^{\equiv}$.*
4. *For any graph G, $G \sqsubset G^{\equiv}$ and $G^{\equiv} \sqsubset G$.*
5. *For any graphs G, H, $G \sqsubset H$ iff $G^{\equiv} \sqsubset H^{\equiv}$.*
6. *For any graphs G, H, $G^{\equiv} \sqsubset H^{\equiv}$ iff $G^{\equiv}$ is an induced subgraph of $H^{\equiv}$.*

These properties allows us to prove relationships between the standard SMP model, deterministic universal SMP, and $\sqsubset$-universal graphs. First we show that deterministic universal SMP protocols can always be made symmetric[1].

▶ **Proposition 2.6.** *If Π is a deterministic universal SMP protocol for the set $\mathcal{F}$, then there exists a deterministic universal SMP protocol Π' that is symmetric and has the same cost as Π.*

Proof. Let $G \in \mathcal{F}$ and let $a, b : V(G) \to \{0,1\}^m$ be the encoding functions for G and F the decision function for graphs of size $|G|$. The restriction of b to the domain $V(G^{\equiv}) \to \{0,1\}^m$ is injective so it has an inverse $b^{-1} : \text{image}(b) \to V(G^{\equiv})$ that satisfies $b^{-1}b(x) \equiv x$; the same holds for a, a^{-1}. Define the encoding function $b' : V(G) \to \{0,1\}^m$ as $b' = ab^{-1}b$ and define the decision function $F'(p,q) = F(p, ba^{-1}(q))$. Then for any $x, y \in V(G), F'(a(x), b'(y)) = F(a(x), ba^{-1}ab^{-1}b(y)) = F(a(x), b(y)) = G(x,y)$ so this is a valid protocol. Since $\text{image}(b') \subseteq \text{image}(a)$ we can write $b'(x) = aa^{-1}b'(x) = aa^{-1}ab^{-1}b(x) = a(x)$ for every x so $b' = a$, thus $F'(a(x), a(y)) = G(x,y) = G(y,x) = F'(a(y), a(x))$ so the protocol is symmetric. ◀

The standard deterministic SMP complexity measure can be expressed in terms of $\equiv$-reductions:

▶ **Proposition 2.7.** *For all graphs G, $D^{\|}(\textsc{Adj}(G)) = \lceil \log |G^{\equiv}| \rceil$.*

Proof. It is well-known that for any function $f : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$, $D^{\|}(f) = \lceil \log \min(r, c) \rceil$ where r is the number of distinct columns in the communication matrix of f, and c is the number of distinct rows [34]. The communication matrix of the function $\textsc{Adj}(G)$ is the adjacency matrix of G, which is symmetric, and two rows (or columns) indexed by u, v are distinct iff $u \not\equiv v$; so the number of distinct rows is the size of $G^{\equiv}$. ◀

The analogous fact for universal SMP is that the deterministic universal SMP cost is determined by the size of the smallest universal graph.

▶ **Proposition 2.8.** *For any graph family $\mathcal{F} = (\mathcal{F}_i)$,*

$$D^{\mathsf{univ}}(\mathcal{F}_i) = \min_U \{ \lceil \log |U^{\equiv}| \rceil : \forall G \in \mathcal{F}_i, G \sqsubset U^{\equiv} \}.$$

Proof. Let U be any graph such that $G \sqsubset U^{\equiv}$ for all $G \in \mathcal{F}_i$ and for each $G \in \mathcal{F}_i$ let g be the embedding $G \to U^{\equiv}$. Consider the protocol where on inputs $(G, x), (G, y)$, Alice and Bob send $g(x), g(y)$ using $\lceil \log |U^{\equiv}| \rceil$ bits and the referee outputs $U^{\equiv}(g(x), g(y))$. This is correct by definition so $D^{\mathsf{univ}}(\mathcal{F}_i) \leq \lceil \log |U^{\equiv}| \rceil$.

[1] Note that this does not imply that every deterministic SMP protocol is symmetric, since in this paper we are only concerned with adjacency on an undirected graph, for which the communication matrix is symmetric. This proposition shows that for symmetric communication matrices, the deterministic SMP protocol is symmetric.

Now suppose there is a protocol Π for $\mathcal{F}_i$ with cost c and decision function F_i, and let $G \in \mathcal{F}_i$. By Proposition 2.6 we may assume that on inputs $(G, x), (G, y)$ Alice and Bob share the encoding function $g : V(G) \to \{0,1\}^c$. Let U be the graph with vertices $\{0,1\}^c$ and $U(u, v) = F(u, v)$. Then $U(g(x), g(y)) = F(g(x), g(y)) = G(x, y)$ so $G \sqsubset U \sqsubset U^{\equiv}$ (by transitivity). Now $|U^{\equiv}| \leq 2^c$ so $c \geq \log |U^{\equiv}|$. ◀

It is easy to see that $D^{\|}$ can be used as a lower bound on D^{univ} but such lower bounds are tight only when the family $\mathcal{F}$ is essentially a "trivial" family of equivalent graphs.

▶ **Lemma 2.9.** *For any family $\mathcal{F} = (\mathcal{F}_i)$, let $U = (U_i)$ be the smallest $\sqsubset$-universal graph sequence for $\mathcal{F}$. Then*

$$\max_{G \in \mathcal{F}_i} D^{\|}(\textsc{Adj}(G)) \leq D^{\mathsf{univ}}(\mathcal{F}_i) = D^{\|}(\textsc{Adj}(U_i)),$$

with equality holding on the left iff $\exists H \in \mathcal{F}_i$ such that $\forall G \in \mathcal{F}_i, G^{\equiv} \sqsubset H^{\equiv}$.

Proof. The equality on the right holds by the two prior propositions. The lower bound follows from the fact that any protocol Π_i for $\mathcal{F}_i$ in the universal model can be used as a protocol in the SMP model. Now we must show the equality condition. Let $U \in \mathcal{F}_i$ be a graph maximizing $|U^{\equiv}|$ over all graphs in $\mathcal{F}_i$, and suppose $D^{\mathsf{univ}}(\mathcal{F}_i) = \max_{G \in \mathcal{F}_i} D^{\|}(\textsc{Adj}(G)) = \max_{G \in \mathcal{F}_i} \lceil \log |G^{\equiv}| \rceil = \lceil \log |U^{\equiv}| \rceil$, so $\lceil \log |U^{\equiv}| \rceil = \min\{\lceil \log |H^{\equiv}| \rceil : \forall G \in \mathcal{F}_i, G \sqsubset H^{\equiv}\}$. Then there exists H such that $U^{\equiv} \sqsubset H^{\equiv}$ and $|U^{\equiv}| = |H^{\equiv}|$. Since $U^{\equiv}$ is an induced subgraph of $H^{\equiv}$ and $|U^{\equiv}| = |H^{\equiv}|$ we must have $U^{\equiv} \simeq H^{\equiv}$ so $\forall G \in \mathcal{F}_i, G^{\equiv} \sqsubset U^{\equiv}$. ◀

2.2 Randomized Universal Communication

Just as deterministic universal communication is equivalent to embedding a family into a universal graph, we will define probabilistic universal graphs and show that they are tightly related to universal communication with shared randomness.

▶ **Definition 2.10.** *For graphs G, H, a random mapping $\phi : V(G) \to V(H)$ (i.e. a distribution over such mappings) is an* ϵ-error embedding *iff $\forall u, v \in V(G)$,*

$$\mathop{\mathbb{P}}_{\phi}[G(u, v) = H(\phi(u), \phi(v))] > 1 - \epsilon.$$

We will write $G \sqsubset_\epsilon H$ if there exists an ϵ-error embedding $G \to H$. A graph U is ϵ-error universal *for a set of graphs S if $\forall G \in S, G \sqsubset_\epsilon U$. $U = (U_i)$ is an ϵ-error universal graph sequence for the family $\mathcal{F} = (\mathcal{F}_i)$ if for each i, U_i is ϵ-error universal for $\mathcal{F}_i$.*

In the randomized setting we obtain equivalence (up to a constant factor) between universal SMP protocols and probabilistic universal graphs.

▶ **Lemma 2.11.** *For any graph family $\mathcal{F} = (\mathcal{F}_i)$ and any $\epsilon > 0$, if there exists a ϵ-error universal SMP protocols for $\mathcal{F}$ with cost $c(i)$, then there exists a 2ϵ-error symmetric universal SMP protocols for $\mathcal{F}$ with cost at most $2c(i)$.*

Proof. On input $G \in \mathcal{F}_i, x, y \in V(G)$, and random string r, Alice and Bob send the concatentations $g_r(x) := a_i(r, G, x)b_i(r, G, x)$ and $g_r(y) := a_i(r, G, y)b_i(r, G, y)$. Then the referee computes

$$F'_i(g_r(x), g_r(y)) = \max\{F_i(a_i(r, G, x), b_i(r, G, y)), F_i(a_i(r, G, y), b_i(r, G, x))\}.$$

It is clear that F'_i is symmetric. If x, y are adjacent then

$$\mathop{\mathbb{P}}_{r}[F'_i(g_r(x), g_r(y)) = 0] \le \mathop{\mathbb{P}}_{r}[F_i(a_i(r, G, x), b_i(r, G, y)) = 0] < \epsilon\,,$$

and if x, y are not adjacent then, by the union bound,

$$\begin{aligned}&\mathop{\mathbb{P}}_{r}[F'_i(g_r(x), g_r(y)) = 1]\\ &\qquad \le \mathop{\mathbb{P}}_{r}[F_i(a_i(r, G, x), b_i(r, G, y)) = 1] + \mathop{\mathbb{P}}_{r}[F_i(a_i(r, G, y), b_i(r, G, x)) = 1] < 2\epsilon\,.\end{aligned}$$ ◀

Applying this symmetrization, we get a relationship between universal SMP protocols and probabilistic universal graphs.

▶ **Lemma 2.12.** *Let $\mathcal{F} = (\mathcal{F}_i)$ be a graph family and $\epsilon > 0$. Then*

1. *There is an ϵ-error universal graph sequence of size at most $2^{2R^{\mathsf{univ}}_{\epsilon/2}(\mathcal{F})}$; and*
2. *If there is an ϵ-error universal graph sequence of size $c(i)$ then $R^{\mathsf{univ}}_{\epsilon}(\mathcal{F}) \le \lceil \log c \rceil$.*

Proof. If Π_i is an ϵ-error symmetric universal protocol for $\mathcal{F}_i$ then there exists a function F_i such that for every $G \in \mathcal{F}_i$ there is a random g such that $\mathop{\mathbb{P}}_{g}[F_i(g(x), g(y)) \ne G(x, y)] < \epsilon$. Using F_i as an adjacency matrix, we get a graph U_i of size at most 2^c, where c is the cost of Π_i, such that for all $G \in \mathcal{F}_i, G \sqsubset_\epsilon U_i$. Then $U = (U_i)$ is an ϵ-error probabilistic universal graph sequence. By Lemma 2.11 we obtain an ϵ-error symmetric protocol with cost $2R^{\mathsf{univ}}_{\epsilon/2}(\mathcal{F})$, so we have proved the first conclusion. The second conclusion follows by definition. ◀

The basic relationships to standard SMP models follow essentially by definition and from the above lemma.

▶ **Lemma 2.13.** *Let $\mathcal{F}$ be any graph family and let $\epsilon > 0$. Let $U = (U_i)$ be an $\sqsubset$-universal graph sequence for $\mathcal{F}$, and $\widetilde{U} = (\widetilde{U}_i)$ an ϵ-error universal graph sequence. Then*

$$\begin{aligned}\max_{G \in \mathcal{F}_i} R^{\|}_{\epsilon}(\textsc{Adj}(G)) &\le R^{\mathsf{univ}}_{\epsilon}(\mathcal{F}_i) \le D^{\|}(\textsc{Adj}(\widetilde{U}_i))\\ &\le 2R^{\mathsf{univ}}_{\epsilon/2}(\mathcal{F}_i) \;\; \textit{and} \;\; R^{\mathsf{univ}}_{\epsilon}(\mathcal{F}_i) \le R^{\|}_{\epsilon}(\textsc{Adj}(U_i))\,.\end{aligned}$$

Proof. The inequalities on the left follow the definitions and from the above lemma. On the right, we can obtain a universal SMP protocol by choosing for each $G \in \mathcal{F}_i$ a (deterministic) embedding $g : G \to U_i$ and then using the randomized SMP protocol for $\textsc{Adj}(U_i)$. ◀

Universal graphs describe an interesting relationship between weakly-universal and universal SMP protocols (and therefore between standard SMP protocols where the referee does and does not share the randomness); namely, the optimal universal protocol is obtained by finding the smallest universal graph for the family of protocol graphs (decision functions) defined by a weakly-universal protocol.

▶ **Proposition 1.8** (Restated). *Let $\mathcal{F}$ be a family of graphs, let $\epsilon > 0$, and let W_ϵ be the set of all ϵ-error weakly-universal SMP protocols for $\mathcal{F}$. For each $\Pi \in W_\epsilon$ let $\mathcal{U}_\Pi = (\mathcal{U}_{\Pi,i})$ be the family of graphs $\mathcal{U}_{\Pi,i} = \{F_i(r, \cdot, \cdot) : r \text{ is a random seed for } \Pi\}$ where F_i is the decision function of Π. Then*

$$R^{\mathsf{univ}}_{\epsilon}(\mathcal{F}) = \min_{\Pi \in W_\epsilon} D^{\mathsf{univ}}(\mathcal{U}_\Pi)\,.$$

Proof. Let $\Pi \in W_\epsilon$; we will construct a universal SMP protocol as follows. On input $(G,x),(G,y)$, Alice and Bob use shared randomness r to simulate Π and obtain vertices $a(r,G,x), b(r,G,y)$ in some graph $U_r \in \mathcal{U}_\Pi$ with $\mathbb{P}_r[U_r(a(r,G,x),b(r,G,y)) \neq G(x,y)] < \epsilon$. They now simulate the deterministic universal SMP protocol, i.e. an embedding $\phi : V(U_r) \to U'$ for some graph U' that is $\sqsubset$-universal for $\{U_r\}$, and send $\phi(a(r,G,x)), \phi(b(r,G,y))$ to the referee who computes $U'(\phi(a(r,G,x)),\phi(b(r,G,x))) = U_r(a(r,G,x),b(r,G,y))$.

Now let Π be an ϵ-error universal SMP protocol. Then $\Pi \in W_\epsilon$ and for each i, $\mathcal{U}_{\Pi,i} = \{U_i\}$, where U_i is the graph of the decision function. $D^{\mathsf{univ}}(\mathcal{U}_\Pi) \leq \lceil \log |U_i| \rceil$, which is the cost of Π, so $\min_{\Pi \in W_\epsilon} D^{\mathsf{univ}}(\mathcal{U}_\Pi) \leq R^{\mathsf{univ}}_\epsilon(\mathcal{F})$. ◀

Newman's Theorem for public-coin randomized (2-way) protocols is a classic result that gives a bound on the number of uniform random bits required to compute a function $f : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$ in terms of the size of the input domain [28]. In the universal model, the input size can be very large since the graph (function) itself is part of the input. However, the shared part of the input does not contribute to the number of random bits required in the universal SMP model.

▶ **Lemma 2.14** (Newman's Theorem for universal SMP). *Let $\epsilon, \delta > 0$ and suppose there is an ϵ-error universal SMP protocol Π for the family $\mathcal{F} = (\mathcal{F}_i)$. Then there is an $(\epsilon+\delta)$-error universal SMP protocol for the family $\mathcal{F}$ that uses at most $\log\log\left(n(i)^{O(\epsilon/\delta^2)}\right)$ bits of randomness and has the same communication cost.*

Proof. Fix i, let F be the deterministic decision function for $\mathcal{F}_i$, and let $a(r,\cdot,\cdot), b(r,\cdot,\cdot)$ be Alice and Bob's encoding functions for the random seed r. For $G \in \mathcal{F}_i$ and $x,y \in V(G)$ we will say a seed r is *bad* for G,x,y if $F(a(r,G,x),b(r,G,y)) \neq G(x,y)$, and we will call this event $\mathsf{bad}(G,x,y,r)$.

Let $r_1,\dots,r_m$ be independent random seeds, and let $i \sim [m]$ be uniformly random, where $m > \frac{3\epsilon}{\delta^2}\ln(n^2)$. Then for every G, the expected number of vertex pairs x,y for which the strings $r_1,\dots,r_m$ fail is

$$\begin{aligned}
&\mathop{\mathbb{E}}_{r_1,\dots,r_m}\left[\sum_{x,y} \mathbb{1}\left[\mathop{\mathbb{P}}_{i\sim[m]}[\mathsf{bad}(G,x,y,r_i)] > \epsilon+\delta\right]\right] \\
&\quad\leq n^2 \max_{x,y} \mathop{\mathbb{E}}_{r_1,\dots,r_m}\left[\mathbb{1}\left[\mathop{\mathbb{P}}_{i}[\mathsf{bad}(G,x,y,r_i)] > \epsilon+\delta\right]\right] \\
&\quad= n^2 \max_{x,y} \mathop{\mathbb{P}}_{r_1,\dots,r_m}\left[\mathop{\mathbb{P}}_{i}[\mathsf{bad}(G,x,y,r_i)] > \epsilon+\delta\right] \\
&\quad= n^2 \max_{x,y} \mathop{\mathbb{P}}_{r_1,\dots,r_m}\left[\sum_{i=1}^{m} \mathbb{1}\left[\mathsf{bad}(G,x,y,r_i)\right] > m(\epsilon+\delta)\right].
\end{aligned}$$

The sum has mean $\mu = \sum_{i=1}^m \mathop{\mathbb{E}}_{r_i}[\mathbb{1}[\mathsf{bad}(G,x,y,r_i)]] < m\epsilon$, so by the Chernoff bound, the probability is at most

$$\begin{aligned}
&n^2 \mathop{\mathbb{P}}_{r_1,\dots,r_m}\left[\sum_{i=1}^{m} \mathbb{1}\left[\mathsf{bad}(G,x,y,r_i)\right] > (1+m\delta/\mu)\mu\right] \\
&\quad\leq n^2 \exp\left(-\frac{m^2\delta^2}{3\mu}\right) \leq n^2\exp\left(-\frac{m\delta^2}{3\epsilon}\right) < 1\,.
\end{aligned}$$

Since the expected number of pairs x,y where choosing $i \sim [m]$ fails with probability more than $\epsilon+\delta$ is less than 1, there must be some values of $r_1,\dots,r_m$ with no bad pairs for G. So

for every $G \in \mathcal{F}_i$ we may choose $r_1, \dots, r_m$ so that choosing i uniformly at random is the only random step; since $m = \frac{6\epsilon}{\delta^2} \ln n = \log n^{O(\epsilon/\delta^2)}$ this requires at most $\log m = \log\log\left(n^{O(\epsilon/\delta^2)}\right)$ random bits. ◀

With this result, we can conclude the proof of Theorem 1.7 in the next lemma.

▶ **Lemma 2.15.** *For any family $\mathcal{F} = (\mathcal{F}_i)$ with size function $n(i)$,*

$$\Omega\left(\frac{D^{\mathsf{univ}}(\mathcal{F}_i)}{\log n(i)}\right) \le R^{\mathsf{univ}}(\mathcal{F}_i) \le D^{\mathsf{univ}}(\mathcal{F}_i)\,.$$

Proof. The upper bound is clear, so we prove lower bound. Let $\Pi = (\Pi_i)$ be a sequence of randomized universal SMP protocols for $\mathcal{F}$. By Newman's theorem, we may assume that Π_i uses at most $\log\log n(i)^c$ random bits for some constant c and has error probability $3/8$. Let F_i be the decision function of Π_i, let $m(i)$ be the cost of Π_i, and let $k = \lceil c \log n(i) \rceil$. To obtain a deterministic protocol, we can define the decision function F_i' on messages of $k \cdot m(i)$ bits as $F_i'(a_1, b_1, a_2, b_2, \dots, a_k, b_k) = \text{majority}(F_i(a_j, b_j))_j$. Alice and Bob iterate over all $k = 2^{\log\log n(i)^c}$ random strings r and send $a(r, G, x), b(r, G, y)$ for each. Since the probability of error is at most $3/8$ when r is uniform, at least $5k/8 > k/2$ of the functions $F_i(a_j, b_j)$ will give the correct answer. This proves that $D^{\mathsf{univ}}(\mathcal{F}_i) = O(R^{\mathsf{univ}}(\mathcal{F}_i) \log n(i))$. ◀

In this paper we show lower bounds for a family $\mathcal{F}$ by giving embeddings of an arbitrary graph G into $\mathcal{F}$, so we need to know the complexity of the family $\mathcal{G} = (\mathcal{G}_n)$ of all graphs with n vertices. For our purposes, it is convenient to require that each graph $G \in \mathcal{G}_n$ has $G(u, u) = 1$ for all u (i.e. all self-loops are present). However, since equality can be checked with cost $O(1)$, the presence or absence of self-loops does not affect the complexity.

▶ **Theorem 1.6** (Restated). $R^{\mathsf{univ}}(\mathcal{G}) = \Theta(n)$.

Proof. For the upper bound, consider the (deterministic) protocol where on input G, x, y, Alice and Bob send x and y and the respective rows of the adjacency matrix of G. This has cost $n + \lceil \log n \rceil = O(n)$ and the referee can determine $G(x, y)$ by finding y in the row sent by Alice.

Let Π be any protocol for $\mathcal{G}_n$ with cost c. By Lemma 2.11, we may assume that Π is symmetric. Let F be the decision function for graphs on n vertices and let $G \in \mathcal{G}_n$ with vertex set $[n]$. Π defines a distribution over functions $g : [n] \to \{0,1\}^c$ so that for all x, y, $\mathbb{P}_g[F(g(x), g(y)) \neq G(x, y)] < \epsilon$. Therefore, for x, y drawn uniformly from $[n]$, $\mathbb{E}_{f,x,y}[\mathbb{1}[F(f(x), f(y)) \neq G(x, y)]] < \epsilon$. Therefore, for every graph $G \in \mathcal{G}_n$ there is a function f_G such that for $x, y \sim [n]$ uniformly at random, $\mathbb{P}_{x,y}[F(f_G(x), f_G(y)) \neq G(x, y)] < \epsilon$. Write $N = \binom{n}{2}$. There are at most 2^{cn} functions $[n] \to \{0,1\}^c$ and there are 2^N simple graphs on $[n]$ so there is some function $f : [n] \to \{0,1\}^c$ where the number of graphs G such that $f_G = f$ is at least $\frac{2^N}{2^{cn}} = 2^{N-cn}$. Let G, G' be any two such graphs. Then

$$\begin{aligned}&\mathbb{P}_{x,y\sim[n]}[G(x, y) \neq G'(x, y)]\\ &\quad\le \mathbb{P}_{x,y\sim[n]}[G(x, y) \neq F(f(x), f(y)) \text{ or } G'(x, y) \neq F(f(x), f(y))] < 2\epsilon\,.\end{aligned}$$

So G, G' differ on at most $2\epsilon N$ pairs. However, the largest number of graphs that differ from any graph G on at most $2\epsilon N$ pairs of vertices is at most

$$\sum_{k=0}^{2\epsilon N} \binom{N}{k} \le 2\epsilon N \binom{N}{2\epsilon N} \le \epsilon N \left(\frac{eN}{2\epsilon N}\right)^{2\epsilon N} = 2^{2\epsilon N \log(e/2\epsilon) + \log(2\epsilon N)}\,.$$

Therefore we must have

$$N - cn \leq 2\epsilon N \log(e/2\epsilon) + \log(2\epsilon N)$$

so $c = \Omega(n)$. ◀

Recall the example in the first paragraph of the introduction, for which we observed that a single decision function would work for many problems. We now make a note about this phenomenon. A communication protocol for a graph family $\mathcal{F} = (\mathcal{F}_i)$ is really a sequence of protocols, one for each set $\mathcal{F}_i$ of graphs with $n(i)$ vertices. Our next proposition addresses the uniformity of the sequence of protocols, that is, the question of how the protocols are related to one another as the size of the input grows. In general, we ask the question: If the family $\mathcal{F}$ has some relationship between $\mathcal{F}_i$ and $\mathcal{F}_{i+1}$, what does this imply about the relationship between the protocols for i and $i+1$? The families of graphs we study in this paper have constant-cost protocols and they are also *upwards families*, which we define next. These families have enough structure so that there exists a single, one-size-fits-all probabilistic universal graph, into which all graphs can be embedded regardless of their size; in other words, the referee can be ignorant not only of the graph G and vertices x, y, but also of the *size* of the graph, without increasing the cost of the protocol.[2]

▶ **Definition 2.16.** *We call a graph family $\mathcal{F} = (\mathcal{F}_i)$ an* upwards family *if for every i and every $G \in \mathcal{F}_i$ there exists $G' \in \mathcal{F}_{i+1}$ such that G is an induced subgraph of G'.*

Many graph families are upwards families, for example: bounded-degree graphs, bounded-arboricity graphs, planar graphs, and transitive reductions of distributive lattices.

▶ **Proposition 2.17.** *If $\mathcal{F}$ is an upwards graph family with an ϵ-error randomized universal graph sequence $U = (U_i)$ satisfying $|V(U_i)| \leq c$ for some constant c (which may depend on ϵ), then there exists a graph U^* of size c such that $\forall G \in \mathcal{F}, G \sqsubset_\epsilon U^*$. Furthermore, for any $i < j$ and any $G \in \mathcal{F}_i$, there exists $G' \in \mathcal{F}_j$ with ϵ-error embedding $g' : V(G') \to V(U^*)$ such that G is an induced subgraph of G' and the restriction of g' to the domain $V(G)$ is an ϵ-error embedding $V(G) \to V(U^*)$.*

Proof. Let $G \in \mathcal{F}_i$ and let $G' \in \mathcal{F}_{i+1}$ be such that G is an induced subgraph of G'. Let $g' : V(G') \to V(U_{i+1})$ the random function determined by the randomized universal graph sequence. Then g' restricted to the domain $V(G) \subset V(G')$ satisfies

$$\mathop{\mathbb{P}}_{g'}[U_{i+1}(g'(x), g'(y)) = G(x,y)] = \mathop{\mathbb{P}}_{g'}[U_{i+1}(g'(x), g'(y)) = G'(x,y)] > 1 - \epsilon\,.$$

Therefore we may replace U_i with U_{i+1} in the sequence, for any i.

Since each U_i has size at most c, there are at most 2^{c^2} graphs U_i appearing in the sequence U. Thus there is some graph U^* that occurs an infinite number of times in the sequence. For every i there exists $j > i$ such that $U_j = U^*$. By applying the above argument, we may replace U_i with $U_j = U^*$ in the sequence. We arrive at the sequence $U' = (U'_i)$ with $U'_i = U^*$ for every i. ◀

[2] Any family $\mathcal{F}$ with a constant-cost protocol can be turned into a protocol ignorant of the size by requiring that Alice and Bob tell the referee which of the 2^{c^2} possible decision functions to use, where $c = 2^{R^{\text{univ}}(\mathcal{F})}$.

2.3 Implicit Graph Representations and Induced-Universal Graphs

Kannan, Naor, and Rudich [23] call a family of graphs an *implicit* graph family if each of the n vertices can be given a label of $O(\log n)$ bits so that adjacency can be determined from the labels of two vertices. They observe that an implicit encoding gives an upper bound on the size of an *induced-universal graph*. We define these terms below in slightly more generality (and omit the requirement that encoding and decoding be done in polynomial time):

▶ **Definition 2.18.** *Let $\mathcal{F} = (\mathcal{F}_i)$ be a graph family and $m(i)$ a function of the graph size. The family $\mathcal{F}$ has an* m-implicit *encoding if $\forall i, \exists F_i : \{0,1\}^{m(i)} \times \{0,1\}^{m(i)} \to \{0,1\}$ such that F_i is symmetric and $\forall G \in \mathcal{F}_i, \exists g : V(G) \to \{0,1\}^{m(i)}$ satisfying $\forall x, y \in V(G), F_i(g_i(x), g_i(y)) = G(x, y)$.*

For a graph family $\mathcal{F} = (\mathcal{F}_i)$, an induced-universal graph sequence *is a sequence $U = (U_i)$ such that for each i and all $G \in \mathcal{F}_i$, G is an induced sugraph of U_i.*

Our notion of $\sqsubset$-universal graphs differs from induced-universal graphs, since the embedding relation $G \sqsubset U_i$ allows non-injective mappings (two vertices of G may be mapped to the same vertex in U_i). This difference accounts for the extra factor $n(i)$ in the next theorem.

▶ **Theorem 2.19** ([33])**.** *Let $\mathcal{F} = (\mathcal{F}_i)$ be a graph family with size $n(i)$. If there exists an m-implicit encoding of $\mathcal{F}$ there is an induced-universal graph sequence $U = (U_i)$ such that $|U_i| \leq n(i)2^{m(i)} = 2^{m(i)+\log n(i)}$.*

Due to the fact that a deterministic universal SMP protocol may always be assumed to be symmetric (Proposition 2.6), it follows by definition and from Lemma 2.15 that:

▶ **Theorem 1.1** (Restated)**.** *A graph family $\mathcal{F} = (\mathcal{F}_i)$ is m-implicit iff $D^{\mathsf{univ}}(\mathcal{F}_i) \leq m(i)$ for every i. Therefore, $\mathcal{F}$ is $O(R^{\mathsf{univ}}(\mathcal{F}) \cdot \log n)$-implicit.*

If one's goal is merely to obtain an $O(1)$-cost universal SMP protocol for a family $\mathcal{F}$, the next observation shows that it suffices to find an $O(1)$-cost, public-coin, 2-way protocol for each member of $\mathcal{F}$. Therefore the family of all graphs with an $O(1)$-cost 2-way protocol is an implicit graph family with a polynomial-size induced-universal graph.

▶ **Corollary 1.2** (Restated)**.** *Let $\mathcal{F} = (\mathcal{F}_i)$ be a family of graphs with size $n(i)$ and suppose that for every graph $G \in \mathcal{F}_i$ there is an ϵ-error 2-way randomized communication protocol with cost at most $c(i)$. Then $R^{\mathsf{univ}}_{\epsilon}(\mathcal{F}) \leq 2^{c(i)}$. Furthermore, for any fixed constant c, the family $\mathcal{F}$ of graphs with $R^{\leftrightarrow}(\textsc{Adj}(G)) \leq c$ is $O(\log n)$-implicit.*

Proof. Every 2-way, deterministic cost c protocol can be represented as a binary tree with at most 2^c nodes, where each node is owned by either Alice or Bob and the message sent at each step is a 0 or 1 informing the other player of which branch to take in the tree. A randomized 2-way protocol is a distribution over such trees. To obtain a universal SMP protocol for the family $\mathcal{F}$, Alice and Bob do the following. On input $G \in \mathcal{F}$ and $x, y \in V(G)$, Alice and Bob use shared randomness to draw the deterministic cost c protocol for G from the distribution defined by the randomized 2-way protocol. Alice sends the size 2^c protocol tree and for each node she owns she identifies the branch to be taken. Bob does the same. The referee may then simulate the protocol. The conclusion follows from Theorem 1.1. ◀

3 Distance Labeling of Distributive Lattices

Distributive lattices and distances on these lattices will be defined in the next subsection, where we also give a necessary lemma characterizing the distances in terms of the *meet* and *join*. We will then present an $O(k \log k)$ weakly-universal protocol and an $O(k^2)$ universal

communication protocol for the family $\mathcal{D}^k$, where $\mathcal{D}$ are the distributive lattices. This implies a $O(k^2 \log n)$-implicit encoding $\mathcal{D}^k$ of the family $\mathcal{D}$ of distributive lattices. The $O(k \log k)$ weakly-universal protocol is optimal for sufficiently small values of k, since it applies to the k-Hamming Distance problem as a special case, for which Sağlam [31] recently gave a matching lower bound (even for 2-way communication). We obtain this result by adapting the optimal $O(k \log k)$ communication protocol for k-Hamming Distance originally presented by Huang et al. [21].

We also consider *modular* lattices, a generalization of distributive lattices, and show that deciding $\mathsf{dist}(x, y) \leq 2$ requires a protocol with cost $\Omega(n^{1/4})$.

3.1 Preliminaries on Distributive Lattices

A lattice is a type of partial order. We briefly review distributive lattices (see e.g. [9] for a good introduction) and then give a characterization of distances in modular and distributive lattices. The undirected graphs we study are the *cover graphs* of partial orders. For x, y in a partial order P, we say that y *covers* x and write $x \prec y$ if $\forall z \in P$: if $x \leq z < y$ then $x = z$. The *cover graph* (which is the undirected version of the *transitive reduction*) is the graph $\mathrm{cov}(P)$ on vertex set P with an edge $\{x, y\}$ iff $x \prec y$ or $y \prec x$.

We will define a few types of lattices.

▶ **Definition 3.1.** *Let $(P, <)$ be a partial order. For a pair $x, y \in P$:*

- *If the set $\{z \in P : x, y \geq z\}$ has a unique maximum, we call that maximum the* join *of x, y and write it as $x \wedge y$;*
- *If the set $\{z \in P : x, y \leq z\}$ has a unique minimum, we call that minimum the* meet *of x, y and write it as $x \vee y$.*

If $\forall x, y \in P$ the elements $x \wedge y, x \vee y$ exist, then P is a lattice. *A lattice L is* ranked *if there exists a rank function such that $x \prec y \implies \mathrm{rank}(x) + 1 = \mathrm{rank}(y)$ and the minimum element 0_L satisfies $\mathrm{rank}(0_L) = 0$. A finite lattice L is* upper-semimodular *if for every $x, y \in L$, $x \wedge y \prec x, y \implies x, y \prec x \vee y$. L is* lower-semimodular *if for every $x, y \in L$, $x, y \prec x \vee y \implies x \wedge y \prec x, y$. L is* modular *if it is both upper- and lower-semimodular. A lattice L is* distributive *if for all $x, y, z \in L$, $x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$. Every distributive lattice is modular and every modular lattice is ranked [9].*

A point x in a lattice L is join-irreducible *if there is no set $S \subseteq L$ such that $x = \bigvee S$ and* meet-irreducible *if there is no set $S \subseteq L$ such that $x = \bigwedge S$. Write $J(L)$ for the set of join-irreducible elements.*

A subset D of a partial order P is a downset *or* ideal *if: for all $x, y \in L$, if $x \in D$ and $y \leq x$ then $y \in D$. We will write $D(P)$ for the set of ideals of P.*

▶ **Theorem 3.2** (Birkhoff (see e.g. [9])). *Every distributive lattice L is isomorphic to the lattice of downsets of the partial order on its join-irreducible elements, ordered by inclusion; i.e. $L \simeq D(J(L))$, with the meet and join operations given by set union and intersection respectively.*

We need to prove some facts about distances in modular lattices.

▶ **Proposition 3.3.** *Let L be a graded lattice and let $x, y \in L$. Then $\mathsf{dist}(x, y) \geq |\mathrm{rank}(x) - \mathrm{rank}(y)|$, with equality if $x < y$ or $y < x$.*

Proof. This follows from the fact that for every edge $u \prec v$ in the path from x to y has $\mathrm{rank}(u) + 1 = \mathrm{rank}(v)$. ◀

To prove our characterization of distance, we define *inversions* in the path.

▶ **Definition 3.4.** *Let L be a lattice and let $c_1, \dots, c_m$ be a path in $\mathrm{cov}(L)$, so that $c_i \prec c_{i+1}$ or $c_{i+1} \prec c_i$ for each i. If $c_{i-1}, c_{i+1} \prec c_i$ or $c_i \prec c_{i-1}, c_{i+1}$ we call c_i an* inversion *on the path.*

▶ **Lemma 3.5.** *The following holds for any x, y in a lattice $\mathcal{M}$:*
1. *If $\mathcal{M}$ is lower-semimodular then $\mathsf{dist}(x, y) = \mathsf{dist}(x, x \wedge y) + \mathsf{dist}(y, x \wedge y)$;*
2. *If $\mathcal{M}$ is upper-semimodular then $\mathsf{dist}(x, y) = \mathsf{dist}(x, x \vee y) + \mathsf{dist}(y, x \vee y)$;*
3. *If $\mathcal{M}$ is distributive then $\mathsf{dist}(x, y) = |X \Delta Y|$ where $X, Y \in D(J(\mathcal{M}))$ are isomorphic images of x, y in Birkhoff's representation.*

Proof. It suffices to prove the first statement: the second follows by the analogous argument and the third follows from the modulartiy of distributive lattices and Birkhoff's representation.

Let $\mathcal{M}$ be lower-semimodular, let $x, y \in \mathcal{M}$, and let $x = c_0, c_1, \dots, c_m = y$ be a shortest path between x and y, so that $\mathsf{dist}(x, y) = \mathsf{dist}(x, c_i) + \mathsf{dist}(y, c_i)$ for any i. The statement holds trivially when $x < y$ or $y < x$ (since $x \wedge y = x$ or $x \wedge y = y$), so we assume x, y are incomparable. We prove the statement by induction on the largest rank of an inversion of the form $c_{i-1}, c_{i+1} \prec c_i$ in the path.

First suppose that c_i is any element of the path and assume for contradiciton that $\mathrm{rank}(c_i) < \mathrm{rank}(x \wedge y)$. Then

$$\mathsf{dist}(x, x \wedge y) = \mathrm{rank}(x) - \mathrm{rank}(x \wedge y) < \mathrm{rank}(x) - \mathrm{rank}(c_i) \leq \mathsf{dist}(x, c_i),$$

a contradiction. Thus $\mathrm{rank}(c_i) \geq \mathrm{rank}(x \wedge y)$ for each element of the path.

Suppose there are no inversions of the form $c_{i-1}, c_{i+1} \prec c_i$. Then $c_i < x, y$ and therefore $c_i \leq x \wedge y$ so $\mathrm{rank}(c_i) \leq \mathrm{rank}(x \wedge y)$, and by the above inequality we have $\mathrm{rank}(c_i) \geq \mathrm{rank}(x \wedge y)$, so $\mathrm{rank}(c_i) = \mathrm{rank}(x \wedge y)$. Therefore, as desired,

$$\begin{aligned}\mathsf{dist}(x, y) &= \mathsf{dist}(x, c_i) + \mathsf{dist}(y, c_i) = \mathrm{rank}(x) - \mathrm{rank}(c_i) + \mathrm{rank}(y) - \mathrm{rank}(c_i) \\ &= \mathrm{rank}(x) - \mathrm{rank}(x \wedge y) + \mathrm{rank}(y) - \mathrm{rank}(x \wedge y) \\ &= \mathsf{dist}(x, x \wedge y) + \mathsf{dist}(y, x \wedge y)\,.\end{aligned}$$

Now let c_i be an inversion of the form $c_{i-1}, c_{i+1} \prec c_i$ with $\mathrm{rank}(c_i) > \mathrm{rank}(x \wedge y)$. Then by lower-semimodulariity there is an element $c_i' = c_{i-1} \wedge c_{i+1} \prec c_{i-1}, c_{i+1}$. Then replacing c_i with c_i' maintains the length of the path. Performing the same operation on all such inversions of maximum rank reduces the maximum rank by 1 and the result holds by induction. ◀

3.2 A Universal Protocol for Distributive Lattices

Write $\mathcal{D} = (\mathcal{D}_n)$ for the family of cover graphs of distributive lattices on n vertices. We first give an optimal protocol for distances in distributive lattices in the *weak* universal model (recall that in this model, the referee sees the shared randomness). This protocol is adapted from a simplified presentation of Huang et al.'s k-Hamming Distance protocol ([21]) communicated to us by E. Blais.

▶ **Theorem 3.6.** *For any $\epsilon > 0$ and integer k, $R_\epsilon^{\mathsf{weak}}(\mathcal{D}^k) = O\left(k \log(k/\epsilon)\right)$.*

Proof. For any distributive lattice $L \simeq D(J(L))$, identify each vertex $x \in L$ with its ideal $X \subseteq J(L)$ of join-irreducibles. Write $e_1, \dots, e_m$ for the basis vectors of $\mathbb{F}_2^m$. Consider the following protocol. On the distributive lattice L and vertices x, y, Alice and Bob perform the following:

1. Define $m = \lceil \frac{(k+2)^2}{\epsilon} \rceil, q = \lceil \log \frac{1}{\epsilon} + \log \sum_{i=0} \binom{m}{i} \rceil$.
2. Let $S = (s_1, \ldots, s_m)$ be a multiset of uniformly random elements of $\mathbb{F}_2^q$.
3. For each join-irreducible element $j \in J(L)$ assign a uniformly random index $i_j \sim [m]$.
4. For each vertex $v \subseteq J(L)$ there is an indicator vector $a(v) \in \mathbb{F}_2^m$ defined by $a(v) = \sum_{j \in v} e_{i_j}$. Label v with $\ell(v) = \sum_{i=1}^m a(v)_i s_i$.
5. Alice sends $\ell(x)$ and Bob sends $\ell(y)$ to the referee.
6. The referee accepts iff $\ell(x) + \ell(y)$ is a sum of at most k elements of S.

By Lemma 3.5 and Birkhoff's theorem, $\mathsf{dist}(x,y) = \mathsf{dist}(x, x\wedge y) + \mathsf{dist}(x\wedge y, y) = |X \setminus Y| + |Y \setminus X| = |X \Delta Y|$, where Δ denotes the symmetric difference. Suppose $\mathsf{dist}(x,y) = |X\Delta Y| \le k$. Then $\ell(x) + \ell(y) = \sum_{j \in X\Delta Y} c(j)$ is a sum of at most k elements of S, so the protocol accepts with probability 1 (so this protocol has 1-sided error).

Now suppose $\mathsf{dist}(x,y) = |X\Delta Y| \ge k+1$. The correctness of the protocol follows from the next two claims along with the observations that $a(x)+a(y) = a(x\wedge y)$ and $\ell(x)+\ell(y) = \ell(x\wedge y)$ (with arithmetic in $\mathbb{F}_2$) and that $\mathsf{dist}(x,y) \ge k+1$ implies $\mathrm{rank}(x \wedge y) \ge k+1$. We will write $|a(v)|$ for the number of 1's in the vector $a(v)$.

▷ Claim 3.7. Any vertex $v \subseteq J(L)$ with $\mathrm{rank}(v) \ge k+1$ has $|a(v)| \ge k+1$ with probability at least $1 - \epsilon/2$.

Proof of claim. If $\mathrm{rank}(v) = k+1$, so v is a set of $k+1$ join-irreducibles, then the probability that any two indices $i_j, i_{j'}$ collide, for $j, j' \in v$, is by the union bound at most

$$\binom{k+1}{2} \mathbb{P}[i_j = i_{j'}] = \frac{k(k+1)}{2}\frac{1}{m} \le \frac{(k+1)^2}{2}\frac{\epsilon}{(k+2)^2} = \epsilon/2\,.$$

For $\mathrm{rank}(v) > k+1$ choose $v' \prec v$ so $k+1 \le \mathrm{rank}(v') < \mathrm{rank}(v)$, so using induction and the assumption $\epsilon < 1/2$,

$$\begin{aligned}\mathbb{P}[|a(v)| \le k] &= \frac{k+1}{m}\mathbb{P}[|a(v')| = k+1] + \frac{k}{m}\mathbb{P}[|a(v')| \le k] < \frac{\epsilon}{k+2} + \frac{\epsilon}{k+2}\cdot\frac{\epsilon}{2} \\ &= \epsilon\left(\frac{1}{k+2} + \frac{\epsilon}{2(k+2)}\right) \le \epsilon\left(\frac{1}{3} + \frac{1}{12}\right) < \epsilon/2\,.\end{aligned}$$

◁

▷ Claim 3.8. For any vertex $v \subseteq J(L)$, if the indicator vector $a(v)$ has weight $\ge k+1$ then, with probability at least $1 - \epsilon/2$, $\ell(v)$ is not a sum of at most k vectors in S.

Proof of claim. Write kS for the set of all sums of at most k vectors of S. Fix any $a(v)$ with weight $\ge k+1$ and let $A = \{i : a(v)_i = 1\}$ so $|A| \ge k+1$. Let $b \in kS$ be any sum of k vectors in S, and let $B \subset [m]$ be a set of indices of size $|B| \le k$ such that $b = \sum_{i\in B} s_i$.

Since $|B| \le k < |A|$ we must always have $A \setminus B \ne \emptyset$ and $\ell(v) + b = \sum_{i \in A\setminus B} s_i$, so $\mathbb{P}[\ell(v) + b = 0] = 2^{-q}$. Therefore, by the union bound over all such vectors b,

$$\mathbb{P}[\ell(v) \in kS] \le \sum_{i=0}^{k} \binom{m}{i} 2^{-q} < \epsilon/2\,.$$

◁

We can put a bound on q by using

$$\sum_{i=0}^{k}\binom{m}{i} \le k\binom{m}{k} \le k\left(\frac{em}{k}\right)^k$$

so

$$q \le 1 + \log\frac{1}{\epsilon} + \log k + k\log\frac{em}{k} \le \log\frac{2k}{\epsilon} + k\log\lceil\frac{ek}{\epsilon}\rceil = O\left(k\log\frac{k}{\epsilon}\right)\,.$$

◀

Observe that the referee must see the set S for the above protocol to work. We can easily modify the above protocol to get $O(k^2)$.

▶ **Theorem 3.9.** *For any $\epsilon > 0$ and any integer k, $R_\epsilon^{\mathsf{univ}}(\mathcal{D}^k) = O\left(k^2 \log(1/\epsilon)\right)$.*

Proof. The protocol is the same as above, with the following modification: Alice and Bob each send the indicator vectors $a(x), a(y) \in \mathbb{F}_2^m$.

The correctness of this protocol for error $1/3$ follows from Claim 3.7. Observe that Alice and Bob use the same strategy to send their messages and that the decision function is symmetric. The communication cost is now at most $m = \lceil 3(k+2)^2/2 \rceil$.

This protocol is one-sided, so to achieve error ϵ we can run the protocol $r = \lceil \log_3(1/\epsilon) \rceil$ times and take the AND of the results. The probability of failure is $(1/3)^r = 3^{-r} < \epsilon$. ◀

Now we apply Theorem 1.1 to obtain Theorem 1.3.

Since the family of distributive lattices is an upwards family (simply append a new least element to obtain a larger distributive lattice), we see from Proposition 2.17 that lattices in $\mathcal{D}^k$ can be randomly embedded into a constant-size graph, for any constant k. In fact, by inspection of the protocol, we see that the family $\mathcal{D}$ can be randomly embedded into a small-dimensional hypercube, while $\mathcal{D}^k$ can be embedded into the k-closure of the $O(k^2)$-dimensional hypercube.

▶ **Corollary 3.10.** *For any $\epsilon > 0$ and any k, there exists a graph U of size $2^{O(k^2 \log(1/\epsilon))}$ such that for all $L \in \mathcal{D}^k$, $L \sqsubset_\epsilon U$.*

3.3 Lower Bound for Modular Lattices

Since Lemma 3.5 works for any modular lattices, it is natural to ask whether we can achieve a similar constant-cost protocol for computing distance thresholds in modular lattices. However, we show that this is impossible.

▶ **Lemma 3.11.** *There is a function $m(n) = O(n^4)$ such that if G is any graph with n vertices (where $G(u,u) = 1$ for all u), there exists a modular lattice M with size $m(n)$ such that G is an induced subgraph of $\mathrm{cov}(M)^2$.*

Proof. Construct the lattice M as follows:

1. Start with vertices V, which are all incomparable.
2. For each edge $e = \{u, v\} \in E$, add vertices a_e, b_e such that $a_e < u, v < b_e$.
3. $\forall e = \{u, v\}, e' = \{u', v'\} \in E$ such that $e \cap e' = \emptyset$ add a vertex $c_{e,e'}$ with $a_e, a_{e'} < c_{e,e'} < b_e, b_{e'}$.
4. Add vertices 0_M and 1_M such that $0_M < a_e$ and $b_e < 1_M$ for all $e \in E$.

First we prove that M is a modular lattice and then we prove the bound on the size.

▷ Claim 3.12. M is a modular lattice.

Proof of claim. Observe that all orderings $<$ directly imposed by this process are covering orders $\prec$. Let $A = \{a_e\}_{e \in E}, B = \{b_e\}_{e \in E}, C = \{c_e\}_{e \in E}$ and V the original set of vertices. By construction, M is graded with $\mathrm{rank}(0_M) = 0, \mathrm{rank}(A) = 1, \mathrm{rank}(V) = \mathrm{rank}(C) = 2, \mathrm{rank}(B) = 3, \mathrm{rank}(1_M) = 4$. Note that for every pair of vertices $x, y \in M, 0_M \leq x, y \leq 1_M$ so upper- and lower-bounds exist.

Assume for contradiction that M is not a modular lattice, so there exist incomparable $x, y \in M$ such that either $x \wedge y$ or $x \vee y$ does not exist, or such that $x \wedge y \prec x, y \not\prec x \vee y$ or $x \wedge y \not\prec x, y \prec x \vee y$.

Case 1: Suppose $\text{rank}(x) \neq \text{rank}(y)$. Then $x \wedge y = 0_M$ and $x \vee y = 1_M$ so $x \wedge y \not\prec x, y \not\prec x \vee y$.
Case 2: Suppose $x, y \in A$ so $x = a_e, y = a_{e'}$. Then $0_M = a_e \wedge a_{e'} \prec a_e, a_{e'}$. If $a_e, a_{e'} < u, v$ for $u, v \in V$ then $u, v \in e \cap e'$ so $u = v$. If $a_e, a_{e'} < v, c_{d,d'}$ for $v \in V$ and $c_{d,d} \in C$ then $v \in e \cap e'$ and $c_{d,d'} = c_{e,e'}$ so $e \cap e' = \emptyset$, a contradiction. Finally, if $a_e, a_{e'} < c_{d,d'}, c_{d',d''}$ then $c_{d,d'} = c_{d',d''} = c_{e,e'}$. So $a_e \vee a_{e'}$ exists and $a_e \wedge a_{e'} \prec a_e, a_{e'} \prec a_e \vee a_{e'}$. The same argument holds for $x, y \in B$.
Case 3: Suppose $x, y \in V$ and assume $a_e, a_{e'} < x, y$. Then $x, y \in e \cap e'$ so $a_e = a_{e'}$. A similar argument holds for $x, y < b_e, b_{e'}$. So $x \wedge y \prec x, y \prec x \vee y$.
Case 4: Suppose $x, y \in C$ so $x = c_{e,e'}, y = c_{d,d'}$. Suppose $a_s, a_t < c_{e,e'}, c_{d,d'}$. Then $s, t \in \{e, e'\} \cap \{d, d'\}$ so either $\{e, e'\} = \{d, d'\}$ or $s = t$. The same argument holds for $c_{e,e'}, c_{d,d'} < b_s, b_t$ so $x \wedge y \prec x, y \prec x \vee y$.
Case 5: Suppose $x \in V, y \in C$ so $y = c_{e,e'}$ which implies $e \cap e' = \emptyset$. If $x \notin e \cup e'$ then $x \wedge c_{e,e'} = 0_M$ and $x \vee c_{e,e'} = 1_M$ so $x \wedge c_{e,e'} \not\prec x, c_{e,e'} \not\prec x \vee c_{e,e'}$; so suppose $x \in e \cup e'$. If $a_e, a_{e'} < x, c_{e,e'}$ then $x \in e \cap e'$ which is a contradiction. Then $x \in e$ or $x \in e'$; say $x \in e$. Then $a_e = x \wedge c_{e,e'}$. The same argument holds for B so $a_e = x \wedge c_{e,e'} \prec x, c_{e,e'} \prec x \vee c_{e,e'} = b_e$. ◁

▷ Claim 3.13. G is an induced subgraph of $\text{cov}(M)^2$.

Proof of claim. Suppose $\{u, v\} \in E$. Then there is $a_e \prec u, v$ so $\mathsf{dist}(u, v) \leq 2$ in $\text{cov}(M)$. Now let $u, v \in V(G)$ and suppose $\mathsf{dist}(u, v) \leq 2$ in $\text{cov}(M)$ so that, by Lemma 3.5, $u \wedge v \prec u, v \prec u \vee w$. By construction, either $u = v$ so $G(u, v) = G(u, u) = 1$, or $u \wedge v = a_e$ for some $e \in E(G)$ so $u, v \in e$ and therefore $G(u, v) = 1$. ◁

The size of M is at most $2 + |E(G)| + |E(G)|^2 = O(n^4)$. Let $m(n)$ be the maximum size of a modular lattice obtained in this way from a graph of size n. We want all constructions to be of the same size, so repeatedly append new least elements until the size reaches $m(n)$; this maintains the modular lattice property. ◀

▶ **Theorem 3.14.** *Let $\mathcal{M} = (\mathcal{M}_n)$ be the family of cover graphs of modular lattices. $R^{\text{univ}}(\mathcal{M}^2) \geq \Omega(n^{1/4})$.*

Proof. Suppose there is a protocol for $\mathcal{M}^2$ with cost $o(n^{1/4})$. Given a graph G of size n, Alice and Bob construct the modular lattice of size $m(n) = O(n^4)$ with G an induced subgraph of $\text{cov}(M)^2$ and run the protocol for $\mathcal{M}^2$ with size $m(n)$ (observe that all possible constructions must be of the same size, since the referee does not know which lattice Alice and Bob construct). This has cost $o(m(n)^{1/4}) = o(n)$, which contradicts Theorem 1.6. ◀

4 Communication on Efficiently Labelable Graphs

In this section we take inspiration from the field of implicit graphs and graph labeling and show that one may often, but not always, obtain constant-cost adjacency and k-distance protocols for families that are commonly studied in the graph labeling literature.

4.1 Trees, Forests, and Interval Graphs

In this section we pick the low-hanging fruit from trees and forests (and interval graphs). Applying Theorem 1.1 with the next lemma, we get Theorem 1.4.

▶ **Lemma 4.1.** *Let $\mathcal{T} = (\mathcal{T}_n)$ be the family of trees of size n. $R^{\text{univ}}_{\epsilon}(\mathcal{T}^k) = O\left(k \log \frac{1}{\epsilon}\right)$, and this protocol will correctly compute the distance in the case $\mathsf{dist}(x, y) \leq k$.*

Proof. Consider the following protocol. On input $(T, x), (T, y)$ for a tree T, Alice and Bob perform the following.

1. Partition the vertices of T into sets $T_1, \ldots, T_m$ such that $T_i = \{v \in V(T) : (i-1)k \leq \text{depth}(v) < ik\}$. For each $v \in V(T)$ let $t(v)$ be the index of the unique set satisfying $v \in T_{t(v)}$.
2. For each vertex $v \in V(T)$ assign a uniformly random color $\ell(v)$ in $[m]$ for $m = \lceil 6/\epsilon \rceil$. Let x' be root of the subtree of $T_{t(x)}$ that contains x, and let x'' be the root of the subtree of $T_{t(x)-1}$ that contains x. Let $x_0, x_1, \ldots, x_k, \ldots, x_{k_1} = x$ be the path from x'' to x (with $x_k = x'$) and let $y_0, \ldots, y_k, \ldots, y_{k_2}$ be the path from y'' to y. Alice and Bob send $\ell(x_0), \ldots, \ell(x_{k_1})$ and $\ell(y_0), \ldots, \ell(y_{k_2})$ respectively.
3. If $\ell(x') = \ell(y')$, let p be the maximum index such that $\ell(x_i) = \ell(y_i)$ for each $k < i \leq p$. Let $d = (k_1 - p) + (k_2 - p)$. If $\ell(x'') = \ell(y'')$, let p be the maximum index such that $\ell(x_i) = \ell(y_i)$ for each $i \leq p$ and let $d = (k_1 - p) + (k_2 - p)$. If $\ell(x'') = \ell(y')$ let p be the maximum index such that $\ell(x_i) = \ell(y_{k+i})$ for each $i \leq p$ and let $d = (k_1 - p) + (k_2 - k - p)$. If $\ell(x') = \ell(y'')$ do the same with x, y reversed. In each case, if $d \leq k$, the referee outputs d, otherwise they output "$> k$". If none of the above cases hold, output "$> k$".

The cost of this protocol is $2k\lceil \log m \rceil = O(k \log(1/\epsilon))$. With probability at least $1 - 4/m > 1 - \epsilon/2$, each of the possible equalities $x'' = y'', x' = y', x'' = y', x' = y''$ will be correctly observed by the referee. If $\{x', x''\} \cap \{y', y''\} = \emptyset$ then x, y are not in the same subtree rooted at depth $\text{depth}(x'')$, so the distance from x to any common ancestor of x, y is at least $\mathsf{dist}(x, x'') > k$. Therefore if $\mathsf{dist}(x, y) \leq k$, one of these equalities will hold. If $x'' = y''$ and q is the maximum integer such that $x_i = y_i$ for all $i \leq q$ then $\mathsf{dist}(x, y) = (k_1 - q) + (k_2 - q)$, because the deepest common ancestor of x, y is at depth $\text{depth}(x_0) + q$. Conditional on the 4 equalities being correctly observed, we will have $d = (k_1 - p) + (k_2 - p) \leq k$ since $p \geq q$. If $p > q$ then $\ell(x_{q+1}) = \ell(y_{q+1})$ even though $x_{q+1} \neq y_{q+1}$, which occurs with probability $1/m < \epsilon/2$. Therefore the probability that $d \neq \mathsf{dist}(x, y)$ is at most $2(\epsilon/2) = \epsilon$ when $\mathsf{dist}(x, y) \leq k$. A similar argument holds in the other 3 cases.

If $\mathsf{dist}(x, y) > k$ then still with probability at least $1 - \epsilon/2$ all 4 possible equalities are correctly observed. Following the same argument as in the equality case, we see that if any of the equalities hold we will have $d = \mathsf{dist}(x, y)$ with probability greater than $1 - \epsilon/2$, for total error probability less that ϵ. If none of the 4 equalities hold then the probability of error is at most $\epsilon/2$. ◀

Since trees have efficient protocols, one might wonder about generalizations of trees. The *arboricity* of a graph is one such generalization, which measures the minimum number of forests required to partition all the edges.

▶ **Definition 4.2.** *A graph $G = (V, E)$ has* arboricity α *iff there exists an edge partition of G into forests $T_1, \ldots, T_\alpha$. Equivalently, for S ranging over the set of subgraphs of G, G has*

$$\max_S \left\lceil \frac{E(S)}{V(S) - 1} \right\rceil \leq \alpha \,.$$

Low-arboricity graphs easily admit an efficient universal SMP protocol for adjacency.

▶ **Proposition 4.3.** *Let $\mathcal{F}$ be any family of graphs with arboricity at most α. For all $\epsilon > 0, R_\epsilon^{\mathsf{univ}}(\mathcal{F}) = O\left(\alpha \log \frac{\alpha}{\epsilon}\right)$.*

Proof. On the graph G and vertices x, y, Alice and Bob perform the following:

1. Compute a partition of G into α forests $T_1, \ldots, T_\alpha$.
2. Assign to each vertex v a uniformly random number $\ell(v) \sim [m]$ for $m = \lceil 2\alpha/\epsilon \rceil$.

3. Let x_i be the parent of x in tree i and let y_i be the parent of y. Alice sends $\ell(x)$ and $\ell(x_i)$ for each i, and Bob does this same with y.
4. The referee accepts iff $\ell(x) = \ell(y_i)$ or $\ell(y) = \ell(x_i)$ for any i.

This protocol has one-sided error since if x, y are adjacent then either $x_i = y$ or $y_i = x$ for some i, so the referee will accept with probability 1. If x, y are not adjacent then the referee will accept with probability at most $2\alpha \cdot \frac{1}{m} < \epsilon$. ◀

However, even graphs of arboricity 2 do not admit efficient protocols or labeling schemes for distance 2, which we can show by embedding an arbitrary graph of size $\Omega(\sqrt{n})$ into the 2-closure of an arboricity 2 graph of size n:

▶ **Proposition 4.4.** *Let $\mathcal{F}$ be the family of arboricity-2 graphs. Then $R^{\mathsf{univ}}(\mathcal{F}^2) \geq \Omega(\sqrt{n})$.*

Proof. The lower bound is obtained via Theorem 1.6 in the same way as in Theorem 3.14, using the following construction. For all simple graphs $G = (V, E)$ with n vertices, there exists a graph A of size $n + \binom{n}{2}$ and arboricity 2 such that G is an induced subgraph of A^2. Let A be the graph defined as follows:

1. Add each vertex $v \in V$ to A;
2. For each pair of vertices $\{u, v\}$ add a vertex $e_{\{u,v\}}$ and add edges $\{u, e_{\{u,v\}}\}, \{v, e_{\{u,v\}}\}$ iff $\{u, v\} \in E$.

This graph has arboricity 2 since for each $e_{\{u,v\}}$ we may assign each of its 2 incident edges a color in $\{1, 2\}$ (if the edges exist). Then the edges with color $i \in \{1, 2\}$ form a forest with roots in V. ◀

Now we give an example of a family, the interval graphs, with size $O(\log n)$ adjacency labels but with no constant-cost universal SMP protocol; in fact, randomization does not give more than a constant-factor improvement for this family. An *interval graph* of size n is a graph G where for each vertex x there is an interval $X \subset [2n]$ such that any two vertices x, y are adjacent in G iff $X \cap Y \neq \emptyset$. These have an $O(\log n)$ adjacency labeling scheme [23] (one can simply label a vertex with its two endpoints in $[2n]$).

There is a simple reduction from the Greater-Than communication problem, in which Alice and Bob receive integers $x, y \in [n]$ and must decide if $x < y$. It is known that the one-way public-coin communication cost of Greater-Than is $\Omega(\log n)$ [25], so $R^{\parallel}(\text{Greater-Than}) = \Omega(\log n)$.

▶ **Proposition 4.5.** *For the family $\mathcal{F}$ of interval graphs, $R^{\mathsf{univ}}(\mathcal{F}) = \Omega(\log n)$.*

Proof. We can use a universal SMP protocol for $\mathcal{F}$ to get a protocol for Greater-Than as follows. Alice and Bob construct the interval graph with intervals $[1, i], [i, n]$ for each $i \in [n]$, so there are $2n$ vertices in G. On input $x, y \in [n]$, Alice and Bob compute adjacency on the intervals $[1, x], [1, y]$ and then again on $[1, x], [y, n]$. Assume both runs of the protcol succeed. Then when the output is 1 for both runs we must have $y \in [1, x]$ so $y \leq x$ and otherwise we have $y \notin [1, x]$ so $x < y$. ◀

4.2 Planar Graphs

Write $\mathcal{P}_n$ for the set of planar graphs of size n and write $\mathcal{P} = (\mathcal{P}_n)$ for the family of planar graphs. Gavoille et al. [15] gave an $O(\sqrt{n} \log n)$ labeling scheme where $\mathsf{dist}(x, y)$ can be computed from the labels of x, y, and Gawrychowski and Uznański [16] improved this to $O(\sqrt{n})$. These labeling schemes recursively identify size-$O(\sqrt{n})$ sets S and record the distance of each vertex v to each $u \in S$, so the $\sqrt{n}$ factor is unavoidable using this technique. We want to solve k-distance with a cost independent of n, so we need a new method. Our main tool is Schnyder's elegant decomposition of planar graphs into trees:

▶ **Theorem 4.6** (Schnyder [32], see [11]). *Define the dimension* $\dim(G)$ *of a graph* G *as is the minimum* d *such that there exist total orders* $<_1, \ldots, <_d$ *on* $V(G)$ *satisfying:*

(*) For every edge $\{u, v\} \in E$ *and* $w \notin \{u, v\}$ *there exists* $<_i$ *such that* $u, v <_i w$.

G *is planar iff* $\dim(G) \leq 3$. *If* G *is planar then there exists a partition* T_1, T_2, T_3 *of the edges into directed trees satisfying the following. Let* T_i^{-1} *be edge-induced directed graph on* $V(G)$ *obtained by reversing the direction of each edge in* T_i. *The graphs with edges* $T_i \cup T_{i-1}^{-1} \cup T_{i+1}^{-1}$ *have linear extensions* $<_i$ *such that* $<_1, <_2, <_3$ *satisfy (*).*

Schnyder's Theorem implies that the arboricity of planar graphs is at most 3, so we may use the protocol for low-arboricity graphs (Proposition 4.3) to determine adjacency in $\mathcal{P}$, so we move on to $\mathcal{P}^2$, which may have large arboricity (arboricity is within a constant factor of *degeneracy*):

▶ **Theorem 4.7** ([1]). *There are planar graphs* P *for which the degeneracy of* P^2 *is* $\Theta(\deg P)$, *where* $\deg P$ *is the maximum degree of any vertex in* P.

We avoid this blowup in arboricity by treating edges of the form $a \leftarrow b \rightarrow c$ separately (with directions taken from the Schnyder wood). The proof uses the following split operation:

▶ **Definition 4.8.** *Let* $G \in \mathcal{P}$ *and fix a planar map and a Schnyder wood* T_1, T_2, T_3. *Define the graph* $\mathsf{split}(G)$ *by the following procedure (see Figure 1):*

1. *For each vertex* $s \in V(G)$ *add vertices* s, s_1, s_2, s_3 *to* $\mathsf{split}(G)$ *(excluding* s_i *if* s *has no incoming edge in* T_i*). Add edges* (s_i, s) *to* T_i';
2. *For each (directed) edge* $(u, v) \in T_i$ *add the edges* $(u_{i-1}, v_i), (u_{i+1}, v_i)$ *(arithmetic mod 3) to* T_i';
3. *For the unique (directed) edge* $(v, u) \in T_i$ *add the edges* $(v_{i-1}, u), (v_{i+1}, u)$ *to* T_i'.

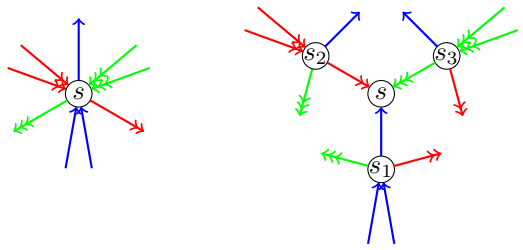


Figure 1 Splitting vertex s, with T_1, T_2, T_3 in blue, red, and green respectively (1,2, and 3 arrowheads).

▶ **Proposition 4.9.** $\mathsf{split}(G)$ *is planar.*

Proof. We prove that splitting any vertex s results in a planar graph. By induction we may then split each vertex in sequence and obtain a planar graph. Let $<_i$ be any total order on $V(G)$ extending $T_i \cup T_{i-1}^{-1} \cup T_{i+1}^{-1}$, which satisfies condition (*) by Schnyder's theorem. Let $<_1', <_2', <_3'$ be the same total orders, extending T_1', T_2', T_3', and augmented to include s_1, s_2, s_3 as follows:

1. For each $u \in V(G)$, $s_i <_j' u$ iff $s <_j u$ and $u <_j' s_i$ iff $u <_j s$;
2. For each i, set $s_i <_i' s <_i' s_{i+1} <_i' s_{i-1}$. This is possible since $\{s_i\}$ do not have a defined ordering in $<_i$ and remain incomparable after the previous step.

Note that for any edge $(u, v) \in T_i'$ we have $u <_i' v$ and $v <_j' u$ for $j \neq i$. It suffices to prove that condition (*) is satisfied by the new orders. Let $\{u, v\} \in E(\mathsf{split}(G))$ and let $w \notin \{u, v\}$. We will show that there exists i such that $u, v <_i' w$.

If $u, v, w \in V(G)$ then we are done since the orders $<'_i$ are the same as $<_i$ on these vertices.

If $u = s_i$ then either $v \in V(G) \setminus s$, in which case $v <_i s$ so $v <'_i u$ and therefore $u <'_j v$ for $j \neq i$, or $v = s$ so $u <'_i v$ and therefore $v <'_j u$ for $j \neq i$. Let $v \neq s$. For any $w \in V(G) \setminus \{v\}$ we have, by (*), either $v, s <_i w$ so $v <'_i u <'_i s <'_i w$, or $v, s <_j w$ so $u <'_j v <'_j w$. If $v = s$ then by construction there exists $(u', u) \in T_i$. By (*), either $u', v <_i w$ so $u <'_i v <'_i w$, or $u', v <_j w$ so $v <'_j u <'_j u' <'_j w$.

The only case remaining is if $w = s_i$ and $u, v \in V(G)$. By construction there exists $(w', w) \in T_i$. Either $u, v <_i w' <'_i w <'_i s$ or by (*) there exists j such that $u, v <_j s$ and since (w, s) is an edge in T'_i, $s <'_j w$ for $j \neq i$. ◀

▶ **Definition 4.10.** *Let $G = (V, E)$ be a planar graph. Fix a planar map and a Schnyder wood T_1, T_2, T_3. For each i, define the graph $G_i = (V, E \setminus T_i)$ as the graph obtained by removing each edge in T_i. Define the* head-to-head closure *of G_i, written $G_i^{\leftarrow\rightarrow}$, as the graph with an edge $\{u, v\}$ iff there exists $w \in V$ such that $u \leftarrow w \rightarrow v$ in G_i. (Observe that the two outgoing edges of w must be in T_{i-1}, T_{i+1}.) Let $G^{\leftarrow\rightarrow}$ be the subgraph of G^2 containing all edges occuring in $G_i^{\leftarrow\rightarrow}$ for each i.*

▶ **Lemma 4.11.** *Let G be a planar graph. For any graph M, if M is a minor of $G_i^{\leftarrow\rightarrow}$ then M is a minor of* $\mathsf{split}(G)$.

Proof. We will prove the following claim.

▷ Claim 4.12. For any set $P = \{P_j\}$ of simple paths $P_j \subseteq V(G_i^{\leftarrow\rightarrow})$, with endpoints $\{(s_j, t_j)\}$ such that no two paths P_j, P_k have the same endpoints and $P_j \cap P_k \subseteq \{s_j, s_k, t_j, t_k\}$, there exists a set of paths $Q = \{Q_j\}$ of paths in $\mathsf{split}(G)$ with the same endpoints such that

$$Q_j \cap Q_k \subseteq \{s_j, s_k, t_j, t_k\} \cup \{(s_j)_{i-1}, (s_k)_{i-1}, (t_j)_{i-1}, (t_k)_{i-1}\} \cup \{(s_j)_{i+1}, (s_k)_{i+1}, (t_j)_{i+1}, (t_k)_{i+1}\},$$

where the vertices s_i, s_{i+1}, s_{i-1} are defined as in the split operation.

Proof of claim. For each path P_j, perform the following. For each edge $\{u, w\}$ in the path P_j, there is some (not necessarily unique) vertex v such that either $(v, u) \in T_{i-1}$ and $(v, w) \in T_{i+1}$, or the same holds with u, w reversed. Add the edges $\{u, u_{i-1}\}, \{u_{i-1}, v_i\}, \{v_i, w_{i+1}\}, \{w_{i+1}, w\}$ to Q_j. If P_j is a singleton $P_j = \{u\}$ so $s_j = t_j$ then add u to Q_j.

Consider two paths Q_j, Q_k constructed this way. $G_i^{\leftarrow\rightarrow}$ has vertex set V and $\mathsf{split}(G)$ has vertex set $V' \supset V$. By construction, $P_j \subseteq Q_j$ and $P_k \subseteq Q_k$ and $(Q_j \cap V) = P_j$. Suppose there exists $z \in Q_j \cap Q_k$ that is not an endpoint, so $z \notin \{s_j, s_k, t_j, t_k\}$. If $z \in V$ then $z \in P_j \cap P_k \subseteq \{s_j, s_k, t_j, t_k\}$, so we only need to worry about $z \in V' \setminus V$.

If $z = v_i$ for some vertex v then there are unique distinct vertices $u_{i-1}, w_{i+1} \in V'$ adjacent to v_i such that $u_{i-1}, w_{i+1} \in Q_j \cap Q_k$. Then $u, w \in Q_j \cap Q_k$ also, so $u, w \in P_j \cap P_k$; but then $u \neq w$ are the start and end points of P_j, P_k, so $P_j = P_k$, a contradiction.

If $z = v_{i-1}$ for some vertex $v \in V$ then $v \in Q_j \cap Q_k$, so by the case above, $v \in \{s_j, s_k, t_j, t_k\}$ and $z \in \{(s_j)_{i-1}, (s_k)_{i-1}, (t_j)_{i-1}, (t_k)_{i-1}\}$. Likewise for $z = v_{i+1}$. ◁

Let M be a minor of $G_i^{\leftarrow\rightarrow}$, so a subdivision of M occurs as a subgraph of $G_i^{\leftarrow\rightarrow}$. Therefore there is a set of paths P in $G_i^{\leftarrow\rightarrow}$ satisfying the conditions of the claim, so that by contracting each path into a single edge, and deleting the rest of the graph, we obtain M. Let $Q = \{Q_j\}$ be the set of paths given by the claim. For endpoints $s_j, t_j \in Q_j$, contract the edges $\{s_j, (s_j)_{i\pm1}\}$ and $\{t_j, (t_j)_{i\pm1}\}$. The result is a contraction of $\mathsf{split}(G)$ and a set of paths Q' that is a subdivision of M, so M is a minor of $\mathsf{split}(G)$, which proves the lemma. ◀

▶ **Corollary 4.13.** *$G_i^{\leftarrow\rightarrow}$ is planar and $G^{\leftarrow\rightarrow}$ has arboricity at most 9.*

Proof. A graph is planar iff it does not contain K_5 or $K_{3,3}$ as a minor (Kuratowski's Theorem). If $G_i^{\leftarrow\rightarrow}$ is not planar then it contains K_5 or $K_{3,3}$ as a minor, so by the above lemma, $\mathsf{split}(G)$ contains K_5 or $K_{3,3}$ as a minor, so $\mathsf{split}(G)$ is not planar, a contradiction. Since planar graphs have arboricity at most 3, the edge union $G^{\leftarrow\rightarrow}$ of 3 planar graphs has arboricity at most 9. ◀

By separating the $\leftarrow\rightarrow$ edges from the remaining edges of $\mathcal{P}^2$, we obtain a constant-cost universal SMP protocol for $\mathcal{P}^2$, and then by applying Theorem 1.1 we obtain Theorem 1.5.

▶ **Lemma 4.14.** *For all $\epsilon > 0$, $R_\epsilon^{\mathsf{univ}}(\mathcal{P}^2) = O\left(\log \frac{1}{\epsilon}\right)$.*

Proof. For a planar graph $G = (V, E)$ with a fixed planar map and a Schnyder wood T_1, T_2, T_3, define the graph $G_i = (V, E \setminus T_i)$ as the graph obtained by removing the edges in tree T_i.

On planar graph $G \in \mathcal{P}_n$ and vertices x, y, Alice and Bob perform the following:

1. For each i define x_i, y_i to be the parents of x, y in T_i. Run the protocol for adjacency with error $\epsilon/7$ on (x, y_i) and (x_i, y) for each i.
2. Run the protocol for low-arboricity graphs on $G^{\leftarrow\rightarrow}$ with error $\epsilon/7$.
3. Accept iff one of the above sub-protocols accepts.

By Corollary 4.13, $G^{\leftarrow\rightarrow}$ has arboricity at most 9, we may apply the protocol for low-arboricity graphs in step 2. If $\mathsf{dist}(x, y) > 2$ then the protocol will correctly reject with probability at least $1 - \epsilon$ since there are 7 applications of $\epsilon/7$-error protocols. It remains to show that if $\mathsf{dist}(x, y) = 2$ then the algorithm will accept.

Suppose x, y are of distance 2. Then the paths between them are of the following forms (with edge directions taken from the Schnyder wood).

1. $x \to v \to y$ or $x \to v \leftarrow y$. This is covered by step 1.
2. $x \leftarrow v \to y$. This is covered by step 2. ◀

Since planar graphs are an upwards family (just insert a new vertex), we obtain a constant-size probabilistic universal graph for $\mathcal{P}^2$.

▶ **Corollary 4.15.** *For any $\epsilon > 0$, there is a graph U of size $O(\log(1/\epsilon))$ such that for every $G \in \mathcal{P}^2$, $G \sqsubset_\epsilon U$.*

5 Discussion and Open Problems

Error-tolerance. In the introduction we mentioned that the universal SMP model allows us to study error-tolerance in the SMP model. This could be done as follows: suppose the referee knows a reference graph G and the players are guaranteed to see a graph that is "close" to G by some metric. How much does this change the complexity of the problem, compared to computing G? One common distance metric in, say, the property testing literature, is to count the number of edges that one must add or delete. That is, for two graphs G, H on vertex set $[n]$, write $\mathsf{dist}(G, H) = \frac{1}{n^2} \sum_{i,j \in [n]} \mathbb{1}\left[G(i, j) \neq H(i, j)\right]$. The distance is usually thought of as a constant. We can easily give a strong negative result for this situation:

▶ **Proposition 5.1.** *Let $\mathcal{F}$ be any family of graphs and $\mathcal{F}_\delta$ the family of graphs G such that $\min_{F \in \mathcal{F}} \mathsf{dist}(G, F) \leq \delta$. Then*

$$R^{\mathsf{univ}}(\mathcal{F}_\delta) = \Omega(\sqrt{\delta} n)\,.$$

Proof. Let G be any graph on $\sqrt{\delta}n$ vertices and let $F \in \mathcal{F}$. Choose any set $S \subseteq V(F)$ with $|S| = |G|$. Construct F' by replacing the subgraph induced by S with the graph G. Then $\mathsf{dist}(F, F') \leq \frac{|G|^2}{n^2} = \delta$ so $F' \in \mathcal{F}_\delta$. Then the conclusion follows from Theorem 1.6. ◀

This suggests that this is not the correct way to model contextual uncertainty in the SMP model, but universal SMP gives a framework for studying many other error tolerance settings. For example, we could suppose that the referee knows a reference planar graph G, and the players are guaranteed to see a graph G' that is close to G and also planar; this would not increase the cost of the protocol due to our results on planar graphs.

Implicit graph conjecture. A major open problem in graph labeling is the *implicit graph conjecture* of Kannan, Naor, and Rudich [23], which asks if every *hereditary* graph family $\mathcal{F}$ (where for each $G \in \mathcal{F}$, every induced subgraph of G is also in $\mathcal{F}$) containing at most $2^{O(n \log n)}$ graphs of size n has an $O(\log n)$ adjacency labeling scheme. Not much progress has been made on this conjecture (see e.g. [33, 10]). We ask a weakened version of this conjecture:

▶ **Question 5.2.** *For every hereditary family $\mathcal{F} = (\mathcal{F}_n)$ such that $|\mathcal{F}_n| \leq 2^{O(n \log n)}$, is $R^{\mathsf{univ}}(\mathcal{F}) = O(\log n)$?*

Good candidates for disproving the implicit graph conjecture are geometric intersection graphs, like disk graphs (intersections of disks in $\mathbb{R}^2$) or k-dot product graphs (graphs whose vertices are vectors in $\mathbb{R}^k$, with an edge if the inner product is at least 1) [33]. These are good candidates because encoding the coordinates of the vertices as integers will fail [22]. Randomized communication techniques may be able to make progress.

Modular lattices. We have shown that there is no constant-cost universal protocol for distance 2 in modular lattices but, like low-arboricity graphs, adjacency (and therefore $O(\log n)$-implicit encodings) may still be possible.

Planar graphs. Our protocol for computing distance 2 on planar graphs did not generalize in a straightforward fashion to distance 3. Nevertheless, we expect that there is a method for computing k-distance on planar graphs with complexity dependent only on k; given that a Schnyder wood partitions each edge into 3 groups, we expect that $\widetilde{O}(3^k)$ should be possible, and maybe only $\mathrm{poly}(k)$, considering that there is a $O(\sqrt{n})$ distance-labeling scheme.

Sharing randomness with the referee. Finally, it seems to be unknown what the relationship is between SMP protocols where the referee shares the randomness, and protocols where the referee is deterministic, even though both models are used extensively in the literature. Our Proposition 1.8 relates these two models via universal SMP but does not yet give a general upper bound on the universal cost in terms of the weakly-universal cost.

▶ **Question 5.3.** *What general upper bounds can we get on universal SMP in terms of weakly-universal SMP?*

References

1 Geir Agnarsson and Magnús M Halldórsson. Coloring powers of planar graphs. *SIAM Journal on Discrete Mathematics*, 16(4):651–662, 2003.

2 Stephen Alstrup, Philip Bille, and Theis Rauhe. Labeling schemes for small distances in trees. *SIAM Journal on Discrete Mathematics*, 19(2):448–462, 2005.

3 Stephen Alstrup, Søren Dahlgaard, and Mathias Bæk Tejs Knudsen. Optimal induced universal graphs and adjacency labeling for trees. *Journal of the ACM (JACM)*, 64(4):27, 2017.

4 Stephen Alstrup, Søren Dahlgaard, Mathias Bæk Tejs Knudsen, and Ely Porat. Sublinear distance labeling. In *24th Annual European Symposium on Algorithms (ESA 2016)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.

5 Stephen Alstrup, Inge Li Gørtz, Esben Bistrup Halvorsen, and Ely Porat. Distance labeling schemes for trees. In *43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016)*, 2016.

6 Stephen Alstrup, Haim Kaplan, Mikkel Thorup, and Uri Zwick. Adjacency labeling schemes and induced-universal graphs. *SIAM Journal on Discrete Mathematics*, 33(1):116–137, 2019.

7 Tiziana Calamoneri. The $L(h, k)$-Labelling Problem: An Updated Survey and Annotated Bibliography. *The Computer Journal*, 54(8):1344–1371, 2011.

8 Clément L Canonne, Venkatesan Guruswami, Raghu Meka, and Madhu Sudan. Communication with imperfectly shared randomness. *IEEE Transactions on Information Theory*, 63(10):6799–6818, 2017.

9 Nathalie Caspard, Bruno Leclerc, and Bernard Monjardet. *Finite Ordered Sets: Concepts, Results and Uses*. Cambridge University Press, 2012.

10 Maurice Chandoo. On the implicit graph conjecture. In *41st International Symposium on Mathematical Foundations of Computer Science (MFCS 2016)*, volume 58, page 23. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2016.

11 Stefan Felsner. *Geometric graphs and arrangements: some chapters from combinatorial geometry*. Springer, 2012.

12 Pierre Fraigniaud and Amos Korman. On randomized representations of graphs using short labels. In *Proceedings of the 21st Annual Symposium on Parallelism in Algorithms and Architectures*, pages 131–137. ACM, 2009.

13 Pierre Fraigniaud and Amos Korman. An Optimal Ancestry Labeling Scheme with Applications to XML Trees and Universal Posets. *Journal of the ACM*, 63(1):6, 2016.

14 Cyril Gavoille and Arnaud Labourel. Shorter implicit representation for planar graphs and bounded treewidth graphs. In *European Symposium on Algorithms*, pages 582–593. Springer, 2007.

15 Cyril Gavoille, David Peleg, Stéphane Pérennes, and Ran Raz. Distance labeling in graphs. *Journal of algorithms*, 53(1):85–112, 2004.

16 Paweł Gawrychowski and Przemysław Uznański. A note on distance labeling in planar graphs, 2016. `arXiv:1611.06529`.

17 Badih Ghazi, Ilan Komargodski, Pravesh Kothari, and Madhu Sudan. Communication with Contextual Uncertainty. *Computational Complexity*, 27(3):463–509, 2018.

18 Badih Ghazi and Madhu Sudan. The power of shared randomness in uncertain communication. In *44th International Colloquium on Automata, Languages, and Programming (ICALP 2017)*, pages 49:1–49:14. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017.

19 Oded Goldreich, Brendan Juba, and Madhu Sudan. A theory of goal-oriented communication. *Journal of the ACM*, 59(2):8, 2012.

20 Elad Haramaty and Madhu Sudan. Deterministic compression with uncertain priors. *Algorithmica*, 76(3):630–653, 2016.

21 Wei Huang, Yaoyun Shi, Shengyu Zhang, and Yufan Zhu. The communication complexity of the Hamming distance problem. *Information Processing Letters*, 99(4):149–153, 2006.

22 Ross J Kang and Tobias Müller. Sphere and dot product representations of graphs. *Discrete & Computational Geometry*, 47(3):548–568, 2012.

23 Sempath Kannan, Moni Naor, and Steven Rudich. Implicit Representations of Graphs. *SIAM Journal on Discrete Mathematics*, 5(4):596–603, 1992.

24 Haim Kaplan and Tova Milo. Short and Simple Labels for Small Distances and Other Functions. In *Workshop on Algorithms and Data Structures*, pages 246–257. Springer, 2001.

25 Peter Bro Miltersen, Noam Nisan, Schmuel Safra, and Avi Wigderson. On data structures and asymmetric communicaton complexity. *Journal of Computer and System Sciences*, 57(1):37–49, 1998.

26 J Ian Munro and Corwin Sinnamon. Time and space efficient representations of distributive lattices. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 550–567. Society for Industrial and Applied Mathematics, 2018.

27 Arvind Narayanan, Narendran Thiagarajan, Mugdha Lakhani, Michael Hamburg, and Dan Boneh. Location privacy via private proximity testing. In *NDSS*, volume 1, 2011.

28 Ilan Newman. Private vs. common random bits in communication complexity. *Information Processing Letters*, 39(2):67–71, 1991.

29 David Peleg. Informative Labeling Schemes for Graphs. *Theoretical Computer Science*, 340(3):577–593, 2005.

30 Richard Rado. Universal graphs and universal functions. *Acta Arithmetica*, 4(9):331–340, 1964.

31 Mert Sağlam. Near log-convexity of measured heat in (discrete) time and consequences. In *59th Annual Symposium on Foundations of Computer Science (FOCS 2018)*, pages 967–978. IEEE, 2018.

32 Walter Schnyder. Planar graphs and poset dimension. *Order*, 5(4):323–343, 1989.

33 Jeremy P Spinrad. *Efficient Graph Representations*. American Mathematical Society, 2003.

34 Andrew C.C. Yao. Some complexity questions related to distributive computing. In *Proceedings of the Eleventh Annual ACM Symposium on Theory of Computing*, pages 209–213. ACM, 1979.

A Appendix

Proof of Proposition 2.5.

1. If $A \sqsubset B$ and $B \sqsubset C$ with ϕ, ψ being the respective embeddings then for all $u, v \in V(A)$ we have

$$C(\psi\phi(u), \psi\phi(v)) = B(\phi(u), \phi(v)) = A(u, v)\,.$$

2. In the "only if" direction, it suffices to choose $G^{\equiv}$. In the other direction, if $\phi : V(G) \to V(H)$ is an embedding and $\phi(u) = \phi(v)$ then for all $w \in V(G), G(u, w) = H(\phi(u), \phi(w)) = H(\phi(v), \phi(w)) = G(v, w)$ so $u \equiv v$.
3. Let g map a vertex of G to its equivalence class and let $u, v \in V(G)$. If $G(u, v) = 1$ then $G^{\equiv}(g(u), g(v)) = 1$ by definition. If $G^{\equiv}(g(u), g(v)) = 1$ then there exists $u' \in g(u), v' \in g(v)$ such that $G(u', v') = 1$, so $G(u, v) = G(u', v) = G(u', v') = 1$.
4. Let g map vertices in $V(G)$ to their equivalence class and let $g(u), g(v) \in V(G^{\equiv})$. If $g(u) \equiv g(v)$ then for any $w, G(u, w) = G^{\equiv}(g(u), g(w)) = G^{\equiv}(g(v), g(w)) = G(v, w)$ so $u \equiv v$ and therefore $g(u) = g(v)$. Therefore the map $g(u) \mapsto \{g(u)\}$ is an isomorphism $G^{\equiv} \to (G^{\equiv})^{\equiv}$.
5. If $G \sqsubset H$ then by transitivity, $G^{\equiv} \sqsubset G \sqsubset H \sqsubset H^{\equiv}$. Likewise, if $G^{\equiv} \sqsubset H^{\equiv}$ then $G \sqsubset G^{\equiv} \sqsubset H^{\equiv} \sqsubset H$.
6. If $G^{\equiv}$ is an induced subgraph of $H^{\equiv}$ then clearly there is an embedding. On the other hand, let $g(u), g(v) \in V(G^{\equiv})$ be the equivalence classes of $u, v \in V(G)$ and suppose there is an embedding $\phi : G^{\equiv} \to H^{\equiv}$. If $\phi(g(u)) = \phi(g(v))$ then $g(u) \equiv g(v)$ so $g(u) = g(v)$ since $(G^{\equiv})^{\equiv} \simeq G^{\equiv}$. Therefore $G^{\equiv}$ is an induced subgraph of $H^{\equiv}$. ◀

Approaching MCSP from Above and Below: Hardness for a Conditional Variant and $\mathsf{AC}^0[p]$

Rahul Ilango
Massachusetts Institute of Technology, Cambridge, MA, USA
rilango@mit.edu

Abstract

The Minimum Circuit Size Problem (MCSP) asks whether a given Boolean function has a circuit of at most a given size. MCSP has been studied for over a half-century and has deep connections throughout theoretical computer science including to cryptography, computational learning theory, and proof complexity. For example, we know (informally) that if MCSP is easy to compute, then most cryptography can be broken. Despite this cryptographic hardness connection and extensive research, we still know relatively little about the hardness of MCSP unconditionally. Indeed, until very recently it was unknown whether MCSP can be computed in $\mathsf{AC}^0[2]$ (Golovnev et al., ICALP 2019).

Our main contribution in this paper is to formulate a new "oracle" variant of circuit complexity and prove that this problem is NP-complete under randomized reductions. In more detail, we define the Minimum Oracle Circuit Size Problem (MOCSP) that takes as input the truth table of a Boolean function f, a size threshold s, and the truth table of an oracle Boolean function $\mathcal{O}$, and determines whether there is a circuit with $\mathcal{O}$-oracle gates and at most s wires that computes f. We prove that MOCSP is NP-complete under randomized polynomial-time reductions.

We also extend the recent $\mathsf{AC}^0[p]$ lower bound against MCSP by Golovnev et al. to a lower bound against the circuit minimization problem for depth-d formulas, (AC^0_d)-MCSP. We view this result as primarily a technical contribution. In particular, our proof takes a radically different approach from prior MCSP-related hardness results.

2012 ACM Subject Classification Theory of computation → Circuit complexity; Theory of computation → Problems, reductions and completeness

Keywords and phrases Minimum Circuit Size Problem, reductions, NP-completeness, circuit lower bounds

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.34

Related Version A full version of this paper is available at `https://eccc.weizmann.ac.il/report/2019/021/`.

Funding This work was supported (in part) by an Akamai Presidential Fellowship.

Acknowledgements I would like to give a special thanks to Eric Allender for innumerable suggestions and perspectives during all stages of this work. To name just a single example, one of his suggestions led me to improve a PARITY-reduction to the presented MAJORITY-reduction for (AC^0_d)-MCSP. I would also like to thank Abhishek Bhrushundi and Aditi Dudeja for their help in results about constant depth formulas that lead to this paper. I am grateful to Ryan Williams for asking interesting questions and helping to improve the exposition of the paper. Finally, I thank Harry Buhrman, Lance Fortnow, Igor Oliveira, Ján Pich, Aditya Potukuchi, Ninad Rajgopal, Michael Saks, and Rahul Santhanam for answering my questions and engaging in many useful discussions about this work.

1 Introduction

The Minimum Circuit Size Problem (MCSP) takes as input a Boolean function f (represented by its truth table) and a size parameter s and asks if there is a circuit of size at most s computing f. Study of this problem began in the 1950s by complexity theorists in the Soviet Union [31], where MCSP was of such great interest that Levin is said to have delayed

publishing his initial NP-completeness results in hope of showing that MCSP is NP-complete.[1] Interest in MCSP was revitalized when Kabanets and Cai [19] connected the problem with the natural proofs framework of Razborov and Rudich [28]. Since then, MCSP has been the subject of intense research. We begin by (non-exhaustively) reviewing this work.[2]

1.1 Known lower bounds, hardness, and non-hardness for MCSP

It is easy to see that MCSP is in NP (the circuit of size at most s can be used as a witness), but, despite work by numerous researchers, the exact complexity of MCSP remains unknown.

Hardness results for MCSP. We believe MCSP is not easy to compute. Kabanets and Cai [19] show that MCSP $\notin$ P conditioned on a widely-believed cryptographic hypothesis.

Thus, the most natural question about MCSP's complexity is whether it is NP-complete. As of yet, we have not managed to uncover even strong supporting evidence for, or against, MCSP being NP-complete. The strongest known hardness result is by Allender and Das [3] who show MCSP is hard for the cryptographically important class SZK under randomized reductions.

Under less powerful types of reductions, we only know hardness for some subclasses of P. For instance, building on the results of Oliveira and Santhanam [26], Golovnev et al. [12] show that the class of functions computable by polynomial-sized formulas (NC^1) reduces to MCSP under AC^0 reductions.

Lower Bounds for MCSP. Unconditionally, we know lower bounds against MCSP for several restricted computational models. Cheraghchi, Kabanets, Lu, and Myrisiotis [10] prove lower bounds for MCSP against DeMorgan formulas, branching programs, and AC^0 circuits that essentially match the best known explicit lower bounds for these models. Golovnev et al. [12] show that MCSP requires exponential-sized $\mathsf{AC}^0[p]$ circuits by proving MAJORITY $\in (\mathsf{AC}^0)^{\mathsf{MCSP}}$. The MAJORITY hardness result of [12] generalizes to the circuit minimization problem for many circuit classes, however, the techniques fail in the case of constant depth formulas.

Non-hardness results for MCSP. Surprisingly, we know MCSP cannot be NP-hard under certain types of reductions. For example, Hirahara and Watanabe [16] show that "oracle-independent polynomial reductions" cannot show that MCSP is hard for a class larger than P. Moreover, while essentially all NP-complete problems are complete under rather weak reductions such as uniform $\mathsf{TIME}[n^{o(1)}]$ reductions (which we do not define here), Murray and Williams [23] prove that MCSP is not even NP-hard under $\mathsf{TIME}[n^{.49}]$ reductions.

Conditioned on a widely-believed cryptographic hypothesis, Allender and Hirahara [4] show that a very weak approximation of MCSP is actually neither in P nor NP-complete.

Approaching MCSP from below. Given the apparent difficulty of resolving the complexity of MCSP, a natural approach is to progress towards MCSP "from below": considering restricted classes of circuits $\mathcal{C}$ that we understand better and determining whether the $\mathcal{C}$-circuit minimization problem, ($\mathcal{C}$)-MCSP, is NP-hard.

---

[1] [6] cites a personal communication from Levin regarding this.

[2] In our review, we are sometimes slightly informal and imprecise in order to be more accessible. We encourage the reader to look at the corresponding references for precise statements.

As it stands, we know that minimizing DNF formulas and minimizing $\mathsf{OR} \circ \mathsf{AND} \circ \mathsf{MOD}_m$ circuits are both NP-hard by Masek [20] and by Hirahara, Oliveira, and Santhanam [15] respectively. It is still open whether the circuit minimization problem for, say, depth-3 formulas is NP-hard.

1.2 Implications of lower bounds and hardness for MCSP

Since the main goal of this paper is to shed light on the complexity of MCSP, it is worth motivating why the complexity of MCSP is important.

Indeed, while researchers have not yet managed to establish the complexity of MCSP, a series of works, beginning with Kabanets and Cai [19], connect the computational complexity of MCSP and its variants to longstanding open questions in the field.

Separations of complexity classes. Several works ([17], [23], [3], [19]) show that MCSP being NP-hard, under various notions of reducibility, implies unknown class separations. For example, Hitchcock and Pavan [17] and Murray and Williams [23] show that if MCSP is NP-hard under polynomial-time reductions, then $\mathsf{ZPP} \neq \mathsf{EXP}$, a major open problem.[3]

Worst-case versus average-case complexity for NP. Using tools developed by Nisan and Wigderson [24] and Carmosino, Impagliazzo, Kabanets, and Kolokova [9], Hirahara [14] gives a "worst-case to average-case" reduction for NP conditioned on a certain approximation to MCSP being NP-hard. Thus, if one could show this approximation to MCSP is NP-hard, the worst-case and average-case complexity of NP would be equivalent.

Circuit Lower Bounds. Recent work ([22], [27], [25]) explores a "hardness magnification" phenomenon whereby even weak circuit lower bounds on certain computational problems imply strong lower bounds on other problems. For example, Mckay, Murray, and Williams [22] show that if MCSP (under a small size parameterization) cannot be solved by circuits of size $n \cdot \mathsf{poly}(\log n)$, then NP does not have polynomial-size circuits.

1.3 Our Contributions

In this work, we focus on hardness results for natural variants of MCSP with the aim of shedding light on the complexity of MCSP itself.

NP-Hardness of oracle MCSP (MOCSP)

As mentioned previously, some research has approached the problem of proving NP-hardness for MCSP "from below," that is, showing that the circuit minimization problem is NP-hard for restricted classes of circuits like DNF circuits and $\mathsf{OR} \circ \mathsf{AND} \circ \mathsf{MOD}_m$ circuits.

In this paper, we attempt to approach MCSP "from above." That is, we study the complexity of a problem that is at least as hard as MCSP. In particular, we define the *Minimum Oracle Circuit Size Problem* (MOCSP) to take as input a truth table T, a size parameter $s \in \mathbb{N}$, and an oracle truth table $\mathcal{O}$. The goal is to determine whether there is an $\mathcal{O}$-oracle circuit with at most s wires that computes T. It is easy to see that $\mathsf{MOCSP} \in \mathsf{NP}$ (the oracle circuit of size s can still act as an NP-witness).

[3] [23] only shows the result under many-one reductions, but their techniques easily generalize to the truth table case. [17] explicitly proves the truth table result using a different approach than [23].

A natural way to think of MOCSP is as a "conditional" version of MCSP where we are now measuring the complexity of computing T when we are given easy access to $\mathcal{O}$.

Although our problem definition is new, this is not the first time someone has considered an "oracle version" of MCSP. Several papers ([2, 5, 16, 18]) study the problem $\mathsf{MCSP}^{\mathsf{A}}$ of minimizing oracle circuits for a *fixed* oracle *language* A. MOCSP differs fundamentally from $\mathsf{MCSP}^{\mathsf{A}}$ in that the oracle circuit gets access to an *arbitrary* Boolean function, not a language, and the function the oracle circuit has access to is an *input* to the problem.

Furthermore, while $\mathsf{MOCSP} \in \mathsf{NP}$, the complexity of $\mathsf{MCSP}^{\mathsf{A}}$ depends on the oracle A. For example, the reference [2] proves $\mathsf{MCSP}^{\mathsf{QBF}}$ is complete for PSPACE under randomized reductions. Moreover, there is a trivial reduction from MCSP to MOCSP: simply provide no oracle truth tables. Therefore, since MOCSP is still in NP and MOCSP and is at least as hard as MCSP, MOCSP is a nice "testing ground" for hardness results we conjecture for MCSP.

Indeed, as is conjectured for MCSP, we *can* prove that MOCSP is NP-hard under randomized reductions.

▶ **Theorem 1.1.** MOCSP *is* NP *hard under:*
- *randomized many-one reductions with one-sided error, and*
- *randomized Turing reductions with zero-sided error.*

These NP-hardness results are both proved by giving a reduction from approximating set cover to MOCSP. It is worth noting that the NP-hardness results of (DNF)-MCSP [20] and $(\mathsf{OR} \circ \mathsf{AND} \circ \mathsf{MOD}_m)$-MCSP [15] are also proved via set cover problems, although both of these reductions are deterministic while our reduction is randomized.

Given that we can show MOCSP is NP-hard under randomized reductions, one might even begin to hope that we can prove hardness under, say, polynomial-time reductions. Unfortunately, this seems difficult. Essentially the same proofs Murray and Williams [23] or Hitchcock and Pavan [17] use to show that MCSP being NP-hard under polynomial-time truth table reductions implies $\mathsf{EXP} \neq \mathsf{ZPP}$ also work for MOCSP.

▶ **Theorem 1.2** (Essentially proven in [17] and [23])**.** *If* NP *reduces to* MOCSP *under polynomial time (truth table) reductions, then* $\mathsf{EXP} \neq \mathsf{ZPP}$*.*

Thus, improving our reduction to a polynomial-time truth table reduction requires separating EXP from ZPP, a longstanding open problem. For completeness, we give the MOCSP version of Murray and Williams' proof in Appendix B.

Even so, we expect that the ground truth is that MOCSP is NP-hard under, at least, deterministic polynomial-time Turing reductions.

▶ **Conjecture 1.3.** MOCSP *is* NP*-hard under polynomial-time Turing reductions.*

A perspective on MOCSP and some questions

In light of the fact that hardness for MCSP beyond SZK under even non-uniform reductions is unknown, we found these MOCSP hardness results to be quite surprising. Moreover, the NP-hardness of MOCSP seems to bolster the conjecture that MCSP is in fact NP-hard. To an optimist, NP-hardness results for MOCSP could even be a first step towards proving NP-hardness for MCSP. Indeed, a PSPACE-hardness result was first proved by Buhrman and Torenvliet [8] for an "oracle" version of space-bounded Kolmogorov complexity before Allender et al. [2] showed PSPACE-hardness for the non-oracle version about four years later.[4]

[4] The conference versions of [8] and [2] are four years apart.

Even if stronger hardness results for MCSP remain out of reach, MOCSP could still yield valuable insights about MCSP. For instance, it would be interesting to see which of the barriers and non-hardness results known for MCSP carry over to MOCSP.

▶ **Open Question 1.4.** *Can one show that other barriers or non-hardness results that hold for* MCSP *also hold for* MOCSP*?*

As an example of the insight given by answers to this question, consider Murray and Williams' [23] result that proving MCSP is NP-hard under polynomial-time reductions implies EXP $\neq$ ZPP. A natural question one might ask is whether we can improve this theorem to show that MCSP being NP-hard under randomized reductions implies unknown class separations. As we note in Theorem 1.2, however, Murray and Williams' proof carries over to MOCSP, and Theorem 1.1 shows MOCSP is indeed NP-hard under randomized reductions. Thus, any improvement of Murray and Williams' result to randomized reductions likely requires a fact about MCSP that we do not know for MOCSP.

In another direction, our results seem to imply that proving hardness for MOCSP is more tractable than proving hardness for MCSP. Since we have already noted that MOCSP can be used as a "testing ground" for questions about MCSP (since any hardness result for MCSP must also hold for MOCSP), one can explore whether other conjectured hardness results for MCSP hold for MOCSP. For example, Hirahara's [14] worst-case to average-case reduction for NP can be based on a certain approximation of MCSP being NP-hard, which would imply that a certain approximation of MOCSP is also NP-hard. Given that we can prove the NP-hardness of MOCSP under randomized reductions, we ask if one can prove something similar for the approximation version of MOCSP.

▶ **Open Question 1.5.** *Can one prove that, for some $\epsilon > 0$, approximating* MOCSP *on n-inputs to a factor of n^ϵ is* NP*-hard under, say,* P/poly *reductions? Conversely, can one prove that there is any barrier to showing such a hardness result?*

Finally, while in this paper we concentrate on approaching MCSP from above via MOCSP, it would be interesting to explore other problems that are harder than MCSP but still in NP. Perhaps this could help clarify what aspects of MCSP make it difficult to prove hardness for.

MAJORITY-hardness for (AC^0_d)-MCSP

As mentioned previously, Golovnev et al. [12] proves that MAJORITY $\in (\mathsf{AC}^0)^{\mathsf{MCSP}}$. Using similar techniques, they also show that, for restricted classes of circuits $\mathcal{C}$ such as formulas and constant depth circuits, the $\mathcal{C}$-circuit minimization problem, denoted $(\mathcal{C})$-MCSP, is hard for MAJORITY under AC^0 reductions. For these MAJORITY reductions to work, [12] requires that the size of the minimum $\mathcal{C}$-circuit on truth tables of length n is roughly $(n^{.49})$-Lipschitz, that is, flipping one bit in any truth table can change the $\mathcal{C}$-circuit complexity by at most an $n^{.49}$ additive term. This Lipschitzness hypothesis is unknown (and in our estimation probably false) in the class of depth-d *formulas*, which we denote AC^0_d.[5] As a result, until this work, it was unknown whether, say, (AC^0_d)-MCSP $\in \mathsf{AC}^0[2]$.

We stress that even though AC^0_d is an extremely restrictive class of formulas, (AC^0_d)-MCSP should be a very hard problem. For example, Allender *et al.* [1] prove that if (AC^0_d)-MCSP had polynomial-size circuits for sufficiently large d, then one could factor Blum integers using circuits of size 2^{n^ϵ} for all $\epsilon > 0$.

[5] We will always use the notation AC^0_d to refer to depth-d *formulas* and never depth-d *circuits*.

In this paper, we prove that (AC^0_d)-MCSP is indeed hard for MAJORITY by giving a win-win argument depending on whether the Lipschitzness hypothesis is true. Applying the lower bounds of Razborov [28] and Smolensky [30] then gives an $\mathsf{AC}^0[p]$ lower bound for (AC^0_d)-MCSP.

▶ **Theorem 1.6.** *Let $d \geq 2$. Then* MAJORITY $\leq^{\mathsf{AC}^0}_{tt}$ (AC^0_d)-MCSP. *Consequently, for any prime p,* (AC^0_d)-MCSP $\notin \mathsf{AC}^0[p]$.

One can view this theorem as (small) progress towards the program of approaching MCSP from below. The natural next step in the program is to prove that (AC^0_3)-MCSP is NP-hard, and, until this work, $\mathsf{AC}^0[p]$ lower bounds were open for this problem (despite knowing such $\mathsf{AC}^0[p]$ lower bounds for less restrictive circuit classes!). Perhaps the techniques introduced here could be useful in proving further hardness results for (AC^0_3)-MCSP.

Indeed, our techniques[6] may be the most interesting part of Theorem 1.6. They are entirely different than the ones used by [12] for general MCSP and, to our knowledge, all known MCSP hardness results. For example, a crucial step in the proof is introducing algebraic machinery and examining the size of a function's smallest circuits modulo a certain prime.

Indeed, all previously known MCSP-related reductions (to our knowledge) continue to work even when the reduction is given access to close approximations of a function's minimum circuit size. Because of this, it was not clear if, for the purpose of reductions, there was any difference between knowing whether a function has circuits of size exactly s or $s+1$. However, the algebraic nature of this MAJORITY reduction actually requires knowing such exact quantities, illustrating the existence of useful techniques that can distinguish between MCSP and approximations of MCSP.

1.4 Proof Overviews

In this section, we give fairly detailed overviews of our proofs. In doing this, we will often state results without filling in lower-level details. To make clear to the reader when we are doing this, we mark such sentences with an italicized *we observe*.

NP-hardness of Oracle MCSP (MOCSP)

Recall, the Minimum Oracle Circuit Size Problem, MOCSP, takes as input a truth table T, a threshold $s \in \mathbb{N}$, and an oracle truth table $\mathcal{O}$ and outputs whether there is an $\mathcal{O}$-oracle circuit with at most s wires that computes T. We denote the output of MOCSP on such an input as $\mathsf{MOCSP}(T, s; \mathcal{O})$. We denote the minimum size (measured by wires) of any $\mathcal{O}$-oracle circuit computing T as $\mathsf{CC}^{\mathcal{O}}(T)$.

We prove that MOCSP is NP-hard by giving a reduction from 6-approximating the Set Cover Problem (6-SetCover) to MOCSP. In our notation, 6-SetCover takes as input sets $S_1, \ldots, S_t \subseteq [n]$ whose union is $[n]$ as well as an integer $c \in [n]$ and requires outputting YES when $c \geq \ell$ and NO when $c < \ell/6$ where ℓ is the optimal cover size, i.e.

$$\ell = \min\{|I| : I \subseteq [t] \text{ and } \bigcup_{i \in I} S_i = [n]\}.$$

Approximating SetCover to an $((1 - o(1)) \ln n)$ factor is known to be NP-hard by Dinur and Steurer [11].

[6] Specifically, our techniques in the case when the Lipschitz hypothesis fails (which we expect to be the ground truth).

Informal idea. We begin by giving a high-level overview of the reduction to orient the reader. (It will be very informal, but we are building to a more detailed description.) Say we are given sets $S_1, \dots, S_t \subseteq [n]$ whose union is $[n]$. One can think of each of these sets S_i as "seeing" a small portion of the ground set $[n]$. Then, for a uniformly random truth table T of length $m \geq (nt)^5$, we let each set $S_1, \dots, S_t$ induce truth tables $T_{S_1}, \dots, T_{S_t}$ respectively where each truth table T_{S_i} has the same length at T and "sees" roughly the same part of T as S_i "sees" of $[n]$. We define our oracle $\mathcal{O}$ to be the concatenation of these truth tables. Finally, we ask how hard it is for a circuit to compute T given oracle access to $\mathcal{O}$.

The idea is that if a circuit looks at a collection of induced truth tables corresponding to a set cover, then it can "see" all of T and thus easily compute T. If not, then a large part of T is still random to the circuit and thus hard to compute. Indeed, we show that with high probability the quantity $\mathsf{CC}^{\mathcal{O}}(T)$ can be used to estimate ℓ up to a constant factor.

We now describe the reduction in more detail. Fix sets $S_1, \dots, S_t \subseteq [n]$ whose union is $[n]$ and, let ℓ be the optimal cover size of $[n]$ by $S_1, \dots, S_t$. Let T be a uniformly random truth table of length $m \geq (nt)^5$.

The truth tables induced by $S_1, \dots, S_t$ and T. We now rigorously define the truth tables $T_{S_1}, \dots, T_{S_t}$ of length m induced by $S_1, \dots, S_t$ and T. To do this, we chose a uniformly random partition of $[m]$ into n parts, that is, let $P : [m] \to [n]$ be a uniformly random function, and let $\mathcal{P} = (P_1, \dots, P_n)$ be the partition of $[m]$ into n parts where for $j \in [n]$

$$P_j = \{x \in [m] : P(x) = j\}.$$

We can then use this partition to "lift" any subset of $[n]$ into a subset of $[m]$ as follows. For a subset $S \subseteq [n]$, let S^m denote the subset of $[m]$ given by $S^m = \bigcup_{j \in S} P_j$.

Next, for a subset $S' \subseteq [m]$, we let $T_{\langle S' \rangle}$ be the truth table of length m that "sees" T on the elements of S' and zeroes everywhere else, that is

$$T_{\langle S' \rangle}(x) = T(x) \wedge \mathbb{1}_{x \in S'}$$

Finally, we define the truth table T_{S_i} induced by S_i to be the truth table $T_{\langle S_i^m \rangle}$ (we are dropping the m superscript and the bracketed subscript for concision).

Building the oracle $\mathcal{O}$. We build our oracle $\mathcal{O}$ by concatenating the induced truth tables $T_{S_1}, \dots, T_{S_t}$. In other words, let $\mathcal{O} : \{0,1\}^{\lceil \log t \rceil} \times \{0,1\}^{\log m} \to \{0,1\}$ be given by

$$\mathcal{O}(i, x) = \begin{cases} T_{S_i}(x) & , \text{ if } i \in [t] \\ 0 & , \text{ otherwise} \end{cases}$$

$\mathsf{CC}^{\mathcal{O}}(T)$ is at most $\ell(3 + \log(tm))$. Suppose, without loss of generality, that $S_1 \cup \dots \cup S_\ell$ is an optimal cover of $[n]$. We claim that $\bigvee_{i \in [\ell]} \mathcal{O}(i, x)$ is a circuit computing T. Unraveling our definitions, we have that

$$\bigvee_{i \in [\ell]} \mathcal{O}(i, x) = \bigvee_{i \in [\ell]} T_{S_i}(x) = \bigvee_{i \in [\ell]} (T(x) \wedge \mathbb{1}_{x \in S_i^m}) = T(x) \wedge (\bigvee_{i \in [\ell]} \mathbb{1}_{x \in S_i^m}) = T(x) \wedge \mathbb{1}_{x \in S_1^m \cup \dots \cup S_\ell^m}.$$

Using the fact that $S_1, \dots, S_\ell$ is a cover of $[n]$, we have that

$$T(x) \wedge \mathbb{1}_{x \in S_1^m \cup \dots \cup S_\ell^m} = T(x) \wedge \mathbb{1}_{x \in [m]} = T(x).$$

Thus, $\bigvee_{i \in [\ell]} \mathcal{O}(i, x)$ computes T and has at most

$$2\ell + \ell(1 + \log t + \log m) \leq \ell(3 + \log t + \log m)$$

wires.

With high probability, $\mathsf{CC}^{\mathcal{O}}(T) > \ell \log(tm)/2$. We do this by a union bound argument. Fix any oracle circuit C with at most $\ell \log(tm)/2$ wires. We show that the probability that $C^{\mathcal{O}}$ computes T is very small.

We begin by showing that on any fixed input x that

$$\Pr[C^{\mathcal{O}}(x) = T(x)] \leq 1 - \frac{1}{2n}.$$

Fix some x. We prove the following claim: that with probability at least $\frac{1}{n}$ over the choice of $P(x)$, the output of $C^{\mathcal{O}}(x)$ does not depend whether the value of $T(x)$ is one or zero. If this claim were true, then we would get the desired bound

$$\Pr[C^{\mathcal{O}}(x) = T(x)] \leq 1 - \frac{1}{2n}$$

since, assuming that $C^{\mathcal{O}}(x)$ does not depend on $T(x)$, the probability $C^{\mathcal{O}}(x) = T(x)$ is exactly $\frac{1}{2}$ (since $T(x)$ is chosen uniformly at random).

To prove the claim, we first note that the only way that the output of $C^{\mathcal{O}}(x)$ can depend on the value of $T(x)$ is if during the computation of $C^{\mathcal{O}}(x)$ C makes an oracle query (i', x') to $\mathcal{O}$ such that the value of $\mathcal{O}(i', x')$ depends on the value of $T(x)$. However, from the definition of $\mathcal{O}$, we know that the output of $\mathcal{O}(i', x')$ depends on $T(x)$ if and only if $x' = x$ and $P(x) \in S_{i'}$. Summarizing, $C^{\mathcal{O}}(x)$ depends on $T(x)$ only if $C^{\mathcal{O}}$ makes a query of the form (i', x) such that $P(x) \in S_{i'}$. But $C^{\mathcal{O}}$ has at most $\ell/2$ oracle gates, so it only has $\ell/2$ "tries" to find a set $S_{i'}$ that contains the uniformly random ground set element $P(x)$, which is less than the number of tries needed to cover the ground set! Hence, it is intuitive that with probability at least $\frac{1}{n}$, $P(x)$ takes on a value that is not in any of the $S_{i'}$ that $C^{\mathcal{O}}$ queries. *We observe* that this intuition can be made rigorous (with some more details), and hence, the claim is true.

At this point, we have completed our sketch of why for any fixed x

$$\Pr[C^{\mathcal{O}}(x) = T(x)] \leq 1 - \frac{1}{2n}.$$

Unfixing x, we would like to say that we can multiply these probabilities together for all $x \in [m]$ to bound the probability that $C^{\mathcal{O}}$ computes T. However, we cannot do this, as these events might have some dependencies between each other. Luckily, since, for any x, $C^{\mathcal{O}}(x)$ makes at most $\ell/2 \leq n$ oracle calls, *we observe* that applying the above probability bound can only "spoil" the bound for at most n other values of x. Hence, we can repeatedly apply the bound to about m/n different values of x yielding

$$\Pr[C^{\mathcal{O}} \text{ computes } T] \leq (1 - \frac{1}{2n})^{m/n} \leq e^{-\frac{m}{2n^2}} \leq e^{-n^3 t^5/2}.$$

Finally, union bounding over the at most

$$2^{O(\ell \log(tm)/2)^2} \leq 2^{O(n^2 t)}$$

oracle circuits with at most $\ell \log(tm)/2$ wires yields the desired result.

RP and ZPP reductions. At this point, we have shown that if $S_1, \ldots, S_t$ admits a c-cover, then $\mathsf{MOCSP}(T, c(3 + \log(tm)); \mathcal{O}) = \mathsf{YES}$ (note that this fact did not depend on our random choice of T or $\mathcal{P}$), and we have shown that if $S_1, \ldots, S_t$ does not admit a c-cover, then $\mathsf{MOCSP}(T, c \log(tm)/2; \mathcal{O}) = \mathsf{NO}$ with high probability. It is easy to see that this yields an RP many-one reduction (a randomized many-one reduction with one-sided error) from 6-SetCover to MOCSP.

Furthermore, since this RP reduction only errs by incorrectly classifying NO instances as YES instances, we can use the search-to-decision reduction for SetCover to check the correctness of purported YES instances, transforming our RP reduction into a ZPP-Turing reduction (a randomized zero-error reduction that can make multiple adaptive queries to MOCSP).

Majority Hardness for (AC^0_d)-MCSP

Recall, AC^0_d is the class of depth-d formulas made up of AND and OR gate with unbounded fan-in. We also define $\mathsf{AND} \circ \mathsf{AC}^0_{d-1}$ and $\mathsf{OR} \circ \mathsf{AC}^0_{d-1}$ be the classes of AC^0_d formulas with a top AND and top OR gate respectively. For $\mathcal{F} \in \{\mathsf{AC}^0_d, \mathsf{AND} \circ \mathsf{AC}^0_{d-1}, \mathsf{OR} \circ \mathsf{AC}^0_{d-1}\}$ and a truth table T, we let $\mathsf{CC}_{\mathcal{F}}(T)$ denote the size of the minimum $\mathcal{F}$-formula computing T where the size of a formula is the number of input leaves.

Our analysis proceeds by considering each $n \in \mathbb{N}$ and splitting into cases depending on whether $\mathsf{CC}_{\mathsf{AC}^0_d}$ is Lipschitz on truth tables of length around n. In more detail, fix some sufficiently large n. Let $q = \Theta(n^2)$ be a power of two. We divide into cases depending on whether there exists an $m \in \{q^{10}, q^{50}\}$ such that $\mathsf{CC}_{\mathsf{AC}^0_d}$ is $(m^{.25})$-Lipschitz on truth tables of length m.

Case 1: Lipschitzness holds for some m

If there does exist an $m \in \{q^{10}, q^{50}\}$ such that $\mathsf{CC}_{\mathsf{AC}^0_d}$ is $(m^{.25})$-Lipschitz on truth tables of length m, then the techniques of Golovnev *et al.* [12] yield an AC^0 truth table reduction from MAJORITY on n-bits to (AC^0_d)-MCSP on m-bits. For completeness, we include a self-contained proof of this case in Appendix A.

Case 2: Lipschitzness fails

Assume that for all $m \in \{q^{10}, q^{50}\}$ $\mathsf{CC}_{\mathsf{AC}^0_d}$ is not $(m^{.25})$-Lipschitz on truth tables of length m. Let $u = q^{10}$ and $v = q^{50}$.

Lipschitzness failing $\implies$ functions easier to compute with a top AND gate. *We observe*, as a straight forward consequence of Lipschitzness failing, that there exists a truth table of length u that has an optimal formula with large top fan-in and a truth table of length v that is easier to compute with a top AND gate:

1. There exists a Boolean function f^u that takes $\log u$ inputs and an AC^0_d formula ϕ^u such that ϕ^u is an optimal AC^0_d formula for f^u and $\phi^u = \phi^u_1 \wedge \cdots \wedge \phi^u_t$ for some $t \geq n$ and some $\phi^u_1, \ldots, \phi^u_t \in \mathsf{AC}^0_{d-1}$.
2. There exists a Boolean function f^v that takes $\log v$ inputs such that $\mathsf{CC}_{\mathsf{OR} \circ \mathsf{AC}^0_{d-1}}(f^v) > \mathsf{CC}_{\mathsf{AND} \circ \mathsf{AC}^0_{d-1}}(f^v) + u \log u$.

We will make use of f^u and ϕ^u to reduce MAJORITY to $\mathsf{CC}_{\mathsf{AND} \circ \mathsf{AC}^0_{d-1}}$ and we will use f^v to reduce $\mathsf{CC}_{\mathsf{AND} \circ \mathsf{AC}^0_{d-1}}$ to $\mathsf{CC}_{\mathsf{AC}^0_d}$.

Using $\mathsf{CC}_{\mathsf{AND} \circ \mathsf{AC}^0_{d-1}}$ and optimal subformulas of ϕ^u to compute a dot product. The heart of our MAJORITY reduction is a fairly elementary observation about optimal $(\mathsf{AND} \circ \mathsf{AC}^0_{d-1})$ formulas. Recall, $\phi^u = \phi^u_1 \wedge \cdots \wedge \phi^u_t$ is an optimal (AC^0_d) formula for f^u and, hence, also an optimal $(\mathsf{AND} \circ \mathsf{AC}^0_{d-1})$ formula for f^u. *We observe* that for any $A \subseteq [t]$, the $(\mathsf{AND} \circ \mathsf{AC}^0_{d-1})$-optimality of ϕ^u implies that $\bigwedge_{i \in A} \phi^u_i$ is also an optimal $(\mathsf{AND} \circ \mathsf{AC}^0_{d-1})$ formula for the function it computes.

Introducing some notation, for a string $x \in \{0,1\}^n$, we let f^u_x be the function given by $\bigwedge_{i \in O_x} \phi^u_i$ where $O_x \subseteq [n]$ are the bits in x that are one. Using the above observation about subformulas being optimal, we have that[7] $\mathsf{CC}_{\mathsf{AND} \circ \mathsf{AC}^0_{d-1}}(f^u_x) = \sum_{i \in O_x} |\phi^u_i|$. Thus, one can think of $\mathsf{CC}_{\mathsf{AND} \circ \mathsf{AC}^0_{d-1}}(f^u_x)$ as computing the dot product between x and the vector $\langle |\phi^u_1|, \ldots, |\phi^u_n| \rangle$.

Note that that the definition of f^u_x depends on the labeling of $\phi^u_1, \ldots, \phi^u_t$, in particular the choice of which ϕ^u_i have $i \leq n$. We will later choose a labeling of the ϕ^u_i that is convenient.

Computing MAJORITY (non-uniformly) using $\mathsf{CC}_{\mathsf{AND} \circ \mathsf{AC}^0_{d-1}}$. Our goal is to compute MAJORITY on a string $x \in \{0,1\}^n$ using the above "dot product" observation. Before we show how to do this, we give some intuition on how we came up with the idea.

Instead of trying to compute MAJORITY, suppose we relaxed the problem to computing PARITY given access to the integer produced by the dot product $x \cdot \langle |\phi^u_1|, \ldots, |\phi^u_n| \rangle$. Well, if it so happened that all the entries in the vector $\langle |\phi^u_1|, \ldots, |\phi^u_n| \rangle$ were odd, then it is clear that the integer produced by $x \cdot \langle |\phi^u_1|, \ldots, |\phi^u_n| \rangle$ is odd if and only if x has an odd number of ones. Our approach for MAJORITY is a generalization of this.

Let $p = O(n)$ be prime greater than n. *We observe*, via a pigeon-hole principle argument, that there exist integers $k \geq 0$ and $1 \leq r \leq p-1$ and indices $i_1, \ldots, i_n \in [m]$ such that

$$|\phi^u_{i_1}|/p^k \equiv \cdots \equiv |\phi^u_{i_n}|/p^k \equiv r \mod p.$$

Thus, after an appropriate relabeling of ϕ^u_i, we have that

$$|\phi^u_1|/p^k \equiv \cdots \equiv |\phi^u_n|/p^k \equiv r \mod p.$$

Hence, we can determine the weight w of x (and hence compute MAJORITY of x) by computing the value of

$$\mathsf{CC}_{\mathsf{AND} \circ \mathsf{AC}^0_{d-1}}(f^u_x)/p^k = \sum_{i \in O_x} |\phi^u_i|/p^k \equiv rw \mod p$$

and multiplying by the inverse of r modulo p. [8]

Reducing computing $\mathsf{CC}_{\mathsf{AND} \circ \mathsf{AC}^0_{d-1}}$ to computing $\mathsf{CC}_{\mathsf{AC}^0_d}$. Ultimately, we want to compute MAJORITY using $\mathsf{CC}_{\mathsf{AC}^0_d}$ not $\mathsf{CC}_{\mathsf{AND} \circ \mathsf{AC}^0_{d-1}}$. By the above procedure, it suffices to show how to compute $\mathsf{CC}_{\mathsf{AND} \circ \mathsf{AC}^0_{d-1}}(f^u_x)$ using $\mathsf{CC}_{\mathsf{AC}^0_d}$.

We can make such a computation as follows. Recall that f^v is a function satisfying

$$\mathsf{CC}_{\mathsf{OR} \circ \mathsf{AC}^0_{d-1}}(f^v) > \mathsf{CC}_{\mathsf{AND} \circ \mathsf{AC}^0_{d-1}}(f^v) + u \log u$$

whose existence is guaranteed by the failure of Lipschitzness. Take the direct product of f^u_x with f^v to obtain a function $g_x(y,z) = f^u_x(y) \wedge f^v(z)$. Since the difference between computing f^v with a top AND gate and a top OR gate is larger than $u \log u$ (which is the maximum complexity of f^u_x), *we observe*[9] any optimal AC^0_d formula for g_x must have a top AND gate, so

$$\mathsf{CC}_{\mathsf{AC}^0_d}(g_x) = \mathsf{CC}_{\mathsf{AND} \circ \mathsf{AC}^0_{d-1}}(g_x).$$

---

[7] Recall, our notion of formula size is the number of input leaves.

[8] In case the reader is unsure of whether the last parts of this procedure are implementable in AC^0, realize that the output of $\mathsf{CC}_{\mathsf{AND} \circ \mathsf{AC}^0_{d-1}}(f^u_x)$ is a binary string of length $O(\log n)$ and that any function on a string of length $O(\log n)$ can be computed by a polynomial-sized DNF. See the proof in Section 4 for more details.

[9] both the "*we observe" statements in this paragraph are consequences of standard direct product theorems for formulas.*

Then, *we observe* that

$$\mathsf{CC}_{\mathsf{AND}\circ\mathsf{AC}^0_{d-1}}(g_x) = \mathsf{CC}_{\mathsf{AND}\circ\mathsf{AC}^0_{d-1}}(f^u_x) + \mathsf{CC}_{\mathsf{AND}\circ\mathsf{AC}^0_{d-1}}(f^v).$$

Hence, if we are non-uniformly given the value of $\mathsf{CC}_{\mathsf{AND}\circ\mathsf{AC}^0_{d-1}}(f^v)$, we can subtract $\mathsf{CC}_{\mathsf{AND}\circ\mathsf{AC}^0_{d-1}}(f^v)$ from $\mathsf{CC}_{\mathsf{AC}^0_d}(g_x)$ to find $\mathsf{CC}_{\mathsf{AND}\circ\mathsf{AC}^0_{d-1}}(f^u_x)$.

▶ Remark 1.7. We remark that the only time we use the failure of the Lipschitzness hypothesis is to show the existence of functions like f^u with high top fan-in and functions like f^v with a large difference between top AND gate and top OR gate complexity. Using known PARITY lower bounds and depth-hierarchy theorems for AC^0 circuits, we can unconditionally prove the existence of f^u and f^v respectively but with slightly worse parameters that would yield a quasi-polynomial reduction (at least in the $d \geq 3$ case) rather than the polynomial reduction we present.

1.5 Paper Organization

In Section 2 we fix notation and review precise definitions. In Section 3 we prove our MOCSP results, and in Section 4 we prove that MAJORITY reduces to (AC^0_d)-MCSP.

2 Preliminaries

For an integer n, we let $[n]$ denote the set $\{1, \ldots, n\}$. For a binary string $x \in \{0,1\}^*$, the *weight* of x, denoted $\mathsf{wt}(x)$, is the number ones in x. We identify a Boolean function $f : \{0,1\}^n \to \{0,1\}$ with its truth table $T \in \{0,1\}^{2^n}$ and often use them interchangeably.

We let log denote the base-2 logarithm and ln represent the base-e logarithm. For functions f and g, we say $f = \tilde{O}(g)$ if there exists a c such that $f(x) \leq \log^c(g(x))g(x)$ for all sufficiently large x. We say that $f = \tilde{\Omega}(g)$ if $g = \tilde{O}(f)$.

We say a function $f : \{0,1\}^n \to \mathbb{N}$ is *c-Lipschitz* if for all $x, y \in \{0,1\}^n$ that differ in at most one bit, $|f(x) - f(y)| \leq c$. (In this paper, the function f for which we care about c-Lipschitzness is the function that maps a truth table to its circuit complexity.)

Complexity classes and reductions

We assume the reader is familiar with the standard complexity classes such as $\mathsf{AC}^0, \mathsf{P}, \mathsf{ZPP}, \mathsf{RP}, \mathsf{NP}, \mathsf{E}$ and the notion of Turing machines. For background on these, we refer to Arora and Barak's excellent textbook [7]. For us, AC^0 always refers to *non-uniform* AC^0.

We review the types of reductions we use in case the reader is not familiar with randomized reductions, truth table reductions, or our notation.

Many-one reductions. We will make use of the following notions of many-one reduction.

- $L \leq^{\mathsf{P}}_m L'$ if there is a polynomial-time Turing machine M such that $x \in L \iff M(x) \in L'$.
- $L \leq^{\mathsf{RP}}_m L'$ if there is a polynomial-time probabilistic Turing machine M taking in a "random" auxiliary input r such that

$$x \in L \implies \forall r\, M(x,r) \in L', \text{ and}$$

$$x \notin L \implies \Pr_r[M(x,r) \notin L'] \geq 2/3,$$

and $|r|$ is polynomial in the length of x.

Truth table reductions. We will also make use of the following notions of truth table reductions.

- We say an oracle circuit C is a *truth table oracle circuit* if there is no directed path between oracle gates in C.
- $L \leq_{tt}^{\mathsf{AC}^0} L'$ if there is a non-uniform (polynomial-sized) AC^0 truth table oracle circuit C such that C computes L when given oracle access to L'.

Turing reductions. Finally, we say $L \leq_{\mathsf{T}}^{\mathsf{ZPP}} L'$ if L can be computed with zero-error by a polynomial-time probabilistic oracle Turing machine M with oracle access to L'. On any single input, M is allowed to output "don't know" with probability at most $1/2$.

AC^0_d formulas, (AC^0_d)-MCSP, and $\mathsf{CC}_{\mathsf{AC}^0_d}$

For an integer $d \geq 2$, we let AC^0_d denote the class of depth-d formulas that use AND and OR gates with unbounded fan-in and fan-out 1 and that takes as "input leaves" the bits of a binary string and the negation of each of those bits.

For an AC^0_d formula ϕ, we define the size of ϕ, denoted $|\phi|$, to be the total number of input leaves ϕ uses. For a Boolean function f, we let $\mathsf{CC}_{\mathsf{AC}^0_d}(f)$ be the size of the smallest AC^0_d formula computing f.

▶ **Definition 2.1** (Minimum Circuit Size Problem for constant depth formulas). *(AC^0_d)-MCSP, is the language given by*

$$\{(T, s) \in \{0,1\}^\star \times \mathbb{N} : T \text{ is the truth table of a Boolean function, and } \mathsf{CC}_{\mathsf{AC}^0_d}(T) \leq s\}.$$

We will also make use of the classes of formulas $\mathsf{OR} \circ \mathsf{AC}^0_{d-1}$ and $\mathsf{AND} \circ \mathsf{AC}^0_{d-1}$, defined as the subclasses of AC^0_d formulas with a top OR gate and a top AND gate respectively. For $\mathcal{C} \in \{\mathsf{OR} \circ \mathsf{AC}^0_{d-1}, \mathsf{AND} \circ \mathsf{AC}^0_{d-1}\}$, we define $\mathsf{CC}_{\mathcal{C}}$ and $(\mathcal{C})$-MCSP analogous to $\mathsf{CC}_{\mathsf{AC}^0_d}$ and (AC^0_d)-MCSP.

We also require the following elementary lemmas regarding AC^0_d formulas.

▶ **Lemma 2.2.** *Let f be a Boolean function. Then $\mathsf{CC}_{\mathsf{AC}^0_d}(f) = \mathsf{CC}_{\mathsf{AC}^0_d}(\neg f)$.*

Proof. One can use DeMorgan's laws to turn any AC^0_d formula for f of size s into an AC^0_d formula for $\neg f$ of size s. ◀

We note that our specific notion of AC^0_d formula size is crucial for the next lemma.

▶ **Lemma 2.3** (Direct product theorem for formulas). *Let $f : \{0,1\}^n \to \{0,1\}$ and $g : \{0,1\}^m \to \{0,1\}$ be Boolean functions that are both not the constant zero function. Define $h : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}$ by $h(x,y) = f(x) \wedge g(y)$. Then*

$$\mathsf{CC}_{\mathsf{AND} \circ \mathsf{AC}^0_{d-1}}(h) = \mathsf{CC}_{\mathsf{AND} \circ \mathsf{AC}^0_{d-1}}(f) + \mathsf{CC}_{\mathsf{AND} \circ \mathsf{AC}^0_{d-1}}(g), \text{ and}$$

$$\mathsf{CC}_{\mathsf{OR} \circ \mathsf{AC}^0_{d-1}}(h) \geq \mathsf{CC}_{\mathsf{OR} \circ \mathsf{AC}^0_{d-1}}(f) + \mathsf{CC}_{\mathsf{OR} \circ \mathsf{AC}^0_{d-1}}(g).$$

Proof. It is easy to see that

$$\mathsf{CC}_{\mathsf{AND} \circ \mathsf{AC}^0_{d-1}}(h) \leq \mathsf{CC}_{\mathsf{AND} \circ \mathsf{AC}^0_{d-1}}(f) + \mathsf{CC}_{\mathsf{AND} \circ \mathsf{AC}^0_{d-1}}(g).$$

On the other hand, since f is not the constant 0 function, it has a 1-valued input x_0. Then, $h(x_0, y)$ computes $g(y)$. Thus, if ϕ is an $\mathsf{OR} \circ \mathsf{AC}^0_{d-1}$ formula for h, then ϕ has at least $\mathsf{CC}_{\mathsf{AND} \circ \mathsf{AC}^0_{d-1}}(g)$ y-input leaves. A similar argument shows that ϕ has at least $\mathsf{CC}_{\mathsf{AND} \circ \mathsf{AC}^0_{d-1}}(f)$ x-input leaves. Hence

$$\mathsf{CC}_{\mathsf{AND} \circ \mathsf{AC}^0_{d-1}}(h) \geq \mathsf{CC}_{\mathsf{AND} \circ \mathsf{AC}^0_{d-1}}(f) + \mathsf{CC}_{\mathsf{AND} \circ \mathsf{AC}^0_{d-1}}(g).$$

A similar argument shows that

$$\mathsf{CC}_{\mathsf{OR} \circ \mathsf{AC}^0_{d-1}}(h) \geq \mathsf{CC}_{\mathsf{OR} \circ \mathsf{AC}^0_{d-1}}(f) + \mathsf{CC}_{\mathsf{OR} \circ \mathsf{AC}^0_{d-1}}(g). \quad \blacktriangleleft$$

Oracle Circuits and Oracle MCSP: MOCSP

An oracle circuit C is made up of NOT gates, fan-in two AND and OR gates, and oracle gates $g_1, \ldots, g_t$ for some integer $t \geq 0$. When given an oracle functions $\mathcal{O}$, we let $C^{\mathcal{O}}(x)$ be the value obtained when evaluating C on input x by using the $\mathcal{O}$ function to compute the outputs of the oracle gates $g_1, \ldots, g_t$.

We define the size of an oracle circuit C, denoted $|C|$, to be the number of wires in C. For Boolean functions $f, \mathcal{O}$, we let $\mathsf{CC}^{\mathcal{O}}(f)$ be the size of the smallest $\mathcal{O}$-oracle circuit that computes f. Analogous to MCSP, we define the following.

▶ **Definition 2.4** (The Minimum Oracle Circuit Size Problem). *The Minimum Oracle Circuit Size Problem, denoted* MOCSP*, takes as input a truth table T, a threshold $s \in \mathbb{N}$, and an oracle truth table $\mathcal{O}$ and outputs whether $\mathsf{CC}^{\mathcal{O}}(T) \leq s$. The output of* MOCSP *on such an input is written as* $\mathsf{MOCSP}(T, s; \mathcal{O})$.

Set Cover

We will make use of the following well known NP-complete problem.

▶ **Definition 2.5** (Set Cover). *Set Cover, denoted* SetCover*, is the problem that takes as input a tuple $(n, c, S_1, \ldots, S_t)$, where $n \in \mathbb{N}$ is a universe size, $c \in \mathbb{N}$ is a proposed cover size $1 \leq c \leq n$, and $S_1, \ldots, S_t \subseteq [n]$ are sets whose union is $[n]$, and outputs whether $c \geq \ell$ where ℓ is the optimal cover size given by*

$$\ell = \min\{|I| : I \subseteq [t] \text{ and } \cup_{i \in I} S_i = [n]\}.$$

We will use that SetCover is NP-hard even to approximate to a roughly $\log n$ factor.

▶ **Theorem 2.6** (Dinur and Steurer [11]). *It is* NP*-hard to approximate* SetCover *to within a factor of $(1 - o(1)) \ln n$*

3 On the NP-hardness of MOCSP

First, we introduce some useful notation and definitions. For a truth table T of length m and a set $R \subseteq [m]$, let $T_{\langle R \rangle}$ be the truth table of length m where the jth bit of $T_{\langle R \rangle}$ equals

$$\begin{cases} \text{the } j\text{th bit of } T & \text{, if } j \in R \\ 0 & \text{, otherwise.} \end{cases}.$$

Equivalently,

$$T_{\langle R \rangle}(x) = T(x) \wedge \mathbb{1}_{x \in R}.$$

Next, we define a uniformly random partition $\mathcal{P} = (P_1, \ldots, P_n)$ of $[m]$ into n parts to be such that each element $i \in [m]$ is put into P_j where $j \in [n]$ is chosen uniformly at random. It will be also useful to think of $\mathcal{P}$ as a uniformly random function $P : [m] \to [n]$.

Next, for a partition $\mathcal{P} = (P_1, \ldots, P_n)$ of $[m]$ and any set $S \subseteq [n]$, we define the *$\mathcal{P}$-lift of* S, denoted $S^{\mathcal{P}}$, to be the subset of $[m]$ given by

$$S^{\mathcal{P}} = \bigcup_{i \in S} P_i.$$

Our main theorem shows that MOCSP can approximate set cover.

▶ **Theorem 3.1.** *Let $S_1, \ldots, S_t \subseteq [n]$ be sets that cover $[n]$. Let T be a truth table of length $m \geq (nt)^5$, and let $\mathcal{P} = (P_1, \ldots, P_n)$ be a uniformly random partition of $[m]$ into n parts. Define the oracle function the $\mathcal{O} : \{0,1\}^{\lceil \log t \rceil} \times \{0,1\}^{\log m} \to \{0,1\}$ by*

$$\mathcal{O}(i, y) = \begin{cases} T_{\langle S_i^{\mathcal{P}} \rangle}(y) & \text{, if } i \in [t] \text{ (identifying } i \text{ as an integer in the natural way)} \\ 0 & \text{, otherwise} \end{cases}.$$

Then

- $\mathsf{CC}^{\mathcal{O}}(T) \leq \ell(3 + \log m + \log t)$, *and*
- $\mathsf{CC}^{\mathcal{O}}(T) > \ell(\log m + \log t)/2$ *with high probability*

where ℓ is size of the optimal cover of $[n]$ by $S_1, \ldots, S_t$.

Proof. First, we show the upper bound.

▷ Claim 3.2. $\mathsf{CC}^{\mathcal{O}}(T) \leq \ell(3 + \log m + \log t)$

Proof. Without loss of generality, assume that the optimal cover size ℓ is witnessed by $S_1 \cup \cdots \cup S_\ell = [n]$. Then, by construction, the function computed by oracle circuit C given by

$$C(x) = \mathcal{O}(1, x) \vee \cdots \vee \mathcal{O}(\ell, x)$$

computes T and has at most

$$2\ell + \ell \cdot (\log m + \log t + 1) \leq \ell(3 + \log m + \log t)$$

wires. In more detail,

$$\bigvee_{i \in [\ell]} \mathcal{O}(i, x) = \bigvee_{i \in [\ell]} \bigvee_{j \in S_i} T_{\langle P_j \rangle}(x) = \bigvee_{j \in S_1 \cup \cdots \cup S_\ell} T_{\langle P_j \rangle}(x) = \bigvee_{j \in [n]} T_{P_j}(x) = T(x).$$

Therefore $\mathsf{CC}^{\mathcal{O}}(T) \leq \ell(3 + \log m + \log t)$. ◁

Now, we show the lower bound. We do this by a union bound argument. Fix some $\mathcal{O}$-oracle circuit C such that with at most $\ell(\log m + \log t)/2$ wires. Then C has at most $\ell/2$ oracle gates. We will show that

$$\Pr_{T, \mathcal{P}}[C^{\mathcal{O}} \text{ computes } T]$$

is exponentially small.

We do this by finding a long sequence of inputs $x_1, \ldots, x_d$ on which $C^{\mathcal{O}}$ has a not too large chance of computing T.

We construct this list of inputs recursively as follows. Let $x_1 = 0^{\log m}$, and let

$$Q_1 = \{x : C^{\mathcal{O}}(x_1) \text{ makes a query of the form } (i, x) \text{ to an oracle gate for some } i\}.$$

Now, for $j \geq 1$, if $\{0,1\}^{\log m} \setminus Q_j$ is non-empty, then let x_{j+1} be an element of $\{0,1\}^{\log m} \setminus Q_j$, and let

$$Q_{j+1} = Q_j \cup \{x : C^{\mathcal{O}}(x_{j+1}) \text{ makes a query of the form } (i, x) \text{ to an oracle gate for some } i\}.$$

If $\{0,1\}^{\log m} = Q_j$, then terminate the sequence.

Since C has at most $\ell/2 \leq n$ oracle gates, we know that $|Q_j| \leq j \cdot n$. Thus, since

$$|Q_d| = |\{0,1\}^{\log m}| = m,$$

the length d of this sequence is at least m/n.

It remains to bound

$$\Pr[C^{\mathcal{O}}(x_j) = T(x_j) \forall j] = \prod_{j=1}^{d} \Pr[C^{\mathcal{O}}(x_j) = T(x_j) | \bigwedge_{k \in [j-1]} C^{\mathcal{O}}(x_k) = T(x_k)].$$

Fix some $j \in [d]$. We will bound

$$\Pr[C^{\mathcal{O}}(x_j) = T(x_j) | \bigwedge_{k \in [j-1]} C^{\mathcal{O}}(x_k) = T(x_k)].$$

For convenience, let E denote the event we are conditioning on.

▷ Claim 3.3.

$$\Pr[C^{\mathcal{O}}(x_j) = T(x_j) | E] \leq 1 - \frac{1}{2n}.$$

Proof. By construction of the sequence $x_1, \ldots, x_d$, we know that on all the inputs $x_1, \ldots, x_{j-1}$, $C^{\mathcal{O}}$ does not make an oracle call of the form (i, x_j) for some i. Thus, since the only time the value of $\mathcal{O}$ depends on $T(x_j)$ and $P(x_j)$ is on inputs of the form (i, x_j) for some i, and $T(x_j)$ and $P(x_j)$ are chosen at independently at random, we know that $T(x_j)$ and $P(x_j)$ are still uniform random variables conditioned on E. In other words,

$$\Pr[T(x_j) = 1 | E] = \frac{1}{2}$$

and

$$\Pr[P(x_j) = r | E] = \frac{1}{n}$$

for all $r \in [n]$.

Now, define the alternate oracle $\mathcal{O}'$ that acts the same as $\mathcal{O}$ except that it rejects all queries containing x_j, that is

$$\mathcal{O}'(i, x) = \begin{cases} 0 & , \text{ if } x = x_j \\ \mathcal{O}(i, x) & , \text{otherwise.} \end{cases}$$

Now let $i_1, \ldots, i_v$ with $v \leq \ell/2$ be such that, using the modified oracle $\mathcal{O}'$, the only oracle queries $C^{\mathcal{O}'}(x_j)$ makes that end with x_j are $(i_1, x_j), \ldots, (i_v, x_j)$ (note that queries of this form are the only ones that could reveal information about $T(x_j)$). Since $v < \ell$ (recall ℓ is the size of the optimal cover) there actually exists an element $r^\star \in [n]$ that is not in $S_{i_1} \cup \cdots \cup S_{i_v}$.

Moreover, observe that if $P(x) = r^\star$, then $C^{\mathcal{O}}(x_j)$ will actually make the same oracle queries (and get the same zero responses) as the modified oracle circuit $C^{\mathcal{O}'}(x_j)$. In this case, since $P(x) = r^\star \notin S_{i_1} \cup \cdots \cup S_{i_v}$, it follows from the definition of $\mathcal{O}$ that

$$\mathcal{O}(i_1, x_j) = \cdots = \mathcal{O}(i_v, x_j) = 0$$

regardless of the value of $T(x_j)$. Thus, the output of $C^{\mathcal{O}}$ on input x does not depend at all on the value of $T(x)$. Hence, the probability it correctly guesses $C^{\mathcal{O}}(x) = T(x)$ is at most one half when $P(x_j) = r^\star$.

Since $P(x_j)$ is chosen uniformly at random, we have that $P(x_j) = r^\star$ with probability $1/n$.

Therefore, the probability

$$\Pr[C^{\mathcal{O}}(x_j) = T(x_j) | E] \leq 1 - \frac{1}{2n}$$ ◁

Using this claim, we now have

$$\prod_{j=1}^{d} \Pr[C^{\mathcal{O}}(x_j) = T(x_j) | \bigwedge_{k \in [j-1]} C^{\mathcal{O}}(x_k) = T(x_k)] \leq (1 - \frac{1}{2n})^d \leq e^{-\frac{d}{2n}} \leq e^{-\frac{m}{2n^2}} \leq e^{-\frac{n^3 t^5}{2}}.$$

On the other hand the number of oracle circuits of size $\ell(\log m + \log t)/2 = O(nt \log n)$ is at most $2^{O(n^2 t^2)}$. Thus, by the union bound the probability that there exists an $\mathcal{O}$-oracle circuit of size at most $\ell(\log m + \log t)/2$ computing T is $o(1)$ as desired. ◀

As an immediate consequence of Theorem 3.1, we get that MOCSP is hard for NP under many-one randomized reductions with one-sided error (the reduction is one-sided because the upper bound on the oracle circuit size of T in Theorem 3.1 is independent of the random choices of T and $\mathcal{P}$).

▶ **Corollary 3.4.** $\mathsf{NP} \leq_m^{\mathsf{RP}} \mathsf{MOCSP}$.

Moreover, the RP-reduction in Corollary 3.4 only errs by outputting YES when the answer should be NO. Thus, using the search-to-decision reduction for SetCover, we can check the correctness of purported YES answers, yielding a ZPP Turing reduction.

▶ **Corollary 3.5.** $\mathsf{NP} \leq_{\mathsf{T}}^{\mathsf{ZPP}} \mathsf{MOCSP}$.

4 MAJORITY $\leq_{tt}^{\mathsf{AC}^0}$ (AC^0_d)-MCSP

Let $d \geq 2$. Our goal in this section is to prove the following theorem.

▶ **Theorem 4.1.** $\mathsf{MAJORITY} \leq_{tt}^{\mathsf{AC}^0} (\mathsf{AC}^0_d)\text{-}\mathsf{MCSP}$.

We will do this by showing that for all sufficiently large $n \in \mathbb{N}$, there exists an AC^0 truth table oracle circuit that computes MAJORITY on n-bits when given access to (AC^0_d)-MCSP.

Fix some n, and let q be the least power of two such that $n \leq \sqrt{q}/2$. We will split our analysis into cases depending on whether there exists an $m \in \{q^{10}, q^{50}\}$ such that $\mathsf{CC}^0_{\mathsf{AC}^0_d}$ is $(m^{.25})$-Lipschitz on inputs of length m.

4.1 Case 1: Lipschitzness Holds

If Lipschitzness holds, then the desired (AC^0_d)-MCSP oracle circuit C exists for computing MAJORITY on n-inputs by the work of Golovnev et al. [12]. At a high-level, C works by using the input string to sample a random variable whose circuit complexity spikes (in expectation) depending on the weight of the input and using Lipschitzness to show that this spike happens with such high probability that it can be derandomized using non-uniformity.

For completeness, we give a self-contained proof of this case in Appendix A.

4.2 Case 2: Lipschitzness fails

Assume that for all $m \in \{q^{10}, q^{50}\}$, $\mathsf{CC}_{\mathsf{AC}^0_d}$ is not $(m^{.25})$-Lipschitz on inputs of length m. Thus, for all $m \in \{q^{10}, q^{50}\}$ there exist functions $f^m, h^m : \{0,1\}^{\log m} \to \{0,1\}$ that differ only on a single input $z^m \in \{0,1\}^{\log m}$ such that $\mathsf{CC}_{\mathsf{AC}^0_d}(h^m) - \mathsf{CC}_{\mathsf{AC}^0_d}(f^m) > m^{.25}$.

We assume, without loss of generality, that for all $m \in \{q^{10}, q^{50}\}$, $f^m(z^m) = 0$ and $h^m(z^m) = 1$. (If this is not the case, then replace f^m and h^m by $\neg f^m$ and $\neg h^m$ respectively and apply Lemma 2.2.)

First, we show that the failure of Lipschitzness implies the existence of functions that are much easier to compute by formulas with an AND gate on top. For $m \in \{q^{10}, q^{50}\}$ let $\mathbb{1}_{z^m} : \{0,1\}^{\log m} \to \{0,1\}$ denote the indicator function that accepts just the string z^m.

▶ **Proposition 4.2.** *Let $m \in \{q^{10}, q^{50}\}$. For sufficiently large n, $\mathsf{CC}_{\mathsf{OR}\circ\mathsf{AC}^0_{d-1}}(f^m) \geq \mathsf{CC}_{\mathsf{AC}^0_d}(f^m) + m^{.24}$, and so any optimal AC^0_d formula for f^m has an AND gate on top.*

Proof. Suppose ϕ is an $\mathsf{OR} \circ \mathsf{AC}^0_{d-1}$ formula computing f^m, that is, f^m is computed by $\phi = \phi_1 \vee \cdots \vee \phi_t$ for some AC^0_{d-1} formulas $\phi_1, \ldots, \phi_t$. Then

$$\mathbb{1}_{z^m} \vee \phi_1 \vee \cdots \vee \phi_t$$

computes h^m. Since $\mathbb{1}_{z^m}$ can be computed by a single AND gate of formula size $\log m$, this shows that $\mathsf{CC}_{\mathsf{AC}^0_d}(h^m) \leq |\phi| + \log m$. Combining this with the fact that $\mathsf{CC}_{\mathsf{AC}^0_d}(h^m) - \mathsf{CC}_{\mathsf{AC}^0_d}(f^m) \geq m^{.25}$ gives the desired result. ◀

At this point, we will need to refer to both q^{10} and q^{50} individually, so for convenience let $u = q^{10}$ and $v = q^{50}$.

Let ϕ^u be an optimal AC^0_d formula for f^u. By Proposition 4.2, for sufficiently large n, we know that $\phi^u = \phi^u_1 \wedge \cdots \wedge \phi^u_t$ for some AC^0_{d-1} formulas $\phi^u_1, \ldots, \phi^u_t$. Moreover, we can assume, without loss of generality, that the top gate of ϕ^u_i is OR for all $i \in [t]$. (If some ϕ^u_i has an AND gate on top, then this AND can be carried out by the AND gate on top of ϕ^u without increasing the size of the formula.)

Our next Proposition shows that ϕ^u has high top fan-in.

▶ **Proposition 4.3.** *For sufficiently large n,*

$$t \geq u^{.24}.$$

Proof. We divide into cases depending on d.

Case 1: $d \geq 3$. Realize that

$$(\phi^u_1 \vee \mathbb{1}_{z^u}) \wedge \cdots \wedge (\phi^u_t \vee \mathbb{1}_{z^u})$$

computes h^u. Since $\mathbb{1}_{z^u}$ can be computed by a single AND gate of formula size $\log u$ and the top gate of each ϕ^u_i is an OR gate and $d \geq 3$, this yields a depth-d formula for h^u of size $\mathsf{CC}_{\mathsf{AC}^0_d}(f^u) + t \log u$. Since $\mathsf{CC}_{\mathsf{AC}^0_d}(h^u) - \mathsf{CC}_{\mathsf{AC}^0_d}(f^u) \geq u^{.25}$, the desired bound on t follows.

Case 2: $d = 2$. Let $\mathbb{1}_{z^u,j} : \{0,1\}^{\log u} \to \{0,1\}$ be the function that accepts a string x if and only if the jth bit of x equals the jth bit of z^u. Observe that, since $\bigwedge_{i\in[t]} \phi_i^u$ computes f^u, we have that

$$\bigwedge_{i\in[t]} \bigwedge_{j\in[\log u]} (\phi_i^u \vee \mathbb{1}_{z^u,j})$$

computes h^u. Since $\mathbb{1}_{z^u,j}$ is computed by a single input leaf and ϕ_i^u has an OR gate on top, this yields a depth-2 formula for h^u of size $(|\phi^u|+1)\log u$. Since ϕ^u is an optimal CNF, each clause ϕ_i^u of ϕ^u is the OR of at most $\log u$ input leaves. In other words, $|\phi_i^u| \leq \log u$. Therefore, we have that

$$\mathsf{CC}^0_{\mathsf{AC}^0_d}(h^u) \leq (|\phi^u|+1)\log u \leq (\sum_{i\in[t]} |\phi_i^u|+1)\log u \leq (t\log u+1)\log u = t\log^2 u + \log u.$$

On the other hand, we know by assumption that

$$\mathsf{CC}^0_{\mathsf{AC}^0_d}(h^u) > \mathsf{CC}^0_{\mathsf{AC}^0_d}(f^u) + u^{.25} \geq u^{.25}.$$

Combining these two inequalities gives us the desired bound on t. ◀

Let p be smallest prime greater than n. (Note that $p \leq 2n$ by Betrand's postulate, also known as Chebyshev's theorem. See [13] for a proof.) We say that an integer j is (k,r)-*good* for integers $k \geq 0$ and $1 \leq r \leq p-1$ if p^k divides j and $j/p^k \equiv r \mod p$. In other words, an integer j is (k,r)-good for $k \geq 0$ and $r \in [p-1]$ if the kth largest entry of the base-p representation of the integer j equals r and all previous entries equal zero. From this "base-p" perspective, it is clear that all positive integers j are (k,r)-good for some $k \geq 0$ and $r \in [p-1]$.

We show that, for some k and r, a large subset of the $|\phi_i^u|$ are (k,r)-good.

▶ **Proposition 4.4.** *For all sufficiently large n, there exist integers $k \geq 0$ and $1 \leq r \leq p-1$ and a set $S \subseteq [t]$ of cardinality n such that, for all $i \in S$, the integer $|\phi_i^u|$ is (k,r)-good.*

Proof. We do this by an averaging argument. First, we show that each $|\phi_i^u|$ is (k,r)-good for a k not too large.

▷ Claim 4.5. For all $i \in [t]$, $|\phi_i^u|$ is (k,r)-good for some $0 \leq k \leq \log(u\log u)+1$ and some $r \in [p-1]$.

Proof. Fix some $i \in [t]$. $|\phi_i^u|$ is a positive integer, so $|\phi_i^u|$ is (k,r)-good for some $k \geq 0$ and some $r \in [p-1]$. We still need to upper bound this k. Note that the size of $|\phi_i^u|$ is at most $u\log u$ since ϕ^u is optimal for f^u and f^u can be computed by a DNF of size $u\log u$. Thus, for p^k to divide $|\phi_i^u|$, we must have that $k \leq \log_p(u\log u)+1 \leq \log(u\log u)+1$. ◁

Since for all $i \in [t]$ we have shown that $|\phi_i^u|$ is (k,r) good for some $0 \leq k \leq \log(u\log u)+1$ and some $r \in [p-1]$, a standard averaging argument implies that there exists a set $S \subseteq [t]$ of cardinality at least

$$\frac{t}{(\log(u\log u)+1)(p-1)}$$

such that for all $i \in S$, $|\phi_i^u|$ is (k,r)-good for some fixed $k \geq 0$ and $1 \leq r \leq p-1$. For sufficiently large n, we have that

$$\frac{t}{(\log(u\log u)+1)(p-1)} \geq \frac{u^{.24}}{4n\log u} \geq n$$

using that $u = q^{10} \geq n^{10}$. We then can truncate S so that it has only n elements as desired. ◀

Assume that n is large enough that all the sufficiently large hypotheses in Propositions 4.2, 4.3, and 4.4 apply. For convenience, relabel $\phi_1, \ldots, \phi_t$ so that the set S guaranteed by Proposition 4.4 is just $S = [n]$. Fix $k \geq 0$ and $r \in [p-1]$ to be the values such that for all $i \in S = [n]$, $|\phi_i^u|$ is (k, r)-good.

Introducing notation, for a set $A \subseteq [n]$, let f_A^u be the function computed by $\bigwedge_{i \in A} \phi_i^u$.

▶ **Lemma 4.6.** *Let $A \subseteq [n]$. Then $\mathsf{CC}_{\mathsf{AND} \circ \mathsf{AC}^0_{d-1}}(f_A^u) = \sum_{i \in A} |\phi_i^u|$.*

Proof. By construction, we have that $\mathsf{CC}_{\mathsf{AND} \circ \mathsf{AC}^0_{d-1}}(f_A^u) \leq \sum_{i \in A} |\phi_i|$. Suppose for contradiction that $\mathsf{CC}_{\mathsf{AND} \circ \mathsf{AC}^0_{d-1}}(f_A^u) < \sum_{i \in A} |\phi_i^u|$.

Let $\theta_1 \wedge \cdots \wedge \theta_\ell$ be a minimum-sized $(\mathsf{AND} \circ \mathsf{AC}^0_{d-1})$ formula for f_A^u. By assumption, we have that $\sum_{j=1}^{\ell} |\theta_j| < \sum_{i \in A} |\phi_i^u|$. We can thus replace the $\bigwedge_{i \in A} \phi_i^u$ in the optimal formula for f^u with $\theta_1 \wedge \cdots \wedge \theta_\ell$ and get a smaller formula. In more detail, we have that

$$f^u = f_A^u \wedge \Big(\bigwedge_{i \in [t] \setminus A} \phi_i^u \Big) = (\theta_1 \wedge \cdots \wedge \theta_\ell) \wedge \Big(\bigwedge_{i \in [t] \setminus A} \phi_i^u \Big)$$

which is a formula of size

$$\sum_{j=1}^{\ell} |\theta_j| + \sum_{i \in [t] \setminus A} |\phi_i^u| < \sum_{i \in A} |\phi_i^u| + \sum_{i \in [t] \setminus A} |\phi_i^u| = \sum_{i=1}^{t} |\phi_i^u| = |\phi^u|$$

which contradicts the optimality of ϕ^u for f^u. ◀

For a string $x \in \{0,1\}^n$, let f_x^u be shorthand for $f_{A_x}^u$ where $A_x \subseteq [n]$ is the set of indices where x is one.

▶ **Proposition 4.7.** *Let $x \in \{0,1\}^n$. Then x has weight w if and only if the integer $\mathsf{CC}_{\mathsf{AND} \circ \mathsf{AC}^0_{d-1}}(f_x^u)$ is (k, rw)-good.*

Proof. By Lemma 4.6 and the fact that $|\phi_i^u|$ is (k, r)-good for all $i \in [n]$, we have that

$$\frac{\mathsf{CC}_{\mathsf{AND} \circ \mathsf{AC}^0_{d-1}}(f_x^u)}{p^k} = \frac{\sum_{i \in A_x} |\phi_i^u|}{p^k} \equiv w \cdot r \mod p$$

where $A_x \subseteq [n]$ are bits of x that are ones. The "only if" part of the statement is guaranteed by the fact that $1 \leq r \leq p-1$ has a multiplicative inverse modulo p since p is prime. ◀

▶ **Theorem 4.8.** *Assume n is sufficiently large. Then there is a depth-8 AC^0 truth table oracle circuit C with $O(n^{250})$ wires such that $C^{(\mathsf{AC}^0_d)\text{-}\mathsf{MCSP}}$ computes $\mathsf{MAJORITY}$ on n-bits.*

Proof. It suffices to show that for every $w \in [n]$, there exists a depth-7 AC^0 oracle circuit C_w with $O(n^{249})$ wires such that $C_w^{(\mathsf{AC}^0_d)\text{-}\mathsf{MCSP}}(x) = 1 \iff \mathsf{wt}(x) = w$. Then $\mathsf{MAJORITY}(x) = \bigvee_{w \geq n/2} C_w(x)$.

Fix some $w \in [n]$. The circuit C_w works as follows. On input $x \in \{0,1\}^n$, first check if x is the all zeroes string. If so, then reject. Otherwise, compute the truth table of the direct product function $g_x : \{0,1\}^{\log u} \times \{0,1\}^{\log v} \to \{0,1\}$ given by $g_x(y,z) = f_x^u(y) \wedge f^v(z)$. Compute $s = \mathsf{CC}_{\mathsf{AC}^0_d}(g_x)$ in binary using oracle access to (AC^0_d)-MCSP. Finally accept if the integer s has the property that $s - \mathsf{CC}_{\mathsf{AC}^0_d}(f^v)$ is (k, rw)-good. Reject otherwise.

We now verify this yields a (non-uniform) AC^0 truth table oracle circuit. We can check if x is the all zeroes string with a single OR gate. This requires one level of depth and $O(n)$ wires. Next, realize the jth bit in the truth table of g_x is either zero for all x or equal to

$$f_x^u(j) = \bigvee_{i \in [n] : \phi_i^u(j) = 1} x_i$$

where x_i denotes the ith bit of x. Thus, using non-uniformity, we can compute the truth table of g_x with $O(nuv) = O(nq^{60}) = O(n^{121})$ wires and depth-one. Next, we can compute $s = \mathsf{CC}^0_{\mathsf{AC}^0_d}(g_x)$ in binary with $O(uv\log(uv))$ calls to (AC^0_d)-MCSP using the fact that $\mathsf{CC}^0_{\mathsf{AC}^0_d}(g_x) \leq uv\log(uv)$ by the DNF bound and the fact that

$$\mathsf{CC}^0_{\mathsf{AC}^0_d}(g_x) = s \iff (\mathsf{AC}^0_d)\text{-}\mathsf{MCSP}(g_x, s) = 1 \text{ and } (\mathsf{AC}^0_d)\text{-}\mathsf{MCSP}(g_x, s-1) = 0.$$

This takes at most $\tilde{O}((uv)^2) = O(n^{241})$ wires, an additional three layers of depth, and $2uv\log(uv)$ oracle calls that all do not depend on each other. Finally, the DNF upper bound guarantees that $\mathsf{CC}^0_{\mathsf{AC}^0_d}(g_x) \leq uv\log(uv) \leq n^{61}$, so the length of the integer $s = \mathsf{CC}^0_{\mathsf{AC}^0_d}(g_x)$ in binary is at most $61\log n$. Therefore we can check if s has the property that $s - \mathsf{CC}^0_{\mathsf{AC}^0_d}(f^v)$ is (k, rw)-good using a DNF with at most n^{62} wires and at most an additional two layers of depth. Combining all this yields a AC^0 circuit of depth-7 with at most $O(n^{241})$ wires and no directed path between oracle gates.

Next, we argue for correctness. Clearly, C_w rejects the all zero string, so assume $x \neq 0^n$. By Proposition 4.7, it suffices to show that, for $s = \mathsf{CC}^0_{\mathsf{AC}^0_d}(g_x)$,

$$s - \mathsf{CC}^0_{\mathsf{AC}^0_d}(f^v) = \mathsf{CC}^0_{\mathsf{AND}\circ\mathsf{AC}^0_{d-1}}(f^u_x).$$

We confirm that neither f^u_x nor f^v is the constant zero function, so that we can use the direct product theorems in Lemma 2.3.

▷ Claim 4.9. Neither f^u_x nor f^v is the constant zero function.

Proof. If f^v were the constant zero function, then $\mathsf{CC}^0_{\mathsf{AC}^0_d}(h^v) \leq \log v$ by DNF computation which contradicts that

$$\mathsf{CC}^0_{\mathsf{AC}^0_d}(h^v) - \mathsf{CC}^0_{\mathsf{AC}^0_d}(f^v) \geq v^{.25}.$$

Next, let $i \in [n]$ be a bit of x that is not zero. (Recall, we assumed that $x \neq 0^n$.) Then f^u_x has accepts every input that that ϕ^u_i accepts. For contradiction, suppose that ϕ^u_i had no ones. Then we can remove ϕ^u_i from the optimal formula $\phi^u = \phi^u_1 \wedge \ldots \phi^u_t$ for f^u and get a smaller formula for f^u which contradicts the optimality of ϕ^u. ◁

Next we show that the optimal AC^0_d formula for g_x has an AND gate on top.

▷ Claim 4.10. $\mathsf{CC}^0_{\mathsf{OR}\circ\mathsf{AC}^0_{d-1}}(g_x) > \mathsf{CC}^0_{\mathsf{AND}\circ\mathsf{AC}^0_{d-1}}(g_x)$. Consequently,

$$\mathsf{CC}^0_{\mathsf{AC}^0_d}(g_x) = \mathsf{CC}^0_{\mathsf{AND}\circ\mathsf{AC}^0_{d-1}}(g_x).$$

Proof. Let $\Delta = \mathsf{CC}^0_{\mathsf{OR}\circ\mathsf{AC}^0_{d-1}}(g_x) - \mathsf{CC}^0_{\mathsf{AND}\circ\mathsf{AC}^0_{d-1}}(g_x)$. We need to show $\Delta > 0$.

$$\begin{aligned}
\Delta &\geq \mathsf{CC}^0_{\mathsf{OR}\circ\mathsf{AC}^0_{d-1}}(f^v) - \mathsf{CC}^0_{\mathsf{AND}\circ\mathsf{AC}^0_{d-1}}(f^v) + \mathsf{CC}^0_{\mathsf{OR}\circ\mathsf{AC}^0_{d-1}}(f^u_x) && \text{(by Lemma 2.3)}\\
&\quad - \mathsf{CC}^0_{\mathsf{AND}\circ\mathsf{AC}^0_{d-1}}(f^u_x)\\
&\geq (v)^{.24} + \mathsf{CC}^0_{\mathsf{OR}\circ\mathsf{AC}^0_{d-1}}(f^u_x) - \mathsf{CC}^0_{\mathsf{AND}\circ\mathsf{AC}^0_{d-1}}(f^u_x) \geq && \text{(by Proposition 4.2)}\\
&\geq (v)^{.24} - u\log u && \text{(by DNF bound on } f^u_x)\\
&\geq n^{50\cdot.24} - n^{10}\log(n^{10}) && \text{(by definition of } u \text{ and } v)\\
&> 0 && \text{(for sufficiently large } n)
\end{aligned}$$

◁

Using the claim we have that

$$\begin{aligned}
& s - \mathsf{CC}^0_d(f^v) \\
=\ & \mathsf{CC}^0_d(g_x) - \mathsf{CC}^0_d(f^v) && \text{(definition)} \\
=\ & \mathsf{CC}^0_{d-1}(\mathsf{AND} \circ \mathsf{AC}^0)(g_x) - \mathsf{CC}^0_d(f^v) && \text{(Claim 4.10)} \\
=\ & \mathsf{CC}^0_{d-1}(\mathsf{AND} \circ \mathsf{AC}^0)(f^u_x) + \mathsf{CC}^0_{d-1}(\mathsf{AND} \circ \mathsf{AC}^0)(f^v) - \mathsf{CC}^0_d(f^v) && \text{(Lemma 2.3)} \\
=\ & \mathsf{CC}^0_{d-1}(\mathsf{AND} \circ \mathsf{AC}^0)(f^u_x) + \mathsf{CC}^0_{d-1}(\mathsf{AND} \circ \mathsf{AC}^0)(f^v) - \mathsf{CC}^0_{d-1}(\mathsf{AND} \circ \mathsf{AC}^0)(f^v) && \text{(Prop 4.2)} \\
=\ & \mathsf{CC}^0_{d-1}(\mathsf{AND} \circ \mathsf{AC}^0)(f^u_x)
\end{aligned}$$

as desired. ◀

References

1. E. Allender, L. Hellerstein, P. McCabe, T. Pitassi, and M. Saks. Minimizing DNF formulas and AC0d circuits given a truth table. In *21st Annual IEEE Conference on Computational Complexity (CCC'06)*, pages 15 pp.–251, July 2006.
2. Eric Allender, Harry Buhrman, Michal Koucký, Dieter van Melkebeek, and Detlef Ronneburger. Power from Random Strings. *SIAM J. Comput.*, 35(6):1467–1493, 2006.
3. Eric Allender and Bireswar Das. Zero Knowledge and Circuit Minimization. *Information and Computation*, 256:2–8, 2017.
4. Eric Allender and Shuichi Hirahara. New Insights on the (Non-)Hardness of Circuit Minimization and Related Problems. In *Symposium on Mathematical Foundations of Computer Science (MFCS)*, pages 54:1–54:14, 2017.
5. Eric Allender, Dhiraj Holden, and Valentine Kabanets. The Minimum Oracle Circuit Size Problem. *Computational Complexity*, 26(2):469–496, 2017.
6. Eric Allender, Michal Koucký, Detlef Ronneburger, and Sambuddha Roy. The pervasive reach of resource-bounded Kolmogorov complexity in computational complexity theory. *Journal of Computer and System Sciences*, 77(1):14–40, 2011.
7. Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009.
8. Harry Buhrman and Leen Torenvliet. Randomness is Hard. *SIAM Journal on Computing*, 30:200–1, 2000.
9. Marco L. Carmosino, Russell Impagliazzo, Valentine Kabanets, and Antonina Kolokolova. Learning Algorithms from Natural Proofs. In *Computational Complexity Conference (CCC)*, pages 10:1–10:24, 2016.
10. Mahdi Cheraghchi, Valentine Kabanets, Zhenjian Lu, and Dimitrios Myrisiotis. Circuit Lower Bounds for MCSP from Local Pseudorandom Generators. In *46th International Colloquium on Automata, Languages, and Programming, ICALP*, pages 39:1–39:14, 2019.
11. Irit Dinur and David Steurer. Analytical Approach to Parallel Repetition. In *Proceedings of the Forty-sixth Annual ACM Symposium on Theory of Computing*, STOC '14, pages 624–633, 2014.
12. Alexander Golovnev, Rahul Ilango, Russell Impagliazzo, Valentine Kabanets, Antonina Kolokolova, and Avishay Tal. AC0[p] lower bounds against MCSP via the coin problem. *Electronic Colloquium on Computational Complexity*, TR19-018, 2019.
13. G.H. Hardy and E. M. Wright. *An Introduction to the Theory of Numbers*. Clarendon Press, Oxford, England, 5th edition, 1979.
14. Shuichi Hirahara. Non-black-box Worst-case to Average-case Reductions within NP. In *Symposium on Foundations of Computer Science (FOCS)*, pages 247–258, 2018.
15. Shuichi Hirahara, Igor C. Oliveira, and Rahul Santhanam. NP-hardness of minimum circuit size problem for OR-AND-MOD circuits. In *Computational Complexity Conference (CCC)*, pages 5:1–5:31, 2018.

16 Shuichi Hirahara and Osamu Watanabe. Limits of minimum circuit size problem as oracle. In *Computational Complexity Conference (CCC)*, pages 18:1–18:20, 2016.

17 John Hitchcock and Aduri Pavan. On the NP-Completeness of the Minimum Circuit Size Problem. In *Conference on Foundation of Software Technology and Theoretical Computer Science (FSTTCS)*, 2015.

18 Russell Impagliazzo, Valentine Kabanets, and Ilya Volkovich. The Power of Natural Properties As Oracles. In *Proceedings of the 33rd Computational Complexity Conference*, CCC '18, pages 7:1–7:20, 2018.

19 Valentine Kabanets and Jin-Yi Cai. Circuit Minimization Problem. In *Proceedings of ACM Symposium on Theory of Computing (STOC)*, pages 73–79, 2000.

20 William J. Masek. Some NP-complete Set Covering Problems. Unpublished Manuscript, 1979.

21 Colin McDiarmid. On the method of bounded differences. In *Surveys in Combinatorics*, (Norwich, 1989), London Math. Soc. Lecture Note Ser. 141 . Cambridge Univ. Press, Cambridge, (1989), pp. 48–188.

22 Dylan M. McKay, Cody D. Murray, and R. Ryan Williams. Weak lower bounds on resource-bounded compression imply strong separations of complexity classes. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019.*, pages 1215–1225, 2019.

23 Cody D. Murray and R. Ryan Williams. On the (Non) NP-hardness of Computing Circuit Complexity. In *Computational Complexity Conference (CCC)*, pages 365–380, 2015.

24 Noam Nisan and Avi Wigderson. Hardness vs Randomness. *J. Comput. Syst. Sci.*, 49(2):149–167, 1994.

25 Igor C. Oliveira, Ján Pich, and Rahul Santhanam. Hardness magnification near state-of-the-art lower bounds. *Electronic Colloquium on Computational Complexity*, TR18-158, 2018.

26 Igor C. Oliveira and Rahul Santhanam. Conspiracies Between Learning Algorithms, Circuit Lower Bounds, and Pseudorandomness. In *Computational Complexity Conference (CCC)*, pages 18:1–18:49, 2017.

27 Igor C. Oliveira and Rahul Santhanam. Hardness Magnification for Natural Problems. In *Symposium on Foundations of Computer Science (FOCS)*, pages 65–76, 2018.

28 A. A. Razborov. Lower bounds on the size of constant-depth networks over a complete basis with logical addition. *Mathematicheskie Zametki*, 41(4):598–607, 1987.

29 Ronen Shaltiel and Emanuele Viola. Hardness Amplification Proofs Require Majority. *SIAM J. Comput.*, 39(7):3122–3154, 2010.

30 R. Smolensky. On representations by low-degree polynomials. In *FOCS*, pages 130–138, 1993.

31 Boris Trakhtenbrot. A Survey of Russian Approaches to Perebor (Brute-Force Searches) Algorithms. *IEEE Ann. Hist. Comput.*, 6(4):384–400, October 1984.

32 Stanislav Žák. A Turing machine time hierarchy. *Theoretical Computer Science*, 26(3):327–333, 1983.

A MAJORITY reduces to (AC^0_d)-MCSP when Lipschitzness holds

Our goal in this section is to find a small (AC^0_d)-MCSP-oracle circuit that computes MAJORITY on n-bits for sufficiently large n. We can do this using the techniques of Golovnev et al. [12]. In order to make our proof relatively self-contained, we differ slightly from the presentation in [12]. In particular, our presentation follows a method for computing MAJORITY that is described in Shaltiel and Viola [29].

At a high-level, this procedure works by using the input string to sample a random variable whose circuit complexity spikes depending on the weight of the input and then using Lipschitzness to prove that this spike occurs with high enough probability that we can derandomize using non-uniformity.

Continuing the notation from Section 4, assume that there is an $m \in \{q^{10}, q^{50}\}$ such that $\mathsf{CC}^0_{\mathsf{AC}^0_d}$ is $(m^{.25})$-Lipschitz on inputs of length m.

We define the random variable $T_{p,m} \in \{0,1\}^m$ where each bit in $T_{p,m}$ is independently chosen to be one with probability p and zero with probability $1-p$.

▶ **Lemma A.1.** $\mathbb{E}[\mathsf{CC}^0_{\mathsf{AC}^0_d}(T_{p,m})] = \tilde{O}(pm)$ *if* $p \geq m^{-1/3}$

Proof. By Hoeffding's inequality we have that the probability that $T_{p,m}$ has greater than k ones is at most $\mathsf{exp}(-2\epsilon^2 m)$. Via computation by DNF, if a truth table $T \in \{0,1\}^m$ has at most k ones, $\mathsf{CC}^0_{\mathsf{AC}^0_d}(T) = k \log m = \tilde{O}(k)$. Similarly, we have that $\max\{C(T) : T \in \{0,1\}^m\} = \tilde{O}(m)$. Hence, we get that

$$\mathbb{E}[\mathsf{CC}^0_{\mathsf{AC}^0_d}(T_{p,m})] = \tilde{O}(k) + \tilde{O}(\mathsf{exp}(-2\epsilon^2 m)m) = \tilde{O}(pm + pm\epsilon + \mathsf{exp}(-2\epsilon^2 m)m).$$

If we set $\epsilon = \sqrt{\frac{\ln m}{2m}}$, then we have

$$\mathbb{E}[\mathsf{CC}^0_{\mathsf{AC}^0_d}(T_{p,m})] \leq \tilde{O}(pm + p\sqrt{m}\ln m + 1) \leq \tilde{O}(pm + \sqrt{m}\ln m)$$

Finally, if $p \geq 1/m^{1/3}$, we have $\mathbb{E}[T_{p,m}] = \tilde{O}(pm)$ as desired ◀

We will make use of the following concentration inequality.

▶ **Theorem A.2** (McDiarmid's "bounded differences inequality" [21])**.** *Let* $f : \{0,1\}^n \to \mathbb{R}$ *be* c*-Lipschitz. Let* $X_1, \ldots, X_n$ *be independent random variables with values in* $\{0,1\}$*. Let* $\mu = \mathbb{E}_{X_1,\ldots,X_n}[f(X_1, \ldots, X_n)]$*. Then*

$$\Pr[|f(X_1, \ldots, X_n) - \mu| \geq \epsilon] \leq 2\mathsf{exp}(-\frac{\epsilon^2}{nc^2}).$$

For $t \in \mathbb{N}$ and $w_1 \neq w_2 \in [t]$, we say a Boolean function $f : \{0,1\}^t \to \{0,1\}$ computes $\mathsf{WTDIS}_t[w_1, w_2]$ if $\mathsf{wt}(x) = w_1$ implies $f(x) = 1$ and $\mathsf{wt}(x) = w_2$ implies $f(x) = 0$. (WTDIS is short for weight distinguishing.)

▶ **Theorem A.3.** *If* n *is sufficiently large, then for all* $1 \leq b \leq \sqrt{q}/2$*, there exists a (non-uniform)* NC^0 *oracle circuit* C *with at most* $O(n^{100})$ *wires such that* $C^{(\mathsf{AC}^0_d)\text{-MCSP}}$ *computes* $\mathsf{WTDIS}_q[w_1, w_1 + b]$ *for some* $w_1 \geq \sqrt{q}/2$*. Moreover,* C *has a single gate.*

Proof. For $w \in [q]$, let $p_w = \frac{w}{2q}$. Let w_0 be the largest integer less than $\sqrt{q}$ such that $q - w_0$ is a multiple of b. (Note that $w_0 \geq \sqrt{q} - b \geq \sqrt{q}/2$).

By Lemma A.1, we have that

$$\mathbb{E}[\mathsf{CC}^0_{\mathsf{AC}^0_d}(T_{p_{w_0},m})] = \tilde{O}(\frac{\sqrt{q}}{q}m) = \tilde{O}(m/\sqrt{q}).$$

On the other hand, since $p_q = 1/2$, $T_{p_q,m}$ is just a binary string of length m picked uniformly at random, so the formula size lower bounds of Shannon and Riordan imply

$$\mathbb{E}[\mathsf{CC}^0_{\mathsf{AC}^0_d}(T_{p_q,m})] = \mathop{\mathbb{E}}_{x\in\{0,1\}^m}[C(x)] = \tilde{\Omega}(m)$$

(note that an AC^0_d formula of size s implies an unrestricted formula of size s). Hence, by an averaging argument there exists a $w_1 \geq w_0 \geq \sqrt{q}/2$ such that

$$\mathbb{E}[\mathsf{CC}^0_d(T_{p_{w_1+b},m})] - \mathbb{E}[\mathsf{CC}(T_{p_{w_1},m})] \geq \frac{\tilde{\Omega}(m) - \tilde{O}(m/\sqrt{q})}{q} = \tilde{\Omega}(m/q).$$

Let $t = \frac{\mathbb{E}[T_{p_{w_1+b},m}]+\mathbb{E}[T_{p_{w_1},m}]}{2}$. Then we have that $\mathbb{E}[T_{p_{w_1+b},m}] - t = \tilde{\Omega}(m/q)$ and $t - \mathbb{E}[T_{p_{w_1},m}] = \tilde{\Omega}(m/q)$.

We now outline a probabilistic oracle circuit D that we will later make into a deterministic NC^0 circuit. D takes as input a string $x \in \{0,1\}^n$ and takes as its random "inputs" strings $u_1, \ldots, u_m \in \{0,1\}^{\log q}$ and $v_1, \ldots, v_m \in \{0,1\}$. The reduction then computes the string $y := y_1 \ldots y_m$ where y_i is zero if v_i is zero and y_i is the u_ith bit of x if v_i is one (recall, q is a power of two). D then outputs (AC^0_d)-$\mathsf{MCSP}(y,t)$.

We now argue for correctness with high probability. Realize each y_i is independent with probability $\frac{\mathsf{wt}(x)}{2n}$ of being 1. Hence, y is just the random variable $T_{p_w,m}$ where $w = \mathsf{wt}(x)$. Hence, if $\mathsf{wt}(x) = w_1$, then

$$\Pr[R(x) \neq 1] = \Pr[\mathcal{C}(T_{p_{w_1},m}) > t]$$

Recall that $t - \mathbb{E}[T_{p_{w_1},m}] = \tilde{\Omega}(m/q)$ and, by assumption, $\mathsf{CC}^0_{\mathsf{AC}^0_d}$ on inputs of length m is $(m^{.25})$-Lipschitz, so by Theorem A.2, we have that this probability is bounded by

$$2\mathsf{exp}(-2\frac{\tilde{\Omega}(m^2)}{\tilde{O}(q^2 m^{1.5})}) \leq \mathsf{exp}(-2\frac{\tilde{\Omega}(q^{.5\cdot 10})}{\tilde{O}(q^2)}) = O(\mathsf{exp}(-q^3))$$

using the fact that $m \geq q^{10}$. A similar analysis shows that the probability D errs if $\mathsf{wt}(x) = w_1 + b$ is at most $O(\mathsf{exp}(-q^3))$. This completes the analysis of D.

We now argue that this reduction can be derandomized using non-uniformity. For each input of weight either w_1 or $w_1 + b$, we have shown the fraction of random strings which err on that input is $O(\mathsf{exp}(-q^3))$. Hence, the fraction of random seeds which err on at least one input of weight w_1 or $w_1 + b$ is at most

$$2^q O(\mathsf{exp}(-q^3)) < 1$$

for large enough n. Thus, there exists some fixed $u_1, \ldots, u_m$ and $v_1, \ldots, v_m$ which work on all inputs of length q. Once we are (non-uniformly) given these $u_1, \ldots, u_m$ and $v_1, \ldots, v_m$ which work on all inputs, we can turn D into an NC^0 oracle circuit C which has just a single gate (an oracle gate) whose inputs are the fixed number t and the string y where each bit of y is either a fixed bit of x or zero. This yields a NC^0 oracle circuit with $O(m) = O(q^{50}) = O(n^{100})$ wires. ◀

▶ **Corollary A.4.** *If n is sufficiently large, then for all distinct $w_1, w_2 \in [n]$ there is an NC^0 oracle circuit C with at most two gates and $O(n^{100})$ wires such that $C^{(\mathsf{AC}^0_d)\text{-}\mathsf{MCSP}}$ computes $\mathsf{WTDIS}_n[w_1, w_2]$.*

Proof. Fix some $w_1 \neq w_2$. Without loss of generality assume $w_1 < w_2$ (if this is not the case, then swap the names of w_1 and w_2 in this proof and add a NOT gate to the top of C). Let $b = w_2 - w_1$. Recall q is the least power of two such that $n \leq \sqrt{q}/2$. Note that $q = \Theta(n^2)$ and $b \leq n \leq \sqrt{q}/2$. Theorem A.3 guarantees there exists an NC^0 oracle circuit D of size $O(n^{20})$ such that D^{MCSP} computes $\mathsf{WTDIS}_q[w_3, w_3 + b]$ for some $w_3 \geq \sqrt{q}/2 \geq n$. Finally, let C be the oracle circuit that on input x outputs $D(y)$ where $y = 1^{w_3 - w_1} 0^{q-n-w_3+w_1} x$. The correctness of this output is guaranteed by the fact that $\mathsf{wt}(y) = w_3$ if and only if $\mathsf{wt}(x) = w_1$ and $\mathsf{wt}(y) = w_3 + b$ if and only if $\mathsf{wt}(x) = w_2$. ◀

▶ **Corollary A.5.** *If n is sufficiently large, then there exists a depth-4* AC^0 *truth table oracle circuit C with $O(n^{102})$ wires such that $C^{(\mathsf{AC}^0_d)\text{-}\mathsf{MCSP}}$ computes* MAJORITY *on strings on length n.*

Proof. It suffices to show that, for all $w \in [n]$, one can check if a string $x \in \{0,1\}^n$ has weight w using a depth-3 AC^0 truth-table oracle circuit C_w of size $O(n^{101})$. If one is able to do this, then MAJORITY is computed by $\bigvee_{w \geq n/2} C_w(x)$.

For $w \in [n]$, let $\mathsf{wt}_w : \{0,1\}^n \to \{0,1\}$ be the Boolean function that outputs one if and only if its input is a string of weight w. Now fix some $w \in [n]$. We claim that

$$\mathsf{wt}_w(x) = \bigwedge_{w' \in [n]: w \neq w'} \mathsf{WTDIS}_n[w, w']$$

If x has weight w, then $\mathsf{WTDIS}_n[w,w'](x) = 1$ for all $w' \neq w$, so

$$\mathsf{wt}_{w'}(x) = 1 = \bigwedge_{w' \in [n]: w \neq w'} \mathsf{WTDIS}_n[w, w'].$$

On the other hand, if x has weight $w' \neq w$,then $\mathsf{WTDIS}_n[w,w'](x) = 0$, so

$$\mathsf{wt}_w(x) = 0 = \bigwedge_{w' \in [n]: w \neq w'} \mathsf{WTDIS}_n[w, w'].$$

Finally, by Corollary A.4 we have that $\bigwedge_{w \in [n]: w \neq w'} \mathsf{WTDIS}_n[w', w]$ is computable by a depth-3 AC^0 truth table oracle circuit with $O(n^{101})$ wires. ◀

B $\mathsf{NP} \leq^{\mathsf{P}}_{tt} \mathsf{MOCSP}$ implies $\mathsf{EXP} \neq \mathsf{ZPP}$

The proof of this result follows essentially exactly from Murray and Williams's [23] proof for MCSP. For completeness, we replicate the proof here (even using their words and structure).

▶ **Proposition B.1.** *If* $\mathsf{NP} \leq^{\mathsf{P}}_{tt} \mathsf{MOCSP}$*, then* $\mathsf{EXP} \subseteq \mathsf{P/poly}$ *implies* $\mathsf{EXP} = \mathsf{NEXP}$.

Proof. Assume $\mathsf{NP} \leq^{\mathsf{P}}_{tt} \mathsf{MOCSP}$ and $\mathsf{EXP} \subseteq \mathsf{P/poly}$. Let $L \in \mathsf{NTIME}(2^{n^c})$ for some $c \geq 1$. It suffices to show that $L \in \mathsf{EXP}$.

We pad L into the $L' = \{x01^{2^{|x|^c}} : x \in L\}$. Note that $L' \in \mathsf{NP}$. Hence there is a polynomial-time truth table reduction from L' to MOCSP. Composing the reduction from L to L' with the reduction from L' to MOCSP, we get a $2^{c'n^c}$-time truth table reduction R from n-bit instances of L to $2^{c'n^c}$-bit instances of MOCSP for some constant c'.

Let $Q(x)$ denote the concatenated string of all MOCSP queries produced by R in order on input x. Define the language

$$BITS_Q := \{(x,i) : \text{ the } i\text{th bit of } Q(x) \text{ is } 1\}$$

$BITS_Q$ is clearly in EXP. Since $\mathsf{EXP} \subseteq \mathsf{P/poly}$, for some $d \geq 1$ there is a circuit family C_n of size at most $n^d + d$ computing $BITS_Q$ on n-bit inputs.

Thus, on a given instance x, we have $\mathsf{CC}(Q(x)) \leq s(|x|)$ where $s(|x|) := (|x| + 2c'|x|^c)^d + d$. Therefore, every MOCSP query $(T, s', \mathcal{O})$ produced by the reduction R on input x satisfies

$$\mathsf{CC}^{\mathcal{O}}(T) \leq \mathsf{CC}(T) \leq e \cdot \mathsf{CC}(Q(x)) \leq e \cdot s(|x|)$$

for some constant e since T is a substring of $Q(x)$ (see Lemma 2.2 in [23] for a proof of this substring fact). This leads to the following exponential time algorithm for L:

> On input x, run the exponential-time reduction $R(x)$ by using the following procedure for answering each MOCSP oracle query $(T, s'; \mathcal{O})$. If $s' > e \cdot s(|x|)$, then respond YES to the query. Otherwise, cycle through every oracle circuit E of size at most s'. If $E^{\mathcal{O}}$ computes T, then respond YES. If no such E is found, then respond NO.

It suffices to show the procedure for answering MOCSP oracle queries runs in exponential time. Let $n = |x|$. First, we need to count the number of oracle circuits E on $(\log |T| \leq c'n^c)$-inputs with size at most $s(n)$. The logarithm of the number of oracle circuit of size at most $s(|x|)$ on $(c'n^c)$-inputs with t oracle functions is at most

$$O(s(n)\log(4+t) + s(|x|)\log(c'n^c)\log(s(|x|) + \log(c'n^c))).$$

Since $t \leq 2^{c'n^c}$ and s is polynomial in n, it is easy to see that the number of such circuits E is at most exponential. Second, one can check if an oracle circuit E satisfies $E^{\mathcal{O}}$ computes T in time polynomial in $(|E| + |T| + |\mathcal{O}|)$ and hence exponential in n. As a result, $L \in \mathsf{EXP}$, completing the proof. ◀

▶ **Theorem B.2.** *If* $\mathsf{NP} \leq_{tt}^{\mathsf{P}} \mathsf{MOCSP}$*, then* $\mathsf{EXP} \neq \mathsf{NP} \cap \mathsf{P/poly}$*. Consequently,* $\mathsf{EXP} \neq \mathsf{ZPP}$*.*

Proof. For contradiction, suppose $\mathsf{NP} \leq_{tt}^{\mathsf{P}} \mathsf{MOCSP}$ and $\mathsf{EXP} = \mathsf{NP} \cap \mathsf{P/poly}$. Then by Proposition B.1 $\mathsf{NEXP} \subseteq \mathsf{EXP} \subseteq \mathsf{NP}$ contradicting the nondeterministic time hierarchy theorem [32]. ◀

Equivalence of Systematic Linear Data Structures and Matrix Rigidity

Sivaramakrishnan Natarajan Ramamoorthy
University of Washington, Seattle, USA
sivanr@cs.washington.edu

Cyrus Rashtchian
University of California, San Diego, USA
crashtchian@eng.ucsd.edu

Abstract

Recently, Dvir, Golovnev, and Weinstein have shown that sufficiently strong lower bounds for linear data structures would imply new bounds for rigid matrices. However, their result utilizes an algorithm that requires an NP oracle, and hence, the rigid matrices are not explicit. In this work, we derive an equivalence between rigidity and the systematic linear model of data structures. For the n-dimensional inner product problem with m queries, we prove that lower bounds on the query time imply rigidity lower bounds for the query set itself. In particular, an explicit lower bound of $\omega\left(\frac{n}{r}\log m\right)$ for r redundant storage bits would yield better rigidity parameters than the best bounds due to Alon, Panigrahy, and Yekhanin. We also prove a converse result, showing that rigid matrices directly correspond to hard query sets for the systematic linear model. As an application, we prove that the set of vectors obtained from rank one binary matrices is rigid with parameters matching the known results for explicit sets. This implies that the vector-matrix-vector problem requires query time $\Omega(n^{3/2}/r)$ for redundancy $r \geq \sqrt{n}$ in the systematic linear model, improving a result of Chakraborty, Kamma, and Larsen. Finally, we prove a cell probe lower bound for the vector-matrix-vector problem in the high error regime, improving a result of Chattopadhyay, Koucký, Loff, and Mukhopadhyay.

2012 ACM Subject Classification Theory of computation → Cell probe models and lower bounds; Theory of computation → Circuit complexity

Keywords and phrases matrix rigidity, systematic linear data structures, cell probe model, communication complexity

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.35

Related Version A full version of the paper is available at `https://arxiv.org/abs/1910.11921`.

Funding *Sivaramakrishnan Natarajan Ramamoorthy*: Supported by the National Science Foundation under agreement CCF-1420268.

Acknowledgements We thank Paul Beame, Sajin Koroth, Pavel Hrubeš, Pavel Pudlák, Anup Rao, Makrand Sinha, Amir Yehudayoff and Sergey Yekhanin for useful discussions. Special thanks to Paul, Anup, Makrand and Amir for the encouragement to write up these results.

1 Introduction

A matrix is *rigid* if it is far in Hamming distance from low rank matrices; it is *explicit* if its entries are computable in polynomial time. A classic result of Valiant proves that explicit rigid matrices imply super-linear lower bounds for linear circuits [35], a major open problem in computational complexity [34, 37]. Implications of new lower bounds for communication complexity and other models are also known [24, 39]. Unfortunately, the current bounds for explicit matrices are very far from the required parameters [16, 33], and natural candidates (e.g., Fourier and Hadamard matrices) have been discovered to

be less rigid than desired [3, 12, 14]. This motivates alternative avenues for constructing rigid matrices. Recently, multiple connections between data structures and circuits have arisen [7, 11, 13, 38]. The premise of these results is that hard problems for these models may shed new light on rigid matrices and circuits. We take a similar angle, studying a generic linear problem for a model that resembles a depth-two circuit with linear gates.

Valiant's result concerns arithmetic circuits computing the linear map $v \mapsto Mv$ for a matrix M. In other words, the circuit computes the inner products between v and the rows of M. We study a related data structure problem, the *inner product problem*. The task is to preprocess an n-bit vector v to compute inner products $\langle q, v\rangle$ over $\mathbb{F}_2$ for queries $q \in Q$, where $Q \subseteq \mathbb{F}_2^n$ is the *query set*. This problem generalizes the prefix-sum problem [17] and vector-matrix-vector problem [8, 23].

We consider solving this problem using a restricted data structure model, the *systematic linear model*. This model may only store v verbatim along with a small number $r \ll n$ of *redundant* bits, which are the evaluations of r linear functions of v. To compute $\langle q, v\rangle$ for $q \in Q$, the query algorithm must output a linear function of these r bits along with any t bits of v, where t is the *query time*. We motivate this model with a simple upper bound. Suppose that the query set Q happens to be close to an r-dimensional subspace U. More precisely, assume that $d_H(q, U) \leq t$ for any $q \in Q$, where $d_H(q, U) := \min_{u \in U} d_H(q, u)$ and $d_H(q, u)$ denotes the Hamming distance. The systematic linear model will store r bits that correspond to inner products between v and some r vectors that form a basis for U. The query algorithm computes $\langle q, v\rangle$ by invoking the identity $\langle q, v\rangle = \langle u, v\rangle + \langle q - u, v\rangle$, using any vector $u \in U$ with $d_H(q, u) \leq t$. Indeed, the r precomputed bits suffice to determine $\langle u, v\rangle$, and at most t bits of v are needed to calculate $\langle q - u, v\rangle$.

We observe that rigidity exactly captures the complexity of the inner product problem in the above model. This connection uses a notion of rigid sets, defined by Alon, Panigrahy and Yekhanin [5]. Our result shows that an efficient algorithm exists in the above model if and only if the query set is not rigid in their sense. Conversely, it is possible to derive new rigidity lower bounds by proving lower bounds for the systematic linear model. A parameter of interest is the size of the rigid set, which corresponds to the number of queries in the inner product problem.

Dvir, Golovnev, and Weinstein also demonstrate a connection between rigidity and a different linear model, which is a restriction of the cell probe model [13]. This model stores $s \geq n$ linear functions, and the query algorithm outputs a linear function of t of these s bits. For the inner product problem with query set Q, they show that a lower bound for linear data structures leads to a semi-explicit rigid set. When $|Q| = m$, their result uses a $\mathsf{poly}(m)$ time algorithm that requires access to an NP oracle. Compared to their work, our connection preserves explicitness and offers a two-way equivalence via the systematic linear model. In particular, when $c'n \leq r \leq cn$ for constants $c', c < 1$, a lower bound of $t = \omega(\log m)$ in the systematic linear model implies that Q is rigid with better parameters than known results. Their work requires a lower bound of $t = \omega(\log m \log n)$ against the linear model, and the resulting set is not explicit. Our results also extend to show that linear data structure lower bounds lead to explicit rigid matrices. However, compared to the work of Dvir, Golovnev, and Weinstein, we require stronger lower bounds to achieve new rigidity parameters.

As an application of our framework, we provide new results for the vector-matrix-vector problem. The task is to preprocess a 0-1 matrix M to compute $u^\intercal Mv$ when given vectors u, v as the query. The boolean semiring version of this problem has received much recent attention due to connections to the online matrix-vector multiplication conjecture [18]. Moreover, this problem has motivated the study of data structures for a super polynomial number of queries,

even when the output is binary [8, 9]. Other prior work has either studied binary output problems with $\mathsf{poly}(n)$ queries (see e.g. [28, 30]) or achieved better lower bounds by looking at multi-output problems (see e.g. [10, 20]). In general, the vector-matrix-vector problem is a good testbed for proving better data structure lower bounds, because linear algebraic tools could provide new insights.

The $\mathbb{F}_2$ variant of this problem specializes the inner product problem because $u^\intercal Mv$ equals the inner product of $uv^\intercal$ and M (viewed as vectors). The query set consists of $\sqrt{n} \times \sqrt{n}$ matrices with rank one; its size m satisfies $\log m = \Theta(\sqrt{n})$. As another contribution, we lower bound the rigidity of this set, and consequently, we obtain a query time lower bound of $\Omega(\frac{n}{r} \log m) = \Omega(n^{3/2}/r)$ for the systematic linear model with redundancy $r \geq \sqrt{n}$. Any asymptotically better lower bounds for this problem (in the systematic linear model) would directly imply that this query set is rigid with better parameters than the currently known results for explicit matrices [4, 5].

As a final result, we prove a new cell probe lower bound for the vector-matrix-vector problem, without restrictions on the data structure. Our result improves the current best lower bound due to Chattopadhyay, Koucký, Loff, and Mukhopadhyay [9]. Our lower bound matches the limit of present techniques and achieves the current best time-space trade-off in terms of query set size.

1.1 Rigid sets, systematic linear model, and the inner product partial function

Throughout, let $m = m(n)$ and $t = t(n)$ and $r = r(n)$ denote positive integers, with $m \geq n \geq t, r$. Alon, Panigrahy and Yekhanin defined the following notion of a rigid set [5].

▶ **Definition** (Rigid Set). *A set $Q \subseteq \mathbb{F}_2^n$ is (r,t)-rigid if for every subspace $U \subseteq \mathbb{F}_2^n$ with dimension at most r, some vector $q \in Q$ has Hamming distance at least t from all vectors in U, that is, $d_H(q, U) \geq t$.*

We define (r', t')-rigid for non-integral r', t' to mean $(\lfloor r' \rfloor, \lceil t' \rceil)$-rigid. It will be convenient to equate a set Q with a matrix M_Q by arranging vectors in Q as rows in M_Q in any order. If Q is (r,t)-rigid and $|Q| = m$, then the corresponding matrix $M_Q \in \mathbb{F}_2^{m \times n}$ is rigid in the usual sense: for any rank r matrix A, some row in $(M_Q - A)$ contains at least t nonzero entries. Hence, we may refer to rigid sets and rigid rectangular matrices interchangeably. A matrix in $\mathbb{F}_2^{m \times n}$ (or a set of n-dimensional vectors) is *explicit* if every entry can be computed in $\mathsf{poly}(n)$ time.

A random $m \times n$ matrix with $m = \mathrm{poly}(n)$ will be $(\epsilon n, \delta n/\log n)$-rigid with high probability for some constants $\epsilon, \delta \in (0,1)$. The key challenge here is to construct explicit rigid matrices, because they provide circuit lower bounds for functions that can be described in polynomial time [35]. Alon, Panigrahy and Yekhanin [5] followed by Alon and Cohen [4] exhibit multiple examples of explicit $m \times n$ matrices that are (r,t)-rigid with

$$t \geq \min\left\{ \frac{cn}{r} \log \frac{m}{r},\ n \right\} \tag{1}$$

where $m \geq n$ and c is a constant. Note that when $r = \epsilon n$, the current best bound is $t = \Omega\left(\log \frac{m}{n}\right)$. For $m = poly(n)$, this amounts to $t = \Omega(\log n)$, exponentially far from the ideal bounds (i.e., matching random constructions). It is an important open problem to improve the dependence on m in Eq. (1) and to find other candidate sets that may be rigid with better parameters.

Our connection between rigidity and data structures arises via the inner product problem. The task is to preprocess a vector $v \in \mathbb{F}_2^n$ to compute inner products. The queries are specified by $Q \subseteq \mathbb{F}_2^n$, which is called the *query set*. The data structure must compute the inner product of v and any $q \in Q$, that is, $\langle q, v \rangle := \sum_{i=1}^{n} q[i] \cdot v[i] \mod 2$, where $q[i]$ denotes the i^{th} coordinate of q.

Consider the following model for solving this problem, known as a systematic linear data structure. During preprocessing, the data structure stores v along with the evaluations of r linear functions $\langle a_1, v \rangle, \ldots, \langle a_r, v \rangle$, where these inner products are single bits, and $a_1, \ldots, a_r$ denote vectors in $\mathbb{F}_2^n$. To compute the answer on query q, the data structure accesses these r bits in addition to any t entries of v. That is, the r linear functions are fixed, and the t bits from v may depend on q and the linear functions. Finally, the query algorithm must output a linear function of these r bits and the t entries of v. In this fashion it must be able to correctly compute $\langle q, v \rangle$ for all queries $q \in Q$. We note that a result of Jukna and Schnitger [19] shows that the $\{a_1, \ldots, a_r\}$ vectors do not depend on v without loss of generality. Letting $T(Q, r)$ denote the minimum value t of the best data structure for this problem (over worst-case v), we formalize the model as follows.

▶ **Definition** (Systematic Linear Model)**.** *Let $Q \subseteq \mathbb{F}_2^n$ be a set. Define $T(Q, r)$ to be the maximum over all $v \in \mathbb{F}_2^n$ of the minimum t sufficient to compute the inner product $\langle q, v \rangle$ for every $q \in Q$ when only allowed to output a linear function of r precomputed linear functions of v along with any t bits of v.*

Note that the model does not charge the query time for accessing the r precomputed bits, even if $t \ll r$. This coincides with the systematic model studied by Chakraborty, Kamma and Larsen [8].

1.2 Equivalence between rigidity and data structures

We prove that the rigidity of a set Q corresponds to the time complexity $T(Q, r)$ in the systematic linear data structure model. Some aspects of this result are implicit in prior work [19, 31], but no previous work seems to show this exact correspondence.

▶ **Theorem 1.** *A set $Q \subseteq \mathbb{F}_2^n$ is (r, t)-rigid if and only if $T(Q, r) \geq t$.*

Proof. We first prove that $T(Q, r) \geq t$ implies that Q is (r, t)-rigid. Assume for contradiction that there is an r-dimensional subspace U such that $d_H(q, U) < t$ for all $q \in Q$. Let $v \in \mathbb{F}_2^n$ be the input data. Store v along with the r bits $\langle b_1, v \rangle, \ldots, \langle b_r, v \rangle$, where $b_1, \ldots, b_r$ form a basis for U. For every $q \in Q$, there exists $u_q \in U$ such that $q - u_q$ has Hamming weight less than t. Using the r redundant bits, the algorithm on query q can compute $\langle u_q, v \rangle$ by writing u_q in terms of the stored basis vectors. Then, it computes $\langle q - u_q, v \rangle$ by accessing fewer than t coordinates of v. Since $\langle q, v \rangle = \langle u_q, v \rangle + \langle q - u_q, v \rangle$, we have that $T(Q, r) < t$, which is a contradiction.

We now prove that if Q is (r, t)-rigid, then $T(Q, r) \geq t$. Let $e_1, \ldots, e_n$ denote the standard basis, and let $k = T(Q, r)$ be the query time. We show that $k \geq t$. Consider a systematic linear data structure whose redundant bits are given by $\langle a_1, v \rangle, \ldots, \langle a_r, v \rangle$. Let U denote the span of $\{a_1, \ldots, a_r\}$. As Q is (r, t)-rigid, there exists $q^* \in Q$ with $d_H(q^*, U) \geq t$. When q^* is the query, assume that the query algorithm accesses the bits $v_{i_1}, \ldots, v_{i_k}$ for indices $i_1, \ldots, i_k$ to compute $\langle q^*, v \rangle$. Now, define U' to be the span of $\{a_1, \ldots, a_r, e_{i_1}, \ldots, e_{i_k}\}$. Observe that all points in U' are at distance at most k from U. Thus, $d_H(q^*, U) \leq d_H(q^*, U') + k$. We will show that $d_H(q^*, U') = 0$, which implies that $k \geq t$. We claim that if $d_H(q^*, U') \geq 1$, then the query algorithm makes an error. Since $d_H(q^*, U') \geq 1$, there exists a vector y with

$\langle y, q^* \rangle = 1$. Moreover, this vector can be taken to be orthogonal to U' so that $\langle y, x \rangle = 0$ for every $x \in U'$. In other words, for every $x \in U'$ we have $\langle y + v, x \rangle = \langle y, x \rangle + \langle v, x \rangle = \langle v, x \rangle$. Hence, the query algorithm sees the same values on input data $y + v$ and v because it only accesses the input via vectors in U', and we have $x \in U'$. Thus, the algorithm on query q^* must err either on input $y + v$ or v because $\langle q^*, y + v \rangle \neq \langle q^*, v \rangle$. ◀

1.3 Relationship to the cell probe model and other models

The systematic linear model specializes the *systematic model* [8, 17]. The latter model still stores the input data $x \in \mathbb{F}_2^n$ verbatim, and it also stores $r < n$ bits that can be precomputed from x, where these need not be linear functions of the input data. The query time is t if the query algorithm reads at most t bits from x to compute a query. The output can also be an arbitrary function of these t bits along with the r precomputed bits. The systematic linear model only makes sense for linear queries, whereas the systematic model applies to arbitrary query functions.

Yao's *cell probe model* is the most general data structure model [40]. On input data $x \in \mathbb{F}_2^n$, the data structure stores s cells, containing w bits that are arbitrary functions of x. Here, w is the *word size* and s is the *space*. The *query time* is t if the algorithm accesses at most t cells to answer any query about x from a set of m possible query functions. There is a rich collection of lower bounds for this model (see e.g. [2, 15, 20, 26, 27, 28, 29]). The best lower bounds known are of the form

$$t \geq \min \left\{ \frac{c \log \frac{m}{n}}{\log \frac{sw}{n}}, \ \frac{cn}{w} \right\}, \tag{2}$$

where $m \geq n$ is the number of queries and c is a constant. It is a long-standing problem to prove that $t = \omega(\log m)$ for any explicit problem, even in the linear space regime $s = O(n)$ and $w = O(1)$.

A special case of the cell probe model is the *linear model* [1, 13]. The latter model stores $s \geq n$ linear functions of x (implicitly $w = 1$ is fixed). The query time is t if the query algorithm reads at most t of these s bits to compute a query. The output is restricted to be a linear function of these t bits. A distinguishing aspect between linear and systematic linear is that in the latter model, the query algorithm is not charged for accessing the r precomputed bits. In Section 2, we compare the linear and systematic linear models in the context of rigidity and previous work [13].

Equivalences all the way down

We note that the systematic data structure model is identical to the common bits model defined by Valiant [36]. Corrigan-Gibbs and Kogan [11] demonstrate a relationship between the common bits model and a variant of the systematic model defined by Gal and Miltersen [17]. The common bits model is nothing but a certain depth two circuit, and the systematic linear model is simply the common bits model with the restriction that the common bits and output gates are linear functions [31]. Hence, in language of data structures, the linearization conjecture of Jukna and Schnitger posits that the systematic linear model is asymptotically as powerful as the systematic model for answering linear queries [19].

1.4 The vector-matrix-vector problem

We now define the vector-matrix-vector problem, which we call "the $\mathsf{u}^\intercal\mathsf{M}\mathsf{v}$ problem" for short. Let n be a perfect square. After preprocessing a matrix $M \in \mathbb{F}_2^{\sqrt{n} \times \sqrt{n}}$, the goal is to output the binary value $u^\intercal M v$ for vectors $u, v \in \mathbb{F}_2^{\sqrt{n}}$. It will be convenient to consider

a $\sqrt{n} \times \sqrt{n}$ matrix as an n-bit vector $\mathsf{vec}(M)$ by concatenating consecutive rows. More formally, let $x = \mathsf{vec}(M)$, and for $i \in \{1, 2, \ldots, n\}$, set $x[i] = M[a, b]$, where a and b satisfy $i = (a-1)\sqrt{n} + b$ and $a, b \in \{1, 2, \ldots, \sqrt{n}\}$. Then, $u^\intercal M v = \langle \mathsf{vec}(uv^\intercal), \mathsf{vec}(M) \rangle$. In this way we consider the $\mathsf{u^\intercal Mv}$ problem a special case of the inner product problem. The query set is the collection of rank one binary matrices. Let $\Upsilon \subseteq \mathbb{F}_2^n$ denote the set of vectors obtained from rank one binary matrices via $M \mapsto \mathsf{vec}(M)$, that is,

$$\Upsilon := \left\{ \mathsf{vec}(uv^\intercal) \;\middle|\; u, v \in \mathbb{F}_2^{\sqrt{n} \times \sqrt{n}} \right\} \subseteq \mathbb{F}_2^n. \tag{3}$$

This set has size $|\Upsilon| = 2^{2\sqrt{n}} - 2^{\sqrt{n}+1} + 1$.

A classic result of Artazarov, Dinic, Kronrod and Faradzev [6] provides a data structure with space $s = \mathsf{poly}(n)$, word size $w = O(\log n)$, and time $t = O(n/\log n)$. In fact, this algorithm operates in the linear cell probe model. It is a central open question to determine whether $t = \Omega(n)$ is necessary in linear space regime, that is, when $s = O(n)$ and $w = O(1)$.

The current best cell probe lower bound for the $\mathsf{u^\intercal Mv}$ problem is due to Chattopadhyay, Koucký, Loff, and Mukhopadhyay [9]. Moreover, their lower bound holds for a randomized model with high error. For constants c and c', they prove that if for every matrix M and every query $uv^\intercal$, the query algorithm correctly computes $u^\intercal M v$ with probability at least $\frac{1}{2} + \frac{1}{2^{c'\sqrt{n}}}$, then

$$t \geq \min\left\{ \frac{c\sqrt{n}}{\log \frac{sw}{\sqrt{n}}},\ \frac{cn}{w} \right\} \tag{4}$$

We know better lower bounds for the $\mathsf{u^\intercal Mv}$ problem in the systematic model. Chakraborty, Kamma, and Larsen [8] prove that t and r must satisfy $t \cdot r = \Omega(n^{3/2}/\log n)$ as long as $r \geq \sqrt{n}$. In the case of $r \leq \sqrt{n}$, they prove that $t = \Omega(n/\log n)$. As the systematic model subsumes the linear version of this model, combining their result with Theorem 1 implies that Υ is (r, t)-rigid with

$$t = \Omega\left(\frac{n^{3/2}}{\max\{\sqrt{n}, r\} \cdot \log n} \right). \tag{5}$$

1.5 New results on the rigidity of Υ and the cell probe complexity of the $\mathsf{u^\intercal Mv}$ problem

We lower bound the rigidity of Υ, defined in Eq. (3). This also implies a lower bound in the systematic linear model. The proof is inspired by a result of Alon, Panigrahy, and Yekahnin [5].

▶ **Theorem 2.** *Let $n \geq 1024$. The set $\Upsilon \subseteq \mathbb{F}_2^n$ of rank one matrices is (r, t)-rigid with $t \geq \frac{n^{3/2}}{128 \cdot \max\{\sqrt{n}, r\}}$.*

We improve the prior bound in Eq. (5) by an $\Omega(\log n)$ factor. For example, when $r \leq \sqrt{n}$, then $t = \Omega(n)$, and when $r = n/2$, then $t = \Omega\left(n^{1/2}\right)$. Theorem 2 matches Eq. (1), the current best bound for explicit rigid sets. We do not know whether there is a subspace U of linear dimension such that all elements of Υ are at distance $o(n)$ from U (unlike for some set rigidity results, where the bounds are tight). As a corollary of Theorem 1, we immediately get that

$$T(\Upsilon, r) \geq \frac{n^{3/2}}{128 \cdot \max\{\sqrt{n}, r\}}.$$

In other words, we prove a lower bound for the $\mathsf{u}^\intercal\mathsf{Mv}$ problem in the systematic linear model that improves the prior bound by an $\Omega(\log n)$ factor. The proof of Theorem 2 appears in Section 3.

We also prove a general cell probe lower bound for the $\mathsf{u}^\intercal\mathsf{Mv}$ problem in the high error regime. Our result improves the previous lower bound in Eq. (4). For example, in the linear space regime, when $s = O(n)$ and $w = O(1)$, we show that $t = \Omega(\sqrt{n})$ while the prior result gives only $t = \Omega(\sqrt{n}/\log n)$.

▶ **Theorem 3.** *Let $M \in \mathbb{F}_2^{\sqrt{n}\times\sqrt{n}}$ be a matrix. If a randomized data structure with space s, word size w, and time t correctly computes queries for the $\mathsf{u}^\intercal\mathsf{Mv}$ problem with probability at least $\frac{1}{2} + \frac{1}{2^{\sqrt{n}/64}}$, then*

$$t \geq \min\left\{\frac{c\sqrt{n}}{\log\frac{s\alpha}{n}}, \frac{cn}{\alpha}\right\}$$

where $0 < c \leq 1/36$ is a universal constant and $\alpha := 2(w + \log\frac{sw}{n})$.

The prior work utilizes a general lifting result for two-way communication complexity from parity decision trees [9]. To obtain the improved bound, we use a variant of the cell sampling technique [21, 27] combined with a reduction to a new lower bound on one-way communication (via discrepancy). The modifications over standard techniques are needed to handle the high error regime for a binary output problem. We note that a recent result of Larsen, Weinstein and Yu also uses one-way communication to prove lower bounds for binary output problems for dynamic data structures [22]. However, their method seems limited to only handling zero error query algorithms. The proof of Theorem 3 appears in Section 4. Specifically, see Lemma 14 in Section 4 for the variant of cell sampling and see Theorem 11 in Section 4.1 for the discrepancy argument.

2 Linear Data Structures and Rigidity

In this section, we relate linear data structures and rigidity. As linear data structures are a special case of the cell probe model, we may obtain rigidity lower bounds from strong enough static data structure lower bounds (when the queries are linear). We also compare with Dvir, Golovnev, and Weinstein, who exhibit a similar connection [13]. We first provide some notation.

▶ **Definition.** *Let $Q \subseteq \mathbb{F}_2^n$ be a set. Define $\mathsf{LT}(Q, s)$ to be the maximum over all $v \in \mathbb{F}_2^n$ of the minimum t sufficient to compute the inner product $\langle q, v\rangle$ for every $q \in Q$ when the query algorithm's output is a linear function of t bits chosen from the s precomputed linear functions of v.*

Table 1 provides a glimpse of our results on linear data structures along with a comparison to [13]. Recall that a set $Q \subseteq \mathbb{F}_2^n$ is explicit if each coordinate of an arbitrary element of the set can be computed in $\mathsf{poly}(n)$ time. The prior work shows that sufficiently strong lower bounds against linear data structures will imply *semi-explicit* rigid sets. A bit more formally, consider a data structure query set $Q \subseteq \mathbb{F}_2^n$ of size m for the inner product problem. They show the following: If $\mathsf{LT}(\mathsf{Q}, \mathsf{c}\cdot\mathsf{n}) \geq t$ for some constant c, then there is a $(n'/2, t/\log n)$-rigid set Q' of size at most m contained in $\mathbb{F}_2^{n'}$, where $n' \geq t$. However, the set Q' is only semi-explicit in that it is in P^{NP} – every element can be computed by a $\mathsf{poly}(m)$ time algorithm with access to an NP oracle.

Table 1 Comparison with [13, Theorem 7.1]: Let $Q \subseteq \mathbb{F}_2^n$ of size m be a query set, $c \geq 1$ and $\delta > 0$ be constants, and let $k = \mathsf{LT}(Q, 3n/2)$. The second column states the lower bound on $\mathsf{LT}(Q, 3n/2)$ that implies existence of rigid sets whose parameters are given in the third column. All rigid sets have size at most $\mathsf{poly}(m)$ and are contained in $\mathbb{F}_2^k$.

m vs n	$k = \mathsf{LT}(Q, 3n/2)$	Rigidity Bounds	Explicitness	Reference
$m = n^c$	$k = \omega\left(\sqrt{n \log n}\right)$	$(k/2, \omega(\log n))$-rigid	$\mathsf{poly}(n)$ time	This work
	$k = \omega\left(\log^2 n\right)$	$(k/2, \omega(\log n))$-rigid	$\mathsf{poly}(n)$ time + NP oracle calls	[13]
$m = n^c$	$k = \Omega\left(n^{(1+\delta)/2}\right)$	$\left(k/2, \Omega\left(n^\delta\right)\right)$-rigid	$\mathsf{poly}(n)$ time	This work
	$k = \Omega\left(n^\delta \log n\right)$	$\left(k/2, \Omega\left(n^\delta\right)\right)$-rigid	$\mathsf{poly}(n)$ time + NP oracle calls	[13]
$m = 2^{c\sqrt{n}}$	$k = \omega\left(n^{3/4}\right)$	$\left(k/2, \omega\left(\sqrt{n}\right)\right)$-rigid	$\mathsf{poly}(n)$ time	This work
	$k = \omega\left(\sqrt{n} \cdot \log n\right)$	$\left(k/2, \omega\left(\sqrt{n}\right)\right)$-rigid	$\mathsf{poly}\left(2^{\sqrt{n}}\right)$ time + NP oracle calls	[13]

We now summarize a few differences between our work and [13]. Our result proves that polynomial lower bounds on the query time imply the existence of an explicit rigid set, which is in contrast to semi-explicit sets obtained by [13]. On the other hand, explicitness comes with a cost; when $m = \mathsf{poly}(n)$, we need much stronger data structure lower bounds to produce explicit rigid sets. When $m \gg \mathsf{poly}(n)$, the algorithm of [13] takes $\mathsf{poly}(m)$ time with access to an NP oracle to compute an element of the semi-explicit rigid set. For problems such as the $\mathsf{u}^\intercal\mathsf{Mv}$ problem, this is super polynomial time. The rest of this section concerns proving the following theorem, which implies all of our results in Table 1.

▶ **Theorem 4.** *Let $k = \mathsf{LT}(Q, 3n/2)$ and let $Q \subseteq \mathbb{F}_2^n$ of size m be an explicit query set. There exists a set $Q' \subseteq \mathbb{F}_2^k$ with size at most $m \cdot \lceil \frac{n}{k} \rceil$, whose elements can be computed in $\mathsf{poly}(n)$ time. Moreover, if $k \geq 2\sqrt{n}$, then Q' is explicit and $\left(\frac{k}{2}, \frac{k^2}{4n}\right)$-rigid.*

Note that for every $s \geq 3n/2$, we have that $\mathsf{LT}(Q, 3n/2) \geq \mathsf{LT}(Q, s)$. Hence, a sufficiently strong lower bound on $\mathsf{LT}(Q, s)$ for any $s \geq 3n/2$ will imply a rigidity lower bound. The following corollary shows the consequence of Theorem 4 for specific values of k.

▶ **Corollary 5.** *Let $k = \mathsf{LT}(Q, 3n/2)$ and let $Q \subseteq \mathbb{F}_2^n$ of size m be an explicit query set. There exists a set $Q' \subseteq \mathbb{F}_2^k$ with size at most $m \cdot \lceil \frac{n}{k} \rceil$, whose elements can be computed in $\mathsf{poly}(n)$ time. Moreover,*

(a) *If $k = \omega\left(\sqrt{n \log m}\right)$, then Q' is explicit and $(k/2, \omega(\log m))$-rigid.*

(b) *If $k = \Omega\left(n^{(1+\delta)/2}\right)$ for some $\delta > 0$, then Q' is explicit and $\left(k/2, \Omega\left(n^\delta\right)\right)$-rigid.*

Corollary 5(a) explains the first and last rows in Table 1, and Corollary 5(b) explains the middle row. Using Corollary 5(a) applied to Υ with $m = 2^{2\sqrt{n}} - 2^{\sqrt{n}+1} + 1$, we obtain that a lower bound of $\mathsf{LT}(\Upsilon, 3n/2) \geq \omega(n^{3/4})$ would imply the existence of an explicit set $Q' \subseteq \mathbb{F}_2^k$ of size $2^{O(\sqrt{n})}$ that is $(k/2, \omega(\sqrt{n}))$-rigid. We note that it is an open question to prove $\mathsf{LT}(\Upsilon, 3n/2) \geq \omega(\sqrt{n})$.

2.1 Proof of Theorem 4

We already know the equivalence between systematic linear data structures and rigidity (from Theorem 1). Therefore, it is sufficient to design a linear data structure from a systematic linear data structure to relate the former with rigidity.

▶ **Proposition 6.** *Let $Q \subseteq \mathbb{F}_2^n$ be a query set. If $T(Q,r) \leq t$, then $\mathsf{LT}(Q, n+r) \leq t+r$.*

Proof. Let $v \in \mathbb{F}_2^n$ be the input data and $\langle a_1, v\rangle, \ldots, \langle a_r, v\rangle$ be the r redundant bits stored by the systematic linear data structure. We now describe a linear data structure for Q with space $n+r$ and query time $t+r$. The data structure stores $\langle a_1, v\rangle, \ldots, \langle a_r, v\rangle, \langle e_1, v\rangle, \ldots, \langle e_n, v\rangle$, where $e_1, \ldots, e_n$ are the standard basis vectors. The query algorithm on $q \in Q$ first accesses $\langle a_1, v\rangle, \ldots, \langle a_r, v\rangle$ and then simulates the query algorithm of the systematic linear data structure on q. Since the systematic linear data structure accesses at most t bits from $\langle e_1, v\rangle, \ldots, \langle e_n, v\rangle$, we can conclude that the query time is at most $t+r$. ◀

We prove that if a set contained in a n-dimensional space is (r,t)-rigid, then there is another $(r, tr/n)$-rigid set which is contained in a $2r$-dimensional space.

▶ **Lemma 7.** *Let r, n be positive integers. If $S \subseteq \mathbb{F}_2^n$ is (r,t)-rigid of size m, then there is a set $S' \subseteq \mathbb{F}_2^{2r}$ of size at most $m \cdot \lceil \frac{n}{2r} \rceil$ that is $(r, tr/n)$-rigid. Moreover, if S is explicit, then each element of S' can be computed in $\mathsf{poly}(n)$ time.*

Proof. Let $k = \lfloor \frac{n}{2r} \rfloor$ and define $S_1, \ldots, S_k \subseteq \mathbb{F}_2^{2r}$ by

$$S_i = \{(s[2r \cdot (i-1)+1], \ldots, s[2r \cdot i]) \mid s \in S\}$$

for each $i \in \{1, 2, \ldots, k\}$. Additionally, if $n/2r$ is not an integer, then define

$$S_{k+1} = \{(s[2r \cdot k + 1], \ldots, s[n], 0, \ldots, 0) \mid s \in S\} \subseteq \mathbb{F}_2^{2r};$$

otherwise set $S_{k+1} = \emptyset$. Define $S' = \bigcup_{i=1}^{k+1} S_i$. We claim that S' is $(r, tr/n)$-rigid. Indeed, for the sake of contradiction assume that there is a subspace V in $\mathbb{F}_2^{2r}$ of dimension r such that all points in S' are at a distance less than tr/n from V. Consider the subspace $\{(v, v, \ldots, v) \mid v \in V\} \subseteq \mathbb{F}_2^{2r \cdot (k+1)}$ and project it to the first n coordinates. Call this subspace V', which has dimension r. Now, the distance of each point in S from V' is less than $\frac{tr}{n} \cdot \lceil \frac{n}{2r} \rceil < t$, which is a contradiction.

Regarding the explicitness of S', it is clear that all coordinates of an element of S' correspond to some coordinate of a specific element of S. Since S is explicit, we can infer that each element of S' can be computed in $\mathsf{poly}(n)$. ◀

Proof of Theorem 4. We know that $\mathsf{LT}(Q, 3n/2) = k$ and $k \leq n$, Proposition 6 implies that $T(Q, k/2) \geq k/2$. Therefore by Theorem 1, we can conclude that Q is $(k/2, k/2)$-rigid. Lemma 7 implies that there exists a set Q' that is $\left(\frac{k}{2}, \frac{k^2}{4n}\right)$-rigid and the size of Q' is at most $m \cdot \lceil \frac{n}{k} \rceil$. Moreover, every element of Q' can be computed in time $\mathsf{poly}(n)$. Since $k/2 \geq \sqrt{n}$, we can conclude that Q' is explicit. ◀

3 Rigidity Lower Bounds for the Set of Rank One Matrices

Before proving Theorem 2, we present preliminaries. Recall two standard binomial estimates:

▶ **Proposition 8.** *For integers $0 \leq k \leq \ell$,*

1. $\log \binom{\ell}{k} \leq k \cdot \log \frac{e\ell}{k}$.
2. *if $k \leq \ell/16$, then $\sum_{i=0}^{k} \binom{\ell}{i} \leq 2^{\ell/4}$.*

We will need a useful property about the distance of a point from a subspace.

▶ **Lemma 9.** *Let $V \subseteq \mathbb{F}_2^\ell$ be a subspace. For $u_1, u_2 \in \mathbb{F}_2^\ell$, $d_H(u_1+u_2, V) \leq d_H(u_1, V) + d_H(u_2, V)$.*

Proof. Let $u_1', u_2' \in V$ be the points in V closest to u_1 and u_2 respectively. Since $u_1' + u_2' \in V$, we have $d_H(u_1 + u_2, V) \leq d_H(u_1 + u_2, u_1' + u_2')$. Note that $d_H(u_1 + u_2, u_1' + u_2')$ is the number of ones in $u_1 + u_2 + u_1' + u_2'$, which is at most the sum of the number of ones in $u_1 + u_1'$ and $u_2 + u_2'$. Therefore,

$$d_H(u_1 + u_2, u_1' + u_2') \leq d_H(u_1, u_1') + d_H(u_2, u_2') = d_H(u_1, V) + d_H(u_2, V). \quad \blacktriangleleft$$

A simple counting argument establishes the existence of a point that is far away in Hamming distance from a collection of large sized sets.

▶ **Lemma 10.** *Let $V_1, \ldots, V_k$ be subsets of $\mathbb{F}_2^\ell$, each of size at most $2^{\ell/2}$. If $k < 2^{\ell/4}$, then there is a vector $v \in \mathbb{F}_2^\ell$ such that the Hamming distance of v from each V_i is at least $\ell/16$.*

Proof. For every $i \in [k]$, define $\mathsf{B}(V_i, \ell/16) = \left|\left\{v \in \mathbb{F}_2^\ell \;\middle|\; d_H(v, V_i) < \ell/16\right\}\right|$. For any $u \in V_i$, the number of vectors in $\mathbb{F}_2^\ell$ at a distance less than $\ell/16$ from u is at most $\sum_{j=0}^{\ell/16} \binom{\ell}{j} \leq 2^{\ell/4}$, where the inequality follows from Proposition 8. Hence $\mathsf{B}(V_i, \ell/16) \leq |V_i| \cdot 2^{\ell/4} = 2^{3\ell/4}$. Since

$$\sum_{i=1}^{k} \mathsf{B}(V_i, \ell/16) \leq k \cdot 2^{3\ell/4} < 2^\ell,$$

there is a $v \in \mathbb{F}_2^\ell$ such that $d_H(v, V_i) \geq \ell/16$ for every $i \in [k]$. $\blacktriangleleft$

3.1 Proof of Theorem 2

Let V be any r'-dimensional subspace of $\mathbb{F}_2^n$, where $r' \geq r$ is the smallest positive integer divisible by $\sqrt{n}$. We first define the inverse of $\mathsf{vec}(\cdot)$. For every $v \in \mathbb{F}_2^n$, define $\mathsf{mat}(v)$ to be the matrix obtained by splitting v into $\sqrt{n}$ length consecutive blocks and stacking each of these blocks to form a $\sqrt{n} \times \sqrt{n}$ matrix. Formally, $\mathsf{mat}(v) \in \mathbb{F}_2^{\sqrt{n} \times \sqrt{n}}$ is such that $\mathsf{mat}(v)[a, b] = v[(a-1)\sqrt{n} + b]$ for every $a, b \in [\sqrt{n}]$. Note that $\mathsf{vec}(\mathsf{mat}(v)) = v$.

We provide a brief outline of the proof of Theorem 2. The first step of the proof is to produce a vector in v that is at a distance of $\Omega(n)$ from V and $\mathsf{mat}(v)$ is low rank. The rank being low is helpful as we can express $\mathsf{mat}(v)$ as the sum of a small number of rank one matrices. Lemma 9 will then imply the existence of a rank one matrix that is far away from V. If we only cared about the existence of a vector that is far away from V, Lemma 10 would suffice. To ensure that simultaneously the rank is small, we first project V on to $n/2r'$ coordinates indexed by consecutive blocks each of length $2r'$. Then we find a vector $v' \in \mathbb{F}_2^{2r'}$ that is far away from all the projections, which is still guaranteed by Lemma 10. Concatenating v' with itself $2r'$ times has the property that its corresponding matrix is low rank.

Let $k = \max\left\{\left\lfloor \frac{n}{2r'} \right\rfloor, 1\right\}$. The goal is to find a $v \in \mathbb{F}_2^n$ such that $d_H(v, V) \geq k \cdot r'/8$ and the rank of $\mathsf{mat}(v)$ is at most $2r'/\sqrt{n}$. If $\left\lfloor \frac{n}{2r'} \right\rfloor \geq 1$, then define $S_1, \ldots, S_k$ such that

$$S_i = \{(i-1) \cdot 2r' + 1, \ldots, i \cdot 2r'\}$$

for $i \in [k]$; otherwise, define $S_1 = [n]$. By definition, the dimension of V_{S_i} is at most $r' = |S_i|/2$, for every $i \in [k]$. Since $r' \geq \sqrt{n}$ and $n \geq 1024$, we can infer that $k \leq 2r'$ and $2r' < 2^{r'/2}$. Lemma 10 implies the existence of a $v' \in \mathbb{F}_2^{2r'}$ with the property that $d_H(v', V_{S_i}) \geq r'/8$ for every $i \in [k]$. Now define $v \in \mathbb{F}_2^n$ by

$$v[i] = \begin{cases} v'[i \bmod 2r'] & \text{if } i \leq k \cdot 2r' \text{ and } i \bmod 2r' \neq 0, \\ v'[2r'] & \text{if } i \leq k \cdot 2r' \text{ and } i \bmod 2r' = 0, \\ 0 & \text{if } i > 2kr', \end{cases}$$

for all $i \in [n]$. In words, v is the length n vector that is the concatenation of k copies of v' along with the vector of zeros of length $n - 2kr'$. By the choice of v, we get that,

$$d_H(v, V) \geq \sum_{i=1}^{k} d_H(v, V_{S_i}) \geq k \cdot r'/8.$$

Moreover, the rank of $\mathsf{mat}(v)$ is at most $\frac{2r'}{\sqrt{n}}$. Therefore we can express

$$\mathsf{mat}(v) = \sum_{i=1}^{2r'/\sqrt{n}} a_i b_i^{\intercal},$$

for some $a_1, b_1, \ldots, a_{\frac{2r'}{\sqrt{n}}}, b_{\frac{2r'}{\sqrt{n}}} \in \mathbb{F}_2^{\sqrt{n}}$. By Lemma 9, we know that

$$d_H(v, V) \leq \sum_{i=1}^{2r'/\sqrt{n}} d_H(\mathsf{vec}(a_i b_i^{\intercal}), V).$$

Hence there exists an $i \in \left[\frac{2r'}{\sqrt{n}}\right]$ such that $d_H(\mathsf{vec}(a_i b_i^{\intercal}), V) \geq \frac{\sqrt{n} \cdot k}{16} \geq \frac{n^{3/2}}{64r'}$. The observation that $r' \leq 2\max\{\sqrt{n}, r\}$ completes the proof of the theorem.

▶ Remark (Extension to strong rigidity). Alon and Cohen [4] defined the notion of *strong* rigidity; a set $Q \subseteq \mathbb{F}_2^n$ is (r,t)-strongly rigid if for every subspace of $\mathbb{F}_2^n$ of dimension at most r, the average distance of all the points to the subspace is at least t. For strong rigidity, the best lower bounds known for explicit sets are also of the form given in Eq. (1). We can show that Υ is (r,t)-strongly rigid with $t \geq \Omega\left(\frac{n^{3/2}}{\max\{\sqrt{n}, r\}}\right)$, matching the best strong rigidity bounds known for explicit sets. We sketch the proof here. We know that

$$u^{\intercal}Mv + (u + e_i)^{\intercal}Mv + u^{\intercal}M(v + e_j) + (u + e_i)^{\intercal}M(v + e_j) = b_i^{\intercal}Mb_j,$$

where $u, v \in \mathbb{F}_2^{\sqrt{n}}$ and $e_1, \ldots, e_{\sqrt{n}}$ are standard basis vectors in $\mathbb{F}_2^{\sqrt{n}}$. This fact can be used to prove that the matrix M_{Υ} corresponding to the set Υ is a generator matrix of a 4-query locally decodable code that tolerates a constant fraction of errors. A result of [13, Theorem 6] shows that Theorem 2 and the locally decodable code property of M_{Υ} imply the strong rigidity of Υ.

4 Cell Probe Lower Bounds for the $u^{\intercal}Mv$ Problem

We know of two techniques for proving cell probe lower bounds matching Eq. (2). One is a technique of Pătraşcu and Thorup [30] who combined the communication complexity simulation of Miltersen [25] with multiple queries on the same input data. The other is the technique we use, which is based on cell sampling. Cell sampling typically requires one to work with large sized fields in order to handle errors. This large field size is needed to encode a large subset of the correctly computed queries using a small subset of cells. Here, we avoid encoding the subset of queries by a reduction to one-way communication complexity.

Proof outline for Theorem 3

By Yao's min-max principle, it suffices to prove a lower bound on deterministic data structures. The hard distribution on the input data M and query $uv^{\intercal}$ we use is given by sampling $M, (u, v)$ uniformly and independently at random. We prove the theorem by contradiction,

and we start by assuming that the query time is small. The proof is carried out in three steps. First, modify the data structure so that for every M, the fraction of queries correctly computed is at least $1/2$. This modification only increases the query time and space by 1, and it can only increase the overall probability of the query algorithm being correct. Second, for a given M, we use a variant of cell sampling (see Lemma 14) to obtain a small subset of cells S and a large subset of queries Q' such that all queries in Q' can be computed by only accessing cells in S. Moreover,

$$\Pr\left[\text{query algorithm correctly computes } u^\intercal Mv \mid uv^\intercal \in Q'\right]$$
$$\approx \Pr\left[\text{query algorithm correctly computes } u^\intercal Mv\right].$$

Third, we show that S can be used to design an efficient protocol for the following communication game: Alice's input is M and Bob's input is $uv^\intercal$, and the goal is for Bob to correctly compute $u^\intercal Mv$ on a sufficiently good fraction of the inputs after receiving a message from Alice.

We now describe the protocol (see Figure 1). Alice sends the locations and contents of S. This ensures that Bob correctly computes $u^\intercal Mv$ on a large fraction of queries in Q'. Alice also communicates the majority value of $u^\intercal Mv$ for $uv^\intercal \notin Q'$ so that Bob is correct on half of his possible inputs that are not in Q'. Overall, Bob's output is correct on a sufficiently good fraction of all $M, (u, v)$. Since we have assumed that the query time is small, we are able to show that Alice's communication is small. This contradicts a lower bound on the communication complexity of this game. More precisely, we prove the following lower bound.

▶ **Theorem 11.** *Suppose that Alice gets a uniformly random matrix $M \in \mathbb{F}_2^{\sqrt{n}\times\sqrt{n}}$ as input and Bob receives a uniform pair $(u, v) \in \mathbb{F}_2^{\sqrt{n}} \times \mathbb{F}_2^{\sqrt{n}}$ as input. If Alice sends a deterministic message to Bob and Bob computes $u^\intercal Mv$ such that*

$$\Pr_{M,u,v}[\textit{Bob computes } u^\intercal Mv \textit{ correctly}] \geq \frac{1}{2} + \frac{1}{2^{\sqrt{n}/8}},$$

then Alice must communicate at least $n/10$ bits.

Previously, in the *randomized* two-way communication setting, Chattopadhyay, Koucký, Loff, and Mukhopadhyay [9] proved a lower bound for the game given in Theorem 11. Their lower bound implies the lower bound in Theorem 11 against randomized protocols. We need a lower bound against *deterministic* protocols under the *uniform distribution* on the inputs, and we cannot use their theorem as a black-box. We provide a straightforward proof of Theorem 11 in Section 4.1 by using the *discrepancy method* on a related communication game (resembling a direct sum, where Bob receives multiple inputs).

Preliminaries

Before presenting the proof of Theorem 3, we define some notation. For a real valued function f with a finite domain $X \times Y$, $\mathbb{E}_{x,y}[f(x,y)] = \frac{1}{|X|\cdot|Y|} \cdot \sum_{x\in X, y\in Y} f(x,y)$. Similarly, for $X' \subseteq X$, $\mathbb{E}_{x,y}[f(x,y) \mid x \in X'] = \frac{1}{|X'|\cdot|Y|} \cdot \sum_{x\in X', y\in Y} f(x,y)$. An argument in the proof of Theorem 3 requires an upper bound on the number of bits to encode the contents and locations of a subset of the cells, which is given by the following proposition.

▶ **Proposition 12.** *Let S be a subset of the cells of a data structure with word length w and size s. Then, the contents and locations of S can be encoded in $|S| \cdot w + |S| \cdot \log \frac{es}{|S|}$ bits.*

Proof. Since each cell stores w bits, the number of bits to encode the contents is $|S| \cdot w$. Since the total number of cells is s, the locations can be encoded in $\log \binom{s}{|S|} \leq |S| \cdot \log \frac{es}{|S|}$ bits, where the inequality followed from Proposition 8. ◀

4.1 Proof of Theorem 11

We start by discussing a slightly related problem, whose solution will lead to the proof strategy used here. Let $M \in \mathbb{F}_2^{\sqrt{n}\times\sqrt{n}}$, $v \in \mathbb{F}_2^{\sqrt{n}}$, and $e_1, \ldots, e_{\sqrt{n}}$ be the standard basis vectors in $\mathbb{F}_2^{\sqrt{n}}$. Consider the communication game in which Alice gets as input a uniform random M and Bob gets as input a uniform random pair (e_i, v). Bob's goal is to compute $e_i^\intercal Mv$ after receiving a message from Alice. To understand how much Alice has to communicate, it is natural to look at the problem where Bob computes $\sum_{i=1}^{\sqrt{n}} e_i^\intercal Mv_i$, where $v_1, \ldots, v_{\sqrt{n}} \in F_2^{\sqrt{n}}$. Now observe that this sum is the same as the trace of $\left(\sum_{i=1}^{\sqrt{n}} e_i v^\intercal\right) M$, which in turn is the inner product between two n-bit vectors. The communication complexity of the inner product between two n-bit vectors is very well understood. Therefore, the lower bound on the amount of communication to compute the inner product between two n-bit vectors translates to a lower bound to the problem of computing $e_i^\intercal Mv$. This strategy applied to our setting gives us the following lower bound, which will be used to prove Theorem 11. Our presentation closely follows [32, Chapter 5].

▶ **Lemma 13.** *Let $0 < \epsilon \leq 1/2$ and let k be an integer. Alice gets a uniformly random $M \in \mathbb{F}_2^{\sqrt{n}\times\sqrt{n}}$ as input and Bob receives k uniform pairs $(u_1, v_1), \ldots, (u_k, v_k) \in \mathbb{F}_2^{\sqrt{n}} \times \mathbb{F}_2^{\sqrt{n}}$ as input. Assume that Alice communicates a deterministic message to Bob, and Bob computes $\sum_{i=1}^k u_i^\intercal Mv_i$ with*

$$\Pr_{M,u_1,v_1,\ldots,u_k,v_k}\left[\textit{Bob computes } \sum_{i=1}^{k} u_i^\intercal Mv_i \textit{ correctly}\right] \geq \frac{1}{2} + \epsilon.$$

If $k \leq \sqrt{n}$, then Alice must communicate at least $9k\sqrt{n}/40 - \log(1/\epsilon)$ bits.

Proof. We use the discrepancy method to prove the communication lower bound. This requires upper bounding the discrepancy of the *communication matrix* under a given distribution. Let R be a rectangle of the communication matrix, which is defined by indicator functions A_R and B_R such that $(M, ((u_1, v_1), \ldots, (u_k, v_k)))$ is in the rectangle R if and only if $A_R(M) = 1$ and $B_R((u_1, v_1), \ldots, (u_k, v_k)) = 1$.

Consider the distribution in which $M, (u_1, v_1), \ldots, (u_k, v_k)$ are chosen at random uniformly and independently. We upper bound the discrepancy under this distribution. In other words, we claim that for every rectangle R,

$$\mathop{\mathbb{E}}_{M,(u_1,v_1),\ldots,(u_k,v_k)}\left[A_R(M)B_R((u_1, v_1), \ldots, (u_k, v_k))(-1)^{\sum_{i=1}^k u_i^\intercal Mv_i}\right] \leq 2 \cdot 2^{-9k\sqrt{n}/40}. \tag{6}$$

By a standard relation in communication complexity between the number of bits communicated and discrepancy of rectangles (see [32, Chapter 5, Theorem 5.2]), Eq. (6) implies that Alice must communicate at least $9k\sqrt{n}/40 - \log(1/\epsilon)$ bits. We are left with the proof of Eq. (6).

$$\begin{aligned}
&\left(\mathop{\mathbb{E}}_{(u_1,v_1),\ldots,(u_k,v_k)}\left[B_R((u_1, v_1), \ldots, (u_k, v_k)) \mathop{\mathbb{E}}_{M}\left[A_R(M)(-1)^{\sum_{i=1}^k u_i^\intercal Mv_i}\right]\right]\right)^2 \\
&\leq \mathop{\mathbb{E}}_{(u_1,v_1),\ldots,(u_k,v_k)}\left[B_R((u_1, v_1), \ldots, (u_k, v_k))^2 \left(\mathop{\mathbb{E}}_{M}\left[A_R(M)(-1)^{\sum_{i=1}^k u_i^\intercal Mv_i}\right]\right)^2\right] \\
&\leq \mathop{\mathbb{E}}_{(u_1,v_1),\ldots,(u_k,v_k)}\left[\left(\mathop{\mathbb{E}}_{M}\left[A_R(M)(-1)^{\sum_{i=1}^k u_i^\intercal Mv_i}\right]\right)^2\right].
\end{aligned}$$

where the first inequality follows from convexity and the second one follows from the fact that $B_R((u_1,v_1),\ldots,(u_k,v_k)) \leq 1$. Now

$$\begin{aligned}
&\mathop{\mathbb{E}}_{(u_1,v_1),\ldots,(u_k,v_k)}\left[\left(\mathop{\mathbb{E}}_{M}\left[A_R(M)(-1)^{\sum_{i=1}^k u_i^\intercal M v_i}\right]\right)^2\right] \\
&\leq \mathop{\mathbb{E}}_{(u_1,v_1),\ldots,(u_k,v_k),M,M'}\left[A_R(M)A_R(M')(-1)^{\sum_{i=1}^k u_i^\intercal M v_i+\sum_{i=1}^k u_i^\intercal M' v_i}\right] \\
&\leq \mathop{\mathbb{E}}_{M,M'}\left[\left|\mathop{\mathbb{E}}_{(u_1,v_1),\ldots,(u_k,v_k)}\left[(-1)^{\sum_{i=1}^k u_i^\intercal (M+M') v_i}\right]\right|\right] \\
&= \mathop{\mathbb{E}}_{M}\left[\left|\mathop{\mathbb{E}}_{(u,v)}\left[(-1)^{u^\intercal M v}\right]\right|^k\right],
\end{aligned}$$

where the last equality follows from the fact that $(u_1,v_1),\ldots,(u_k,v_k)$ are chosen independent of each other and $M+M'$ is uniformly distributed as M and M' are chosen uniformly and independently at random. We are left with upper bounding $\mathbb{E}_M\left[\left|\mathbb{E}_{(u,v)}\left[(-1)^{u^\intercal M v}\right]\right|^k\right]$. First note that if M has rank r, then $\mathbb{E}_{u,v}\left[(-1)^{u^\intercal M v}\right] = 2^{-r}$. This is because,

$$\begin{aligned}
\mathop{\mathbb{E}}_{u,v}\left[(-1)^{u^\intercal M v}\right] &= \frac{1}{2^{\sqrt{n}}}\cdot\left(\sum_{v:Mv=0} 1\right) + \frac{1}{2^{\sqrt{n}}}\cdot\left(\sum_{v:Mv\neq 0}\mathop{\mathbb{E}}_{u}\left[(-1)^{u^\intercal M v}\right]\right) \\
&= \frac{2^{\sqrt{n}-r}}{2^{\sqrt{n}}} + 0 = 2^{-r}.
\end{aligned}$$

In addition, $\Pr_M\left[\text{rank of } M \leq 9\sqrt{n}/20\right] \leq 2^{-9n/10}$. Indeed, the number of matrices in $\mathbb{F}_2^{\sqrt{n}\times\sqrt{n}}$ of rank at most k is at most

$$\binom{2^{\sqrt{n}}}{k}\cdot\left(2^k\right)^{\sqrt{n}} \leq 2^{2k\sqrt{n}}.$$

Therefore, using the law of total expectation, we have that

$$\mathop{\mathbb{E}}_{M}\left[\left|\mathop{\mathbb{E}}_{(u,v)}\left[(-1)^{u^\intercal M v}\right]\right|^k\right] \leq \Pr_M\left[\text{rank of } M \leq 9\sqrt{n}/20\right] + 2^{-9k\sqrt{n}/20} \leq 2\cdot 2^{-9k\sqrt{n}/20},$$

where the last inequality followed from the fact that $k \leq \sqrt{n}$. ◀

Proof of Theorem 11. Let c be the number of bits communicated by Alice. We show that $c > n/10$. Define $Z_M(u,v) = 1$ if Bob correctly computes $u^\intercal M v$ and $Z_M(u,v) = -1$ otherwise. By the definition of $Z_M(u,v)$ and the lower bound on the probability of Bob's computation being correct, we have that $\mathbb{E}_{M,u,v}\left[Z_M(u,v)\right] \geq 2\cdot 2^{-\sqrt{n}/8}$.

We note that it is without loss of generality that $\mathbb{E}_{u,v}\left[Z_M(u,v)\right] \geq 0$ for every $M \in \mathbb{F}_2^{\sqrt{n}\times\sqrt{n}}$. This is because Alice on input M can send an extra bit indicating whether $\mathbb{E}_{u,v}\left[Z_M(u,v)\right] < 0$ and Bob will flip his output accordingly.

We now use the given protocol to design a protocol for a new communication game: Suppose that Alice gets a uniformly random $M \in \mathbb{F}_2^{\sqrt{n}\times\sqrt{n}}$ as input and Bob receives $\sqrt{n}$ uniform pairs $(u_1,v_1),\ldots,(u_{\sqrt{n}},v_{\sqrt{n}}) \in \mathbb{F}_2^{\sqrt{n}}\times\mathbb{F}_2^{\sqrt{n}}$ as input. We will use Lemma 13 with $k = \sqrt{n}$ to obtain the desired lower bound on c.

We claim that there is a communication protocol in which Alice communicates c bits and Bob computes $\sum_{i=1}^{\sqrt{n}} u_i^\intercal M v_i$ such that

$$\Pr_{M,u_1,v_1,\ldots,u_{\sqrt{n}},v_{\sqrt{n}}}\left[\text{Bob computes } \sum_{i=1}^{\sqrt{n}} u_i^\intercal M v_i \text{ correctly}\right] \geq \frac{1}{2} + \frac{2^{\sqrt{n}-1}}{2^{n/8}}. \tag{7}$$

Alice's message is same as before, and Bob computes each of $u_i^\intercal M v_i$ separately and outputs the sum modulo 2. We now prove Eq. (7). For a fixed M, the probability that Bob correctly computes $\sum_{i=1}^{\sqrt{n}} u_i^\intercal M v_i$ is $\frac{1}{2}\left(1 + \left(\mathbb{E}_{u,v}\left[Z_M(u,v)\right]\right)^{\sqrt{n}}\right)$. Therefore the overall probability that Bob correctly computes $\sum_{i=1}^{\sqrt{n}} u_i^\intercal M v_i$ is at least

$$\frac{1}{2}\left(1 + \frac{\sum_M \left(\mathbb{E}_{u,v}\left[Z_M(u,v)\right]\right)^{\sqrt{n}}}{2^n}\right) \geq \frac{1}{2}\left(1 + \left(\mathop{\mathbb{E}}_{M,u,v}\left[Z_M(u,v)\right]\right)^{\sqrt{n}}\right) \geq \frac{1}{2} + \frac{2^{\sqrt{n}-1}}{2^{n/8}},$$

where the first inequality follows from convexity of the function $f(x) = x^k$ with $k = \sqrt{n}$. Applying Lemma 13 with $k = \sqrt{n}$ implies that $c > n/10$, which completes the proof of the theorem. ◀

4.2 Proof of Theorem 3

If $n < 36$, the theorem is vacuously true as $c \leq 1/36$. For the rest of the argument we will assume that $n \geq 36$. We prove a lower bound on the query time t against deterministic data structures with space s and word size w. Suppose that the input data M and query $uv^\intercal$ is given by choosing M, u, v uniformly and independently at random, and the query algorithm is guaranteed to satisfy

$$\Pr_{M,u,v}\left[\text{query algorithm computes } u^\intercal M v \text{ correctly}\right] \geq \frac{1}{2} + 2^{-\sqrt{n}/16}.$$

By Yao's minmax principle, this will imply a lower bound on randomized data structures.

We first modify the given data structure to ensure that for every $M \in \mathbb{F}_2^{\sqrt{n}\times\sqrt{n}}$, the probability that $u^\intercal M v$ is correctly computed is at least $1/2$. Assume that we have a data structure with query time t', space s' and word size w. The modified data structure stores an extra bit indicating whether the $\Pr_{u,v}\left[\text{query algorithm computes } u^\intercal M v \text{ correctly}\right]$ is less than $1/2$ or not for a given M. The query algorithm is the same as before, but accesses this extra bit to flip the output if it is set to 1. Clearly, the new data structure has query time $t = t' + 1$, space $s = s' + 1$ and word size w. Moreover, under this modification, we have

- $\Pr_{M,u,v}\left[\text{query algorithm computes } u^\intercal M v \text{ correctly}\right] \geq 1/2 + 2^{-\sqrt{n}/16}$.
- $\Pr_{u,v}\left[\text{query algorithm computes } u^\intercal M v \text{ correctly}\right] \geq 1/2$ for every M.

In the rest of the proof, we work with this modification and show that $t \geq \Omega\left(\min\left\{\frac{n}{\beta}, \frac{\sqrt{n}}{\log\frac{s\beta}{n}}\right\}\right)$, where $\beta = 2(w + \log sw/n)$. Observe that $\beta \leq n/256$; otherwise the lower bound is vacuous.

Assume by contradiction that $t \leq \min\left\{\frac{n}{256\beta}, \frac{\sqrt{n}}{256\log\frac{s\beta}{n}}\right\}$. Define $Z_M(u,v) = 1$ if the query algorithm correctly computes $u^\intercal M v$, and -1 otherwise. We have

$$\begin{aligned}\mathop{\mathbb{E}}_{M,u,v}\left[Z_M(u,v)\right] &= 2\Pr_{M,u,v}\left[\text{query algorithm computes } u^\intercal M v \text{ correctly}\right] - 1 \\ &\geq 2\cdot 2^{-\sqrt{n}/16}. \end{aligned} \tag{8}$$

Note that $\mathbb{E}_{M,u,v}\left[Z_M(u,v)\right]$ captures the *advantage* or *bias* of the data structure - it is much more convenient to work with the advantage than the probability of the query algorithm being correct.

Input: Alice's input is M and Bob's input is (u, v)
Output: Alice communicates a deterministic message and Bob computes $u^\intercal Mv$.

1 Let $Q_1 = \{uv^\intercal \mid Z_M(u,v) = 1\}$ and $Q_2 = \{uv^\intercal \mid Z_M(u,v) = -1\}$;
2 Apply Lemma 14 with Q_1, Q_2 to obtain a subset of cells S and subsets $Q_1' \subseteq Q_1$ and $Q_2' \subseteq Q_2$;
3 Let $b \in \{0,1\}$ be such that $\Pr_{u,v}[u^\intercal Mv = b \mid uv^\intercal \notin Q'] \geq \Pr_{u,v}[u^\intercal Mv = 1-b \mid uv^\intercal \notin Q']$, where $Q' = Q_1' \cup Q_2'$;
4 Alice communicates b followed by locations and contents of S;
5 **if** $uv^\intercal \in Q'$ **then** Bob uses the query algorithm to compute $u^\intercal Mv$;
6 **else** Bob outputs b;

Figure 1 One-way protocol on inputs $M, (u, v)$ computing $u^\intercal Mv$.

The following lemma, a variant of cell sampling, guarantees the existence of a small subset S of cells such that a large number of queries Q' can be computed by only accessing S, and $\mathbb{E}_{u,v}[Z_M(u,v) \mid uv^\intercal \in Q'] \approx \mathbb{E}_{u,v}[Z_M(u,v)]$.

▶ **Lemma 14.** *Let $M \in \mathbb{F}_2^{\sqrt{n}\times\sqrt{n}}$, and define the sets $Q_1 = \{uv^\intercal \mid Z_M(u,v) = 1\}$ and $Q_2 = \{uv^\intercal \mid Z_M(u,v) = -1\}$. If $t \leq \min\left\{\frac{n}{256\beta}, \frac{\sqrt{n}}{256\log\frac{s\beta}{n}}\right\}$, then there exits a subset of cells S, and subsets $Q_1' \subseteq Q_1$ and $Q_2' \subseteq Q_2$ such that*

1. $|S| = \left\lceil \frac{n}{128\beta} \right\rceil$,
2. $\Pr_{u,v}[uv^\intercal \in Q_1'] - \Pr_{u,v}[uv^\intercal \in Q_2'] \geq \mathbb{E}_{u,v}[Z_M(u,v)] \cdot 2^{-\sqrt{n}/16}$,
3. *$Q_1' \cup Q_2'$ is the set of all queries computed by accessing cells only in S.*

We move on to the final step of the proof of Theorem 3. What is left is to design a one-way protocol using the sets guaranteed by Lemma 14. The protocol is described in Figure 1. We will show the validity of this protocol by showing that both Alice and Bob know the subset Q' of queries. Since Alice's input is M, she knows the contents of all the cells, which gives S. With regard to knowing Q', the locations and contents of cells in S suffice. This is because the query algorithm can be simulated on all queries to check if any cell outside of S is being accessed. We are proving Theorem 3 by contradicting Theorem 11, which is achieved by the following.

▶ **Lemma 15.** *The protocol in Figure 1 has the following guarantees (a) Alice communicates fewer than $n/10$ bits, and (b) $\Pr_{M,u,v}[$Bob computes $u^\intercal Mv$ correctly$] \geq 1/2 + 1/2^{\sqrt{n}/8}$.*

Now, we need to prove Lemmas 14 and 15 to complete the proof of Theorem 3.

Proof of Lemma 14. Let S be a uniformly random subset of the cells of size $|S| = \left\lceil \frac{n}{128\beta} \right\rceil$. Define $D(u,v,S) = Z_M(u,v)$ if the query algorithm only accesses cells in S to compute $u^\intercal Mv$; otherwise $D(u,v,S) = 0$. By linearity of expectation,

$$\mathop{\mathbb{E}}_{u,v,S}[D(u,v,S)] = \mathop{\mathbb{E}}_{u,v}[Z_M(u,v)] \cdot \frac{\binom{s-t}{|S|-t}}{\binom{s}{|S|}} = \mathop{\mathbb{E}}_{u,v}[Z_M(u,v)] \cdot \frac{|S| \cdot (|S|-1) \cdots (|S|-t+1)}{s \cdot (s-1) \cdots (s-t+1)}$$
$$\geq \mathop{\mathbb{E}}_{u,v}[Z_M(u,v)] \cdot \left(\frac{|S|-t}{s}\right)^t.$$

Recall that $|S| \geq \frac{n}{128\beta}$ and $t \leq \frac{n}{256\beta}$. Moreover, $\beta = 2(w + \log sw/n) \geq 2$. This implies that

$$\left(\frac{|S|-t}{s}\right)^t \geq 2^{-t\cdot\log\frac{256s\beta}{n}} \geq 2^{-16t\cdot\log\frac{s\beta}{n}}.$$

So we get $\mathbb{E}_{u,v,S}\left[D(u,v,S)\right] \geq \mathbb{E}_{u,v}\left[Z_M(u,v)\right]\cdot 2^{-16\cdot t\cdot\log\frac{s\beta}{n}}$. Therefore, there exists an S such that

$$\mathop{\mathbb{E}}_{u,v}\left[D(u,v,S)\right] \geq \mathop{\mathbb{E}}_{u,v}\left[Z_M(u,v)\right]\cdot 2^{-16\cdot t\cdot\log\frac{s\beta}{n}} \geq \mathop{\mathbb{E}}_{u,v}\left[Z_M(u,v)\right]\cdot 2^{-\sqrt{n}/16},$$

where the last inequality follows from the fact that $16\cdot t\cdot\log\frac{s\beta}{n} \leq \sqrt{n}/16$. Setting

$$Q_1' = \{uv^\intercal \in Q_1 \mid D(u,v,S) = 1\} \text{ and } Q_2' = \{uv^\intercal \in Q_2 \mid D(u,v,S) = -1\}$$

completes the proof of the lemma. ◀

Proof of Lemma 15. We first prove part (a). Recall that $\beta = 2\left(w + \log\frac{sw}{n}\right)$. Let c be the number of bits communicated by Alice. By Proposition 12 and the definition of β,

$$\begin{aligned} c &\leq 1 + \left\lceil\frac{n}{128\beta}\right\rceil\cdot w + \left\lceil\frac{n}{128\beta}\right\rceil\cdot\log\frac{128e\cdot s\beta}{n} \\ &= 1 + \left\lceil\frac{n}{128\beta}\right\rceil\cdot\left(w + \log\frac{s\beta}{n}\right) + \left\lceil\frac{n}{128\beta}\right\rceil\cdot\log 128e. \end{aligned}$$

Since $\beta \geq 2w$, $\beta \geq 2\log\frac{s}{n}$ and $\beta \geq \log\beta$, we get that $w + \log\frac{s\beta}{n} \leq 2\beta$. Moreover, using the fact that $\left\lceil\frac{n}{128\beta}\right\rceil \leq \frac{n}{128\beta} + 1$, $\beta \geq 2$ and $\beta \leq n/256$, we can say that

$$\begin{aligned} c &\leq 1 + \frac{2n}{128} + 2\beta + \frac{n\log 128e}{128\beta} + \log 128e \\ &\leq 1 + \frac{2n}{128} + \frac{4.5n}{128(\beta/2)} + \frac{n}{128} + \log 128e \leq 10 + \frac{7.5n}{128} < \frac{n}{10}, \end{aligned}$$

where the last inequality follows from $n \geq 36$.

We now prove part (b) of the claim. Define $Z_M'(u,v) = 1$ if the Bob correctly computes $u^\intercal Mv$ and $Z_M'(u,v) = -1$ otherwise. The probability with which Bob correctly computes $u^\intercal Mv$ is given by $\left(1 + \mathbb{E}_{M,u,v}\left[Z_M'(u,v)\right]\right)/2$. We will show that $\mathbb{E}_{M,u,v}\left[Z_M'(u,v)\right] \geq 2\cdot 2^{-\sqrt{n}/8}$, which will imply that the probability of being correct is at least $1/2 + 2^{-\sqrt{n}/8}$.

Let Q_1, Q_2, Q_1', Q_2', and Q' be as defined in the protocol in Figure 1. We first establish some properties about these sets. We know that $\Pr_{u,v}[uv^\intercal \in Q_1] - \Pr_{u,v}[uv^\intercal \in Q_2] = \mathbb{E}_{u,v}\left[Z_M(u,v)\right]$. Moreover, the application of Lemma 14 in the protocol is valid since $t \leq \frac{n}{256\alpha}$, and hence

$$\Pr_{u,v}\left[uv^\intercal \in Q_1'\right] - \Pr_{u,v}\left[uv^\intercal \in Q_2'\right] \geq \mathop{\mathbb{E}}_{u,v}\left[Z_M(u,v)\right]\cdot 2^{-\sqrt{n}/16}. \tag{9}$$

Since Bob can simulate the query algorithm on Q' by accessing only S, which is guaranteed by Lemma 14, we have

$$\begin{aligned} \mathop{\mathbb{E}}_{u,v}\left[Z_M'(u,v)\right] &= \Pr_{u,v}\left[uv^\intercal \in Q'\right]\left(\Pr_{u,v}\left[uv^\intercal \in Q_1' \mid uv^\intercal \in Q'\right] - \Pr_{u,v}\left[uv^\intercal \in Q_2' \mid uv^\intercal \in Q'\right]\right) + \\ &\quad \Pr_{u,v}\left[uv^\intercal \notin Q'\right]\left(\Pr_{u,v}\left[u^\intercal Mv = b \mid uv^\intercal \notin Q'\right] - \Pr_{u,v}\left[u^\intercal Mv = 1-b \mid uv^\intercal \notin Q'\right]\right) \\ &\geq \left(\Pr_{u,v}\left[uv^\intercal \in Q_1'\right] - \Pr_{u,v}\left[uv^\intercal \in Q_2'\right]\right) \geq \mathop{\mathbb{E}}_{u,v}\left[Z_M(u,v)\right]\cdot 2^{-\sqrt{n}/16}, \end{aligned}$$

where the first inequality follows from the choice of b and the second inequality used Eq. (9).

To conclude,

$$\begin{aligned}\mathop{\mathbb{E}}_{M,u,v}[Z'_M(u,v)] = \mathop{\mathbb{E}}_{M}\left[\mathop{\mathbb{E}}_{u,v}[Z'_M(u,v)]\right] &\geq \mathop{\mathbb{E}}_{M}\left[\mathop{\mathbb{E}}_{u,v}[Z_M(u,v)]\cdot 2^{-\sqrt{n}/16}\right] \\ &= \mathop{\mathbb{E}}_{M}\left[\mathop{\mathbb{E}}_{u,v}[Z_M(u,v)]\right]\cdot 2^{-\sqrt{n}/16} \\ &= \mathop{\mathbb{E}}_{M,u,v}[Z_M(u,v)]\cdot 2^{-\sqrt{n}/16} \geq 2\cdot 2^{-\sqrt{n}/8},\end{aligned}$$

where the last inequality follows from Eq. (8). ◀

References

1 Pankaj K. Agarwal. Geometric Range searching. In *CRC Handbook of Computational Geometry*. CRC, 1997.

2 Miklós Ajtai. A Lower Bound for Finding Predecessors in Yao's Cell Probe Model. *Combinatorica*, 8(3), 1988.

3 Josh Alman and Ryan Williams. Probabilistic Rank and Matrix Rigidity. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 641–652. ACM, 2017.

4 Noga Alon and Gil Cohen. On Rigid Matrices and U-Polynomials. *Computational Complexity*, 24(4):851–879, December 2015.

5 Noga Alon, Rina Panigrahy, and Sergey Yekhanin. Deterministic Approximation Algorithms for the Nearest Codeword Problem. In *APPROX '09 / RANDOM '09*, pages 339–351, 2009.

6 V. L. Artazarov, E. A. Dinic, D. A. Kronrod, and I. A. Faradzev. On Economical Construction of the Transitive Closure of a Directed Graph. *Soviet Math. Dokl.*, 11:1209–1210, 1970.

7 Joshua Brody and Kasper Green Larsen. Adapt or Die: Polynomial Lower Bounds for Non-Adaptive Dynamic Data Structures. *Theory of Computing*, 11(19):471–489, 2015.

8 Diptarka Chakraborty, Lior Kamma, and Kasper Green Larsen. Tight Cell Probe Bounds for Succinct Boolean Matrix-Vector Multiplication. In *Proc. 50th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 1297–1306, 2018.

9 Arkadev Chattopadhyay, Michal Koucký, Bruno Loff, and Sagnik Mukhopadhyay. Simulation Beats Richness: New Data-Structure Lower Bounds. In *Proc. 50th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 1013–1020, 2018.

10 Raphaël Clifford, Allan Grønlund, and Kasper Green Larsen. New Unconditional Hardness Results for Dynamic and Online Problems. In *IEEE 56th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1089–1107, 2015.

11 Henry Corrigan-Gibbs and Dmitry Kogan. The Function-Inversion Problem: Barriers and Opportunities. *Electronic Colloquium on Computational Complexity (ECCC)*, 25:182, 2018.

12 Zeev Dvir and Benjamin Edelman. Matrix Rigidity and the Croot-Lev-Pach Lemma. *arXiv preprint*, 2017. `arXiv:1708.01646`.

13 Zeev Dvir, Alexander Golovnev, and Omri Weinstein. Static Data Structure Lower Bounds Imply Rigidity. In *Proc. 51st Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 967–978, 2019. see `arXiv:1811.02725v3`. `doi:10.1145/3313276.3316348`.

14 Zeev Dvir and Allen Liu. Fourier and Circulant Matrices Are Not Rigid. In *34th Computational Complexity Conference (CCC 2019)*, volume 137, pages 17:1–17:23, 2019.

15 Michael Fredman and Michael Saks. The Cell Probe Complexity of Dynamic Data Structures. In *ACM Symposium on Theory of Computing (STOC)*, 1989.

16 Joel Friedman. A Note on Matrix Rigidity. *Combinatorica*, 13(2):235–239, June 1993.

17 Anna Gál and Peter Bro Miltersen. The Cell Probe Complexity of Succinct Data Structures. *Theoretical Computer Science*, 379(3):405–417, 2007.

18 Monika Henzinger, Sebastian Krinninger, Danupon Nanongkai, and Thatchaphol Saranurak. Unifying and Strengthening Hardness for Dynamic Problems via the Online Matrix-Vector Multiplication Conjecture. In *Proc. 47th Annual ACM Symposium on Theory of Computing (STOC)*, pages 21–30, 2015.

19 Stasys Jukna and Georg Schnitger. Min-Rank Conjecture for Log-Depth Circuits. *Journal of Computer and System Sciences*, 77(6):1023–1038, 2011.

20 Kasper Green Larsen. Higher Cell Probe Lower Bounds for Evaluating Polynomials. In *FOCS*, pages 293–301. IEEE Computer Society, 2012.

21 Kasper Green Larsen. The Cell Probe Complexity of Dynamic Range Counting. In *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 - 22, 2012*, pages 85–94, 2012.

22 Kasper Green Larsen, Omri Weinstein, and Huacheng Yu. Crossing the Logarithmic Barrier for Dynamic Boolean Data Structure Lower Bounds. In *Proc. 50th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 978–989, 2018.

23 Kasper Green Larsen and Ryan Williams. Faster Online Matrix-Vector Multiplication. In *Proc. 28th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2182–2189, 2017.

24 Satyanarayana V. Lokam. Complexity Lower Bounds Using Linear Algebra. *Foundations and Trends® in Theoretical Computer Science*, 4(1–2):1–155, 2009. doi:10.1561/0400000011.

25 Peter Bro Miltersen. Lower Bounds for Union-Split-Find Related Problems on Random Access Machines. In *Proceedings of the 26th Annual Symposium on the Theory of Computing*, pages 625–634, New York, May 1994. ACM Press.

26 Peter Bro Miltersen, Noam Nisan, Shmuel Safra, and Avi Wigderson. On Data Structures and Asymmetric Communication Complexity. *Journal of Computer and System Sciences*, 57(1):37–49, 1998.

27 Rina Panigrahy, Kunal Talwar, and Udi Wieder. Lower Bounds on Near Neighbor Search via Metric Expansion. In *Proc. 51st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 805–814, 2010.

28 Mihai Pătraşcu. Unifying the Landscape of Cell-Probe Lower Bounds. *SIAM Journal on Computing*, 40(3):827–847, 2011. See also FOCS'08, arXiv:1010.3783.

29 Mihai Pătraşcu and Erik D. Demaine. Logarithmic Lower Bounds in the Cell-Probe Model. *SIAM Journal on Computing*, 35(4):932–963, 2006. See also STOC'04, SODA'04.

30 Mihai Pătraşcu and Mikkel Thorup. Higher Lower Bounds for Near-Neighbor and Further Rich Problems. *SIAM Journal on Computing*, 39(2):730–741, 2010. See also FOCS'06.

31 Pavel Pudlák, Vojtech Rödl, and Jirí Sgall. Boolean Circuits, Tensor Ranks, and Communication Complexity. *SIAM Journal on Computing*, 26(3):605–633, 1997.

32 Anup Rao and Amir Yehudayoff. Communication Complexity. preprint at https://homes.cs.washington.edu/~anuprao/pubs/book.pdf.

33 M.A. Shokrollahi, D.A. Spielman, and V. Stemann. A Remark on Matrix Rigidity. *Information Processing Letters*, 64(6):283–285, 1997. doi:10.1016/S0020-0190(97)00190-7.

34 Amir Shpilka and Amir Yehudayoff. Arithmetic Circuits: A Survey of Recent Results and Open Questions. *Foundations and Trends® in Theoretical Computer Science*, 5(3–4):207–388, 2010.

35 Leslie G. Valiant. Graph-Theoretic Arguments in Low-Level Complexity. In Jozef Gruska, editor, *Mathematical Foundations of Computer Science*, pages 162–176, 1977.

36 Leslie G. Valiant. Why Is Boolean Complexity Theory Difficult? In *Poceedings of the London Mathematical Society Symposium on Boolean Function Complexity*, pages 84–94, New York, NY, USA, 1992. Cambridge University Press. URL: http://dl.acm.org/citation.cfm?id=167687.167712.

37 Emanuele Viola. On the Power of Small-Depth Computation. *Foundations and Trends® in Theoretical Computer Science*, 5(1):1–72, 2009.

38 Emanuele Viola. Lower Bounds for Data Structures with Space Close to Maximum Imply Circuit Lower Bounds. *Electronic Colloquium on Computational Complexity (ECCC)*, 25:186, 2018.

39 Henning Wunderlich. On a Theorem of Razborov. *Computational Complexity*, 21(3):431–477, 2012.

40 Andrew Yao. Should Tables be Sorted? *JACM: Journal of the ACM*, 28, 1981.

Computationally Data-Independent Memory Hard Functions

Mohammad Hassan Ameri
Department of Computer Science, Purdue University, West Lafayette, IN, USA
https://www.cs.purdue.edu/homes/mameriek/
mameriek@purdue.edu

Jeremiah Blocki
Department of Computer Science, Purdue University, West Lafayette, IN, USA
https://www.cs.purdue.edu/homes/jblocki
jblocki@purdue.edu

Samson Zhou
School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA
https://samsonzhou.github.io/
samsonzhou@gmail.com

Abstract

Memory hard functions (MHFs) are an important cryptographic primitive that are used to design egalitarian proofs of work and in the construction of moderately expensive key-derivation functions resistant to brute-force attacks. Broadly speaking, MHFs can be divided into two categories: data-dependent memory hard functions (dMHFs) and data-independent memory hard functions (iMHFs). iMHFs are resistant to certain side-channel attacks as the memory access pattern induced by the honest evaluation algorithm is independent of the potentially sensitive input e.g., password. While dMHFs are potentially vulnerable to side-channel attacks (the induced memory access pattern might leak useful information to a brute-force attacker), they can achieve higher cumulative memory complexity (CMC) in comparison than an iMHF. In particular, any iMHF that can be evaluated in N steps on a sequential machine has CMC *at most* $\mathcal{O}\left(\frac{N^2 \log\log N}{\log N}\right)$. By contrast, the dMHF scrypt achieves maximal CMC $\Omega(N^2)$ – though the CMC of scrypt would be reduced to just $\mathcal{O}(N)$ after a side-channel attack.

In this paper, we introduce the notion of computationally data-independent memory hard functions (ciMHFs). Intuitively, we require that memory access pattern induced by the (randomized) ciMHF evaluation algorithm appears to be independent from the standpoint of a computationally bounded eavesdropping attacker – even if the attacker selects the initial input. We then ask whether it is possible to circumvent known upper bound for iMHFs and build a ciMHF with CMC $\Omega(N^2)$. Surprisingly, we answer the question in the affirmative when the ciMHF evaluation algorithm is executed on a two-tiered memory architecture (RAM/Cache).

We introduce the notion of a k-restricted dynamic graph to quantify the continuum between unrestricted dMHFs ($k = n$) and iMHFs ($k = 1$). For any $\epsilon > 0$ we show how to construct a k-restricted dynamic graph with $k = \Omega(N^{1-\epsilon})$ that provably achieves maximum cumulative pebbling cost $\Omega(N^2)$. We can use k-restricted dynamic graphs to build a ciMHF provided that cache is large enough to hold k hash outputs and the dynamic graph satisfies a certain property that we call "amenable to shuffling". In particular, we prove that the induced memory access pattern is indistinguishable to a polynomial time attacker who can monitor the locations of read/write requests to RAM, but not cache. We also show that when $k = o\left(N^{1/\log\log N}\right)$, then any k-restricted graph with constant indegree has cumulative pebbling cost $o(N^2)$. Our results almost completely characterize the spectrum of k-restricted dynamic graphs.

2012 ACM Subject Classification Security and privacy → Hash functions and message authentication codes

Keywords and phrases Computationally Data-Independent Memory Hard Function, Cumulative Memory Complexity, Dynamic Pebbling Game

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.36

Related Version https://arxiv.org/pdf/1911.06790.pdf

Funding The opinions in this paper are those of the authors and do not necessarily reflect the position of the National Science Foundation or IARPA.
Mohammad Hassan Ameri: Supported in part by the National Science Foundation under award #1755708 and by IARPA under the HECTOR program.
Jeremiah Blocki: Research supported in part by NSF Award #1755708.

Acknowledgements Part of this work was done while Samson Zhou was a postdoctoral fellow at Indiana University. We would like to thank anonymous ITCS 2020 reviewers for thoughtful comments which helped us to improve the paper.

1 Introduction

Memory hard functions (MHFs) [1, 27] are a central component in the design of password hashing functions [9], egalitarian proofs of work [21], and moderately hard key-derivation functions [27]. In the setting of password hashing, the objective is to design a function that can be computed relatively quickly on standard hardware for honest users, but is prohibitively expensive for an offline attacker to compute millions or billions of times while checking each password in a large cracking dictionary. The first property allows legitimate users to authenticate in a reasonable amount of time, while the latter goal discourages brute-force offline guessing attacks, even on low-entropy secrets such as passwords, PINs, and biometrics. The objective is complicated by attackers that use specialized hardware such as Field Programmable Gate Arrays (FPGAs) or Application Specific Integrated Circuits (ASICs) to significantly decrease computation costs by several orders of magnitude, compared to an honest user using standard hardware. For example, the Antminer S17, an ASIC Bitcoin miner exclusively configured for SHA256 hashes, can compute up to 56 trillion hashes per second, while the rate of many standard CPUs and GPUs are limited to 200 million hashes per second and 1 billion hashes per second, respectively.

Memory hard functions were developed on the observation that memory costs such as chip area tend to be equitable across different architectures. Therefore, the cost of evaluating an ideal "egalitarian" function would be dominated by memory costs. Blocki et al. [12] argued that key derivation functions without some form of memory hardness provide insufficient defense against a economically motivated offline attacker under the constraint of reasonable authentication times for honest users. In fact, most finalists in the 2015 Password Hashing Competition claimed some form of memory hardness [22, 9, 24]. To quantify these memory costs, memory hardness [27] considers the cost to build, obtain, and empower the necessary hardware to compute the function. One particular metric heavily considered by recent cryptanalysis [2, 5, 4, 3, 15] is cumulative memory complexity (CMC) [7], which measures the amortized cost of any parallel algorithm evaluating the function on several distinct inputs. Despite known hardness results for quantifying [16] or even approximating [13] a function's CMC, even acquiring asymptotic bounds provide automatic bounds for other attractive metrics such as space-time complexity [26] or energy complexity [29, 14].

Data-Dependent vs. Data-Independent Memory Hard Functions

At a high level, memory hard functions can be categorized into two design paradigms: data-dependent memory hard functions (dMHFs) and data-independent memory hard functions (iMHFs). dMHFs induce memory access patterns that depend on the input, but can achieve

high memory hardness with potentially relatively easy constructions [6]. However, dMHFs are also vulnerable to side-channel attacks due to their inherent data dependent memory access patterns [8]. Examples of dMHFs include scrypt [27], Argon2d [10] and Boyen's halting puzzles [19]. On the other hand, iMHFs have memory access patterns that are independent of the input, and therefore resist certain side-channel attacks such as cache timing [8]. Examples of iMHFs include 2015 Password Hashing Competition (PHC) winner Argon2i [9], Balloon Hashing [17] and DRSample [4]. iMHFs with high memory hardness can be more technically challenging to design, but even more concerning is the inability of iMHFs to be maximally memory hard.

Alwen and Blocki [2] proved that the CMC of any iMHF running in time N is at most $\mathcal{O}\left(\frac{N^2 \log\log N}{\log N}\right)$, while the dMHF scrypt has cumulative memory complexity $\Omega(N^2)$ [6], which matches the maximal amount and cannot be obtained by any iMHF. However, the cumulative memory complexity of a dMHF can be greatly decreased through a side-channel attack, if an attacker has learned the memory access pattern induced by the true input. Namely, a brute-force attacker can preemptively quit evaluation on a guess y once it is clear that the induced memory access pattern on input y differs from that on the true input x. For example, the cumulative memory complexity of scrypt after a side-channel attack is just $\mathcal{O}(N)$.

Ideally, we would like to obtain a family of memory hard functions with cumulative memory complexity $\Omega(N^2)$ without any vulnerability to side-channel attacks. A natural approach would be some sort of hybrid between data-dependent and data-independent modes, such as Argon2id, which runs the MHF in data-independent mode for $\frac{N}{2}$ steps before switching to data-dependent mode for the final $\frac{N}{2}$ steps. Although the cumulative memory complexity is the maximal $\Omega(N^2)$ if there is no side-channel attack, the security still reduces to that of the underlying iMHF (e.g., Argon2i) if there is a side-channel attack. Hence even for a hybrid mode, the cumulative memory complexity is just $\mathcal{O}\left(\frac{N^2 \log\log N}{\log N}\right)$ (or lower) in the face of a side-channel attack. Thus in this paper we ask:

> In the presence of side-channel attacks, does there exist a family of functions with $\Omega(N^2)$ cumulative memory complexity?

1.1 Our Contributions

Surprisingly, we answer the above question in the affirmative for a natural class of side-channel attacks that observe the read/write memory locations. We introduce the concept of computationally data-independent memory hard functions to overcome the inability of data-independent memory hard functions to be maximally memory hard [2] without the common side-channel vulnerabilities of data-dependent memory hard functions [8]. Our constructions work by randomly "shuffling" memory blocks in cache before they are stored in RAM (where the attacker can observe the locations of read/write requests). Intuitively, each time $\mathsf{MHF.Eval}(x)$ is executed the induced memory access pattern will appear different due to this scrambling step. The goal is to ensure that an attacker can not even distinguish between the observed memory access pattern on two known inputs $x \neq y$.

Towards this goal we define k-restricted dynamic graphs as a tool to quantify the continuum between dMHFs and iMHFs. Intuitively, in a k-restricted dynamic graph $G = (V = [N], E)$ we have $\mathsf{parents}(v) = \{v-1, r(v)\}$ where the second (data-dependent) parent $r(v) \in R_v$ must be selected from a fixed (data-independent) restricted set $R_v \subseteq V$ of size $|R_v| \leq k$. When $k = 1$ the function is an iMHF (the parent $r(v) \in R_v$ of each node v is fixed in a data-independent manner) and when $k = N$ the function is an unrestricted dMHF – scrypt

and Argon2d are both examples of unrestricted dMHFs. Intuitively, when k is small it becomes easier to scramble the labels R_v in memory so that the observed memory access patterns on two known inputs $x \neq y$ are computationally indistinguishable.

We then develop a graph gadget that generates a family of ciMHFs using k-restricted graphs. Using this family of ciMHFs, we characterize the tradeoffs between the value of k and the overall cumulative memory cost of k-restricted graphs.

Impossibility Results for Small k

Since k-restricted graphs correspond to iMHFs for $k = 1$, and it is known that $\mathsf{cc}(\mathbb{G}) = \mathcal{O}\left(\frac{N^2 \log\log N}{\log N}\right)$ for any family $\mathbb{G}$ of iMHFs [2], then one might expect that it is impossible to obtain maximally memory hard ciMHFs for small k. Indeed, our first result shows that this intuition is correct; we show that for any $k = o\left(N^{1/\log\log N}\right)$, then any family of k-restricted graphs $\mathbb{G}$ with constant indegree has $\mathsf{cc}(\mathbb{G}) = o(N^2)$.

▶ **Theorem 1.** *Let $\mathbb{G}$ be any family of k-restricted dynamic graphs with constant $\mathsf{indeg}(\mathbb{G})$. Then*

$$\mathsf{cc}(\mathbb{G}) = \mathcal{O}\left(\frac{N^2}{\log\log N} + N^{2-1/2\log\log N}\sqrt{k^{1-1/\log\log N}}\right).$$

Thus for $k = o\left(N^{1/\log\log N}\right)$, we have $\mathsf{cc}(\mathbb{G}) = o(N^2)$.

We prove this result in Theorem 10 and Corollary 11 in Section 3 by generalizing ideas from the pebbling attack of Alwen and Blocki [2] against any iMHF to k-restricted dynamic graphs graphs. The pebbling attack of Alwen and Blocki [2] exploited the fact that any constant indegree DAG G is somewhat depth-reducible e.g., we can always find a set $S \subseteq V(G)$ of size $e = \mathcal{O}\left(\frac{N \log\log N}{\log N}\right)$ such that any path in $G - S$ has length at most $d = \frac{N}{\log^2 N}$. The attack then proceeds in a number of *light phases* and *balloon phases*, where the goal of light phase i is to place pebbles on the interval $[ig + 1, (i + 1)g]$, for some parameter g to be optimized. At the same time, the attacker discards pebbles on all nodes v unless $v \in S$ or unless v is a parent of one of the next g nodes $[ig + 1, (i + 1)g]$ that we want to pebble. Once light phase i is completed, balloon phase i uses the pebbles on S to recover all previously discarded pebbles. Note that balloon phase i thus promises that pebbles are placed on the parents of the nodes $[(i + 1)g + 1, (i + 2)g]$, so that light phase $i + 1$ can then be initiated and so forth.

One key difference is that we must maintain pebbles on all gk nodes $u \in \bigcup_{v\in[ig+1,(i+1)g]} R_v$ that are "potential parents" of the next g nodes $[ig+1, (i+1)g]$. The total cost of the pebbling attack is $\mathcal{O}\left(eN + gkN + \frac{N^2 d}{g}\right)$, which is identical to [2] when $k = 1$ for (e, d)-reducible DAGs. In general for small values of k, the dynamic pebbling strategy can still achieve cumulative memory cost $o(N^2)$ after optimizing for g.

Maximally Hard k-restricted dMHF

In Section 4, we show how to construct a k-restricted dynamic graph for $k = \mathcal{O}\left(N^{1-\epsilon}\right)$, which has cumulative pebbling cost $\Omega(N^2)$ for any constant $\epsilon > 0$. Intuitively, our goal is to force the pebbling strategy to maintain $\Omega(N)$ pebbles on the graph for $\Omega(N)$ steps or pay a steep penalty. In particular, we want to ensure that if there are $o(N)$ pebbles on the graph at time i then the cumulative pebbling cost to advance a pebble just $2k = \mathcal{O}\left(N^{\epsilon}\right)$ steps is at least $\Omega(N^{2-\epsilon})$ with high probability. This would imply that the pebbling strategy either keeps $\Omega(N)$ pebbles on the graph for $\Omega(N)$ steps or that the pebbling strategy pays a penalty of $\Omega(N^{2-\epsilon})$ at least $\Omega\left(\frac{N}{k}\right) = \Omega(N^{\epsilon})$ times. In either case the cumulative pebbling cost will be $\Omega(N^2)$.

One of our building blocks is the "grates" construction of Schnitger [30] who, for any $\epsilon > 0$, showed how to construct a constant indegree DAG G_ϵ that is (e, d)-depth robust graph with $e = \Omega(N)$ and $d = \Omega(N^{1-\epsilon})$. Our second building block is the superconcentrator [28, 26] graph. By overlaying the DAG G_ϵ with a superconcentrator, we can spread out the data-dependent edges on the top layer of our graph to ensure that (with high probability) advancing a pebble $2k = \mathcal{O}(N^\epsilon)$ steps on the top layer starting from a pebbling configuration with $o(N)$ pebbles on the graph requires us to repebble an (e, d)-depth robust graph with $e = \Omega(N)$ and $d = \Omega(N^{1-\epsilon})$. This is sufficient since Alwen et al. [5] showed that the cumulative pebbling cost of any (e, d)-depth robust graph is at least ed.

Open Question

We emphasize that we only show that any dynamic pebbling strategy for our k-restricted dynamic graph has cumulative cost $\Omega(N^2)$. This is not quite the same as showing that our dMHF has CMC $\Omega(N^2)$ in the parallel random oracle model. For static graphs, we know that the CMC of an iMHF is captured by the cumulative pebbling cost of the underlying DAG [7]. We take the dynamic pebbling lower bound as compelling evidence that the corresponding MHF has maximum cumulative memory cost. Nevertheless, proving (or disproving) that the CMC of a dMHF is captured by the cumulative cost of the optimal dynamic pebbling strategy for the underlying dynamic graph is still an open question that is outside the scope of the current work.

ciMHF Implementation Through Shuffling

The only problem is that the above k-restricted dynamic graph is actually a data-dependent construction; once the input x is fixed, the memory access patterns of the above construction is completely deterministic! Thus a side-channel attacker that obtains a memory access pattern will possibly be able to distinguish between future inputs. Our solution is to have a hidden random key K for each separate evaluation of the password hash. The hidden random key K does not alter the hash value of x in any manner, so we emphasize that there is no need to know the value of the hidden key K to perform computation. However, each computation using a separate value of K induces a different memory access pattern, so that no information is revealed to side-channel attackers looking at locations of read/write instructions.

Let L be a set of the last N consecutive nodes from our previous graph construction, which we suppose is called G_0. We form G by appending a path of length N to the end of G_0. We introduce a gadget that partitions the nodes in L into blocks $B_1, B_2, \ldots, B_{N/k}$ of size k each. We then enforce that for $i \in [N]$ and $j = i \mod k$, the i^{th} node in the final N nodes of G has a parent selected uniformly at random from B_{j+1}, depending on the input x. Thus to compute the label of i, the evaluation algorithm should know the labels of all nodes in B_{j+1}.

We allow the evaluation algorithm to manipulate the locations of these labels so that the output of the algorithm remains the same, but each computation induces a different memory access pattern. Specifically, the random key K induces a shuffling of the locations of the information within each block of the block partition gadget. Thus if the size of each block is sufficiently large, then with high probability, two separate computations of the hash for the same password will yield distinct memory access patterns, effectively computationally data-independent. Then informally, a side-channel attacker will not be able to use the memory access patterns to distinguish between future inputs.

In fact, this approach works for a general class of graphs satisfying a property that we call "amenable to shuffling". We characterize the properties of the dynamic graphs that are amenable to shuffling in Section 5 and show that k-restricted dynamic graphs that are ameanble to shuffling can be used in the design of MHFs to yield computationally data-independent sequential evaluation algorithms.

▶ **Theorem 2.** *For each DAG G that is amenable to shuffling, there exists a computationally data-independent sequential evaluation algorithm computing a MHF based on the graph G that runs in time $\mathcal{O}(N)$. (Informal, see Theorem 24.)*

We believe that our techniques for converting graphs that are amenable to shuffling to ciMHFs may be of independent interest.

Finally, we provide a version of our dMHF with $\Omega(N^2)$ cumulative memory complexity that is amenable to shuffling. Combining this maximally hard k-restricted dMHF using a DAG that is amenable to shuffling with Theorem 2, we obtain a maximally hard ciMHF.

▶ **Theorem 3.** *Let $0 < \epsilon < 1$ be a constant and $k = \Omega(N^\epsilon)$. Then there exists a family $\mathbb{G}$ of k-restricted graphs with $\mathsf{cc}(\mathbb{G}) = \Omega(N^2)$ that is amenable to shuffling.*

We prove Theorem 3 in Section 6, introducing the necessary formalities for computationally data-independent memory hard functions and the underlying systems model. Our results in Theorem 1 and Theorem 3 almost completely characterize the spectrum of k-restricted graphs. In fact, for a graph G drawn uniformly at random from our distribution $\mathbb{G}$ in Theorem 3 and any pebbling strategy S, not only do we have $\mathop{\mathbb{E}}_{G \sim \mathbb{G}}[\mathsf{cc}(S, G)] = \Omega(N^2)$, but we also have $\mathsf{cc}(S, G) = \Omega(N^2)$ with high probability.

2 Preliminaries

We use the notation $[N]$ to denote the set $\{0, 1, \ldots, N-1\}$. For two numbers x and y, we use $x \circ y$ to denote their concatenation.

Given a directed acyclic graph (DAG) $G = (V, E)$ and a node $v \in V$, we use $\mathsf{parents}_G(v) = \{u \ : \ (u, v) \in E\}$ to denote the parents of node v. We use $\mathsf{ancestors}_G(v) = \bigcup_{i \geq 1} \mathsf{parents}^i_G(v)$ to denote the set of all ancestors of v – here, $\mathsf{parents}^2_G(v) = \mathsf{parents}_G(\mathsf{parents}_G(v))$ and $\mathsf{parents}^{i+1}_G(v) = \mathsf{parents}_G\left(\mathsf{parents}^i_G(v)\right)$. We use $\mathsf{indeg}(v) = |\mathsf{parents}(v)|$ to denote the number of incoming edges into v and define $\mathsf{indeg}(G) = \max_{v \in V} \mathsf{indeg}(v)$. Given a set $S \subseteq V$, we use $G - S$ to refer to the graph obtained by deleting all nodes in S and all edges incident to S. We use $\mathsf{depth}(G)$ to denote the number of nodes in the longest directed path in G.

▶ **Definition 4.** *A DAG $G = (V, E)$ is* (e, d)-reducible *if there exists a subset $S \subseteq V$ of size $|S| \leq e$ such that any directed path P in G of length d contains at least one node in S. We call such a set S a* depth-reducing set*. If G is not (e, d)-reducible, then we say that G is* (e, d)-depth robust*.*

For a DAG $G = (V = [N], E)$, we use $G_{\leq i}$ to denote the subgraph of G induced by $[i]$. In other words, $G_{\leq i} = (V', E')$ for $V' = [i]$ and $E' = \{(a, b) \in E \,|\, a, b \leq i\}$.

The Parallel Random Oracle Model

We review the parallel random oracle model (pROM), as introduced by Alwen and Serbinenko [7]. There exists a probabilistic algorithm $\mathcal{A}^{\mathcal{H}}$ that serves as the main computational unit, where $\mathcal{A}^{\mathcal{H}}$ has access to an arbitrary number of parallel copies of an oracle $\mathcal{H}$ sampled uniformly at random from an oracle set $\mathbb{H}$ and proceeds to do computation in a number of

rounds. In each round i, $\mathcal{A}^{\mathcal{H}}$ maintains a state σ_i along with initial input x. $\mathcal{A}^{\mathcal{H}}$ determines a batch of queries $\mathbf{q}_i$ to send to $\mathcal{H}$, receives and processes the responses to determine an updated state σ_{i+1}. At some point, $\mathcal{A}^{\mathcal{H}}$ completes its computation and outputs the value $\mathcal{A}^{\mathcal{H}}(x)$.

We say that $\mathcal{A}^{\mathcal{H}}$ computes a function $f_{\mathcal{H}}$ on input x with probability ϵ if $\mathbf{Pr}\left[\mathcal{A}^{\mathcal{H}}(x) = f_{\mathcal{H}}\right] \geq \epsilon$, where the probability is taken over the internal randomness of $\mathcal{A}$. We say that $\mathcal{A}^{\mathcal{H}}$ uses t running time if it outputs $\mathcal{A}^{\mathcal{H}}(x)$ after round t. In that case, we also say $\mathcal{A}^{\mathcal{H}}$ uses space $\sum_{i=1}^{t} |\sigma_i|$ and that $\mathcal{A}^{\mathcal{H}}$ makes q queries if $\sum_{i=1}^{t} \leq q$.

The Ideal Cipher Model

In the ideal cipher model (ICM), there is a publicly available random block cipher, which has a κ-bit key K and an N bit input and output. Equivalently, all parties, including any honest parties and adversaries, have access to a family of 2^{κ} independent random permutations of $[N]$. Moreover for any given key K and $x \in [N]$, both encryption $\mathsf{Enc}(K, x)$ and decryption $\mathsf{Dec}(K, x)$ queries can be made to the random block cipher.

Graph Pebbling

The goal of the (black) pebbling game is to place pebbles on all sink nodes of some input directed acyclic graph (DAG) $G = (V, E)$. The game proceeds in rounds, and each round i consists of a number of pebbles $P_i \subseteq V$ placed on a subset of the vertices. Initially, the graph is unpebbled, $P_0 = \emptyset$, and in each round $i \geq 1$, we may place a pebble on $v \in P_i$ if either all parents of v contained pebbles in the previous round ($\mathsf{parents}(v) \subseteq P_{i-1}$) or if v already contained a pebble in the previous round ($v \in P_{i-1}$). In the sequential pebbling game, at most one new pebble can be placed on the graph in any round (i.e., $|P_i \backslash P_{i-1}| \leq 1$), but this restriction does not apply in the parallel pebbling game.

We use $\mathcal{P}_G^{\parallel}$ to denote the set of all valid parallel pebblings of a fixed graph G. The *cumulative cost* of a pebbling $P = (P_1, \ldots, P_t) \in \mathcal{P}_G^{\parallel}$ is the quantity $\mathsf{cc}(P) := |P_1| + \ldots + |P_t|$ that represents the sum of the number of pebbles on the graph during every round. The (parallel) *cumulative pebbling cost* of the fixed graph G, denoted $\mathsf{cc}(G) := \min_{P \in \mathcal{P}_G^{\parallel}} \mathsf{cc}(P)$, is the cumulative cost of the best legal pebbling of G.

▶ **Definition 5** (Dynamic/Static Pebbling Graph)**.** *We define a* dynamic pebbling graph *as a distribution $\mathbb{G}$ over directed acyclic graphs $G = (V = [N], E)$ with edges $E = \{(i-1, i) : i \leq N\} \cup \{(r(i), i) : i \leq N\}$, where $r(i) < i - 1$ is a randomly chosen directed edge. We say that an edge $(r(i), i)$ is* dynamic *if $r(i)$ is not chosen until a black pebbled is place on node $i - 1$. We say that the graph is* static *if none of the edges are dynamic.*

We now define a labeling of a graph G.

▶ **Definition 6.** *Given a DAG $G = (V = [N], E)$ and a random oracle function $H : \Sigma^* \to \Sigma^w$ over an alphabet Σ, we define the labeling of graph G as $L_{G,H} : \Sigma^* \to \Sigma^*$. In particular, given an input x the (H, x) labeling of G is defined recursively by*

$$L_{G,H,x}(v) = \begin{cases} H(v \circ x), & \mathsf{indeg}(v) = 0 \\ H\left(v \circ L_{G,H,x}(v_1) \circ \cdots \circ L_{G,H,x}(v_d)\right), & \mathsf{indeg}(v) > 0, \end{cases}$$

where $v_1, \ldots, v_d$ are the parents of v in G, according to some predetermined lexicographical order. We define $f_{G,H}(x) = L_{G,H,x}(s_1) \circ \ldots \circ L_{G,H,x}(s_k)$, where $s_1, \ldots, s_k$ are the sinks of G sorted lexicographically by node index. If there is a single sink node s_G then $f_{G,H}(x) = L_{G,H,x}(s_G)$. We omit the subscripts G, H, x when the dependency on the graph G and hash function H is clear. For a distribution of dynamic graphs $\mathbb{G}$, we say $f_{\mathbb{G},H}(x) = f_{G,H}(x)$ once a dynamic graph G has been determined from the choice of H and x.

For a node i, we define $\mathsf{PotentialParents}(i)$ to be set Y_i of minimal size such that $\mathbf{Pr}[r(i) \in Y_i] = 1$. We now define k-restricted dynamic graphs, which can characterize both dMHFs and iMHFs.

▶ **Definition 7** (k-Restricted Dynamic Graph). *We say that a dynamic pebbling graph $\mathbb{G}$ is k-restricted if for all i, $\mathsf{PotentialParents}(i) \leq k$.*

Observe that $k = 1$ corresponds to an iMHF while $k = N$ corresponds to a dMHF. Hence, k-restricted dynamic graphs can be viewed as spectrum between dMHFs and iMHFs.

We define the cumulative cost of pebbling a dynamic graph similar to the definition of cumulative cost of pebblings on static graphs. We first require the following definition of a dynamic pebbling strategy:

▶ **Definition 8** (Dynamic Pebbling Strategy). *A dynamic pebbling strategy S is a function that takes as input*
(1) *an integer $i \leq N$*
(2) *an initial pebbling configuration $P_0^i \subseteq [i]$ with $i \in P_0^i$*
(3) *a partial graph $G_{\leq i+1}$*
The output of $S(i, P_0^i, G_{\leq i+1})$ is a legal sequence of pebbling moves $P_1^i, \ldots, P_{r_i}^i$ that will be used in the next phase, to place a pebble on node $i+1$, so that $i+1 \in P_{r_i}^i \subseteq [i+1]$. Given $G \sim \mathbb{G}$ we can abuse notation and write $S(G)$ for the valid pebbling produced by S on the graph G i.e., $P_1^0, \ldots, P_{r_0}^0, P_1^1, \ldots, P_{r_1}^1, \ldots, P_{r_1}^{N-1}, \ldots, P_{r_{N-1}}^{N-1}$. Here, $P_1^i, \ldots, P_{r_i}^i = S(i, P_0^i, G_{\leq i+1})$ where $P_0^i = P_{r_{i-1}}^{i-1}$ and for $i = 1$ we set $P_0^0 = \emptyset$.

We thus define $\mathsf{cc}(S, G)$ to the pebbling cost of strategy S when we sample a dynamic graph G and $\mathsf{cc}(S, \mathbb{G}) = \mathop{\mathbb{E}}_{G \sim \mathbb{G}}[\mathsf{cc}(S, G)]$. Finally, we define $\mathsf{cc}(\mathbb{G}) = \min_S \mathsf{cc}(S, \mathbb{G})$, where the minimum is taken over all dynamic pebbling strategies S. More generally, we define $\mathsf{cc}(S, \mathbb{G}, \delta) = \max\{k : \mathop{\mathbf{Pr}}_{G \in \mathbb{G}}[\mathsf{cc}(S, G) \geq k] \geq 1 - \delta\}$. Fixing δ to be some negligible function of N, we can define $\mathsf{cc}_\delta(\mathbb{G}) = \min_S \mathsf{cc}(S, \mathbb{G}, \delta)$.

3 General Attack Against k-Restricted Graphs

In this section, we describe a general attack against k-restricted graphs. We show that the attack incurs cost $o(N^2)$ for $k = o\left(N^{1/\log\log N}\right)$, proving that there is no maximally memory hard k-restricted graph for small k.

We first require the following formulation of Valiant's Lemma, which shows the existence of a subroutine $\mathsf{Valiant}(G, e, d)$ to find a depth-reducing set S of size at most e within a graph G, for $e = \frac{\eta\delta N}{\log(N) - \eta}$ and $d = \frac{N}{2^\eta}$, where $\eta > 0$.

▶ **Lemma 9** (Valiant's Lemma). *[32] For any DAG $G = (V, E)$ with N nodes, indegree δ, and $\eta > 0$, there exists an efficient algorithm $\mathsf{Valiant}(G, e, d)$ to compute a set S of size $|S| \leq e := \frac{\eta\delta N}{\log(N) - \eta}$ such that $\mathsf{depth}(G - S) \leq d := \frac{N}{2^\eta}$.*

The high level intuition of the generic attack is as follows. By Valiant's Lemma (Lemma 9), $G \sim \mathbb{G}$ is (e, d)-reducible for $e = \frac{\eta\delta N}{\log(N) - \eta}$ and $d = \frac{N}{2^\eta}$. We will construct a dynamic pebbling strategy $\mathcal{A}$ that for all times t, maintains a depth-reducing set S_t such that $\mathsf{depth}(G_t - S_t) \leq d$, where G_t is the portion of G revealed after running $\mathcal{A}$ for time t, $G_t = G_{\leq i}$ for $i = 1 + \max \bigcup_{j=1}^t P_j$, where each $P_j \subseteq [N]$ represents the set of pebbled nodes during round j. Observe that for any i, $G_{\leq i}$ is (e, d)-reducible and hence G_t is also (e, d)-reducible for all times t. Thus, the depth reducing set S_t has size at most e for all times t and can be computed by a subroutine $\mathsf{Valiant}$, by Lemma 9. We now describe how $\mathcal{A}$ maintains this depth-reducing set through a series of *light phases* and *balloon phases*.

We first set a parameter g that we will eventually optimize. The goal of each light phase i is to pebble the next g nodes that have yet to be revealed. That is, if x_i is the largest node for which $\mathcal{A}$ has placed a pebble at some point prior to light phase i, then the goal of light phase i is to pebble the interval $[x_i + 1, x_i + g]$. To begin light phase i at some time t_i, we require that $(\mathsf{PotentialParents}([x_i + 1, x_i + g]) \cup S_{t_i}) \subseteq P_{t_i}$ for some depth-reducing set S_{t_i} of size at most e, such that $\mathsf{depth}(G_{t_i} - S_{t_i}) \leq d$. Once this pre-condition is met, then light phase i simply takes g steps to pebble $[x_i + 1, x_i + g]$, since pebbles are already placed on $\mathsf{PotentialParents}([x_i + 1, x_i + g])$. Hence, the post-condition of light phase i at some time u_i is pebbles on the node $x_i + g$ and some depth-reducing set S_{u_i} of size at most e, such that $\mathsf{depth}(G_{u_i} - S_{u_i}) \leq d$.

The goal of each balloon phase i is to place pebbles on all revealed nodes of the graph, to meet the pre-condition of light phase $i + 1$. To begin balloon phase i at some time r_i, we first have a necessary pre-condition that pebbles are placed on some depth-reducing set S_{r_i} of size at most e such that $\mathsf{depth}(G_{r_i} - S_{r_i}) \leq d$. Once this pre-condition is met, then balloon phase i simply takes d steps to pebble the entire graph G_{r_i}, meeting the post-condition of balloon phase i.

We now formally prove the cumulative memory complexity of the attack in Algorithm 1.

▶ **Theorem 10.** *Let $\mathbb{G}$ be any family of k-restricted dynamic graphs with constant* $\mathsf{indeg}(\mathbb{G})$. *Then*

$$\mathsf{cc}(\mathbb{G}) = \mathcal{O}\left(\frac{N^2}{\log\log N} + N^{2-1/2\log\log N}\sqrt{k^{1-1/\log\log N}}\right).$$

Proof. We analyze the cost of the pebbling strategy of Algorithm 1. Since G is drawn from a distribution of k-restricted dynamic graphs, then for any node x_i, $r(x_i)$ must be one of at most k labels. Thus for any consecutive g nodes, $|\mathsf{PotentialParents}([x_i + 1, x_i + g])| \leq gk$. Hence it suffices to keep gk pebbles on the set of potential parents $\mathsf{PotentialParents}([x_i + 1, x_i + g])$ to pebble the interval $[x_i + 1, x_i + g]$, as well as a depth-reducing set of size at most e, for each of the g steps during light phase i. On the other hand, balloon phase i takes d steps, each of which trivially contains at most N pebbles.

$\mathcal{A}$ proceeds using $\frac{N}{g}$ total rounds of light and balloon phases, by pebbling g consecutive nodes at a time. Therefore, the total cost of the attack is $\mathcal{O}\left(Ngk + Ne + \frac{N}{g} \cdot dN\right)$, where the first and second terms originate from the cost of the light phases and the third term results from the cost of the balloon phases. Since we set $e = \frac{\eta\delta N}{\log(N)-\eta}$ and $d = \frac{N}{2^\eta}$ from Valiant's Lemma (Lemma 9) so that the total cost is $\mathcal{O}\left(\frac{\eta\delta N^2}{\log(N)-\eta} + Ngk + \frac{N^3}{2^\eta g}\right)$. By setting $g = \frac{N}{\sqrt{k2^\eta}}$, the total cost is $\mathcal{O}\left(\frac{N^2\eta\delta}{\log N-\eta} + N^2\sqrt{\frac{k}{2^\eta}}\right)$. Finally, by setting $\eta = \frac{\log k+\log N-\log\delta}{\log\log N}$, the total cost is $\mathcal{O}\left(\frac{N^2}{\log\log N} + N^{2-1/2\log\log N}\sqrt{k^{1-1/\log\log N}}\right)$. ◀

Note that if $k = o(N^{1/\log\log N})$, then $\mathsf{cc}(\mathbb{G}) = o(N^2)$.

▶ **Corollary 11.** *Let $\mathbb{G}$ be any k-restricted dynamic graph with $k = o(N^{1/\log\log N})$ and constant indegree. Then* $\mathsf{cc}(\mathbb{G}) = o(N^2)$.

Algorithm 1 Generic pebbling strategy against dynamic DAG G.

Input: An integer i, an initial pebbling configuration $P_0^i \subseteq [i]$ with $i \in P_0^i$, a partial graph $G_{\leq i+1}$, and parameters d, e, g.
Output: A legal pebbling of $G_{\leq i+1}$.

invariant $\leftarrow$ True
if $i \pmod g \equiv 0$ and $\mathsf{depth}(G_{\leq i+1} - P_0^i) > d$ **then**
    invariant $\leftarrow$ False
else if $\mathsf{depth}(G_{\leq i+1} - P_0^i) > d$ or $\{i\} \cup \mathsf{PotentialParents}([i+1, i+g]) \not\subseteq P_0^i$ **then**
    invariant $\leftarrow$ False
if invariant **then** ▷ If pre-conditions met.
    if $i \pmod g \equiv 0$ **then** ▷ Balloon phase
        for $j = 1$ to $j = d$ **do**
            $P_j^i = P_{j-1}^i \cup D_j$, where D_j are the nodes at depth d from P_0^i.
        $P_{d+1}^i = \mathsf{Valiant}(G_{\leq i}, e, d) \cup \mathsf{PotentialParents}([i+1, i+g])$. ▷ See Lemma 9
    else ▷ Light phase
        $P_1^i = P_0^i \cup \{i+1\}$.
else ▷ If pre-conditions not met.
    for $j = 1$ to $j = i+1$ **do**
        $P_j^i = P_{j-1}^i \cup \{j\}$.

4 k-Restricted Graphs with high CMC

In this section, we describe a construction of k-restricted graphs with high cumulatively memory complexity that builds into our ultimate ciMHF implementation. We first describe the block partition extension gadget, which requires an input graph G and outputs a family of k-restricted dynamic graphs. However, naïvely choosing the input graph G does not yield a construction with high CMC.

Intuitively, the block partition extension gadget takes the last N nodes of G and partitions them into $\frac{N}{k}$ blocks of k nodes each. The gadget then creates N more nodes in a path, such that a parent $r(j)$ of node j in this path is drawn uniformly at random from block i, where $i = j \pmod{\frac{N}{k}}$. The intuition is that by drawing parents uniformly at random from each block in round robin fashion, we encourage an algorithm to keep $\Omega(N)$ nodes on the graph for $\Omega(N)$ steps. Of course, the graph could always maintain $o(n)$ pebbles on the graph and repebble when necessary, but we can discourage this strategy by making the repebbling procedure as expensive as possible.

A first attempt would be to choose a highly depth-robust graph G, such as a grates graph, which informally has long paths of length $\Omega(N^{1-\epsilon})$ for any constant $0 < \epsilon < 1$, even when $\Omega(N)$ nodes are removed from G. Thus if an algorithm does not maintain $\Omega(N)$ pebbles on the graph, the repebbling strategy costs at least $\Omega(N^{2-\epsilon})$. Although this is a good start, this does not quite match the $\Omega(N^2)$ CMC of various dMHFs. We defer full discussion of how to increase the CMC to $\Omega(N^2)$ to later in this section.

Instead, we first define a specific way to obtain a k-restricted dynamic graph given a graph G with N nodes and a parameter k.

▶ **Definition 12** (Block Partition Extension). *Given a DAG $G = (V = [\alpha N], E)$ with αN nodes containing a set of $O = [(\alpha-1)N + 1, \alpha N]$ output nodes of size N and a parameter k, let $O_i = [(\alpha-1)N + 1 + ik, (\alpha-1)N + (i+1)k]$ for $i \in \left[\frac{N}{k}\right]$ so that $\{O_i\}$ forms a partition of O. We define the* block partition extension *of G, denoted $\mathsf{BlockPartition}_k(G)$, as*

a distribution of graphs $\mathbb{G}_{G,k}$*. Each graph* G' *sampled from* $\mathbb{G}$ *has vertices* $V' = [(\alpha+1)N]$ *and edges* $E' = E \cup F$*, where* F *is defined as the edges* $(i-1, i)$ *and* $(r(i), i)$ *for each* $i \in [\alpha N + 1, (\alpha+1)N]$*, where* $r(i)$ *is drawn uniformly at random from* $O_{i \mod \frac{N}{k}}$*.*

An example of the block partition extension is given in Figure 1.

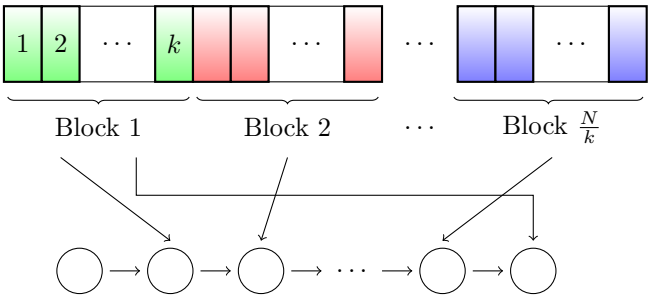


Figure 1 Parent $r(i)$ is drawn uniformly at random from the nodes partitioned to each block.

Our ultimate construction also requires the use of superconcentrator graphs, defined as follows:

▶ **Definition 13.** *A graph* G *with* $\mathcal{O}(N)$ *vertices is a* superconcentrator *if there exists an input set* I *and an output set* O *with* $|I| = |O| = N$ *such that for all* $S_1 \subseteq I, S_2 \subseteq O$ *with* $|S_1| = |S_2| = k$*, there are* k *node disjoint paths from* S_1 *to* S_2*.*

It is known that there exists superconcentrators with $|I| = |O| = N$, constant indegree and $\mathcal{O}(N)$ total nodes [28, 26]. We now show that a set Y, which contains more nodes than a set S of removed nodes, has at least $N - |S|$ ancestors in $G - S$.

▶ **Lemma 14.** *Given a superconcentrator* G *with* N *input nodes* I *and* N *output nodes* O*, let* S *and* $Y \subseteq O$ *be sets of nodes with* $|S| < |Y|$*. Then* $|I \cap \mathsf{ancestors}_{G-S}(Y)| \geq N - |S|$*.*

Proof. Let $X \subseteq I$ be the last $|Y|$ nodes of I. Since G is a superconcentrator, then G contains at least $|Y|$ node disjoint paths between X and Y. Since $|S| < |Y|$, then one of these paths from X to Y that does not intersect S. Thus, X contains some ancestor of Y in $G - S$ and in fact by considering the paths associated with decreasing order of nodes in X, it follows that $|I \cap \mathsf{ancestors}_{G-S}(Y)| \geq N - |S|$. ◀

We require the use of grates graphs $\{\mathsf{grates}_{N,\epsilon}\}_{N=1}^{\infty}$ [30]. For each constant $\epsilon > 0$ and each $N \geq 1$ the graph $\mathsf{grates}_{N,\epsilon} = (V_N, E_{N,\epsilon})$ has $\mathcal{O}(N)$ nodes and constant indegree $\mathsf{indeg}(\mathsf{grates}_{N,\epsilon}) = \mathcal{O}(1)$. Moreover, the graph $\mathsf{grates}_{N,\epsilon}$ contains source nodes $I_N \subset V_N$ and N sinks $O_N \subset V_N$. Given a set $S \subseteq V_N$ of deleted nodes we say that an output node $y \in O_N$ is c-good with respect to S if $\left|I_N \cap \mathsf{ancestors}_{\mathsf{grates}_{N,\epsilon}-S}(y)\right| \geq cN$ i.e., for at least cN input nodes $x \in I_N$ the graph $\mathsf{grates}_{N,\epsilon} - S$ still contains a path from x to y. The grates graph contains several properties summarized below.

▶ **Theorem 15.** *[30] For each* $\epsilon > 0$ *there exist constants* $\gamma, c > 0$ *such that for all* $N \geq 1$ *the graph* $\mathsf{grates}_{N,\epsilon}$ *is* $(\gamma N, cN^{1-\epsilon})$*-depth robust. Furthermore, for each set* $S \subseteq V_N$ *of size* $|S| \leq \gamma N$ *at least* cN *output nodes are still* c*-good with respect to* S*. Formally,*

$$|\{x \in O \ : \ |I_N \cap \mathsf{ancestors}_{G-S}(x)| \geq cN\}| \geq cN \ .$$

We require the use of graph overlays, defined as follows:

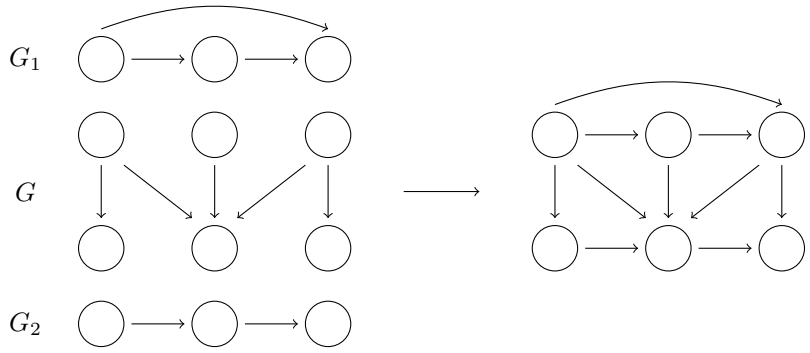


Figure 2 An example of a graph overlay $\mathsf{overlay}(G_1, G, G_2)$.

▶ **Definition 16** (Graph overlays). *Given a DAG $H = (V = [N], E)$ with sources $I = \{1, \ldots, n_1\}$ and sinks $O = \{N - n_2 + 1, \ldots, N\}$, a DAG $G_1 = (V_1 = [n_1], E_1)$, and a DAG $G_2 = (V_2 = [n_2], E_2)$, we define:*

(1) *the* graph overlay $G' = \mathsf{overlay}(G_1, H, G_2)$ *by* $G' = ([N], E')$, *where* $(i, j) \in E'$ *if and only if* $(i, j) \in E$ *or* $(i, j) \in E_1$ *or* $(i + N - n_2, j + N - n_2) \in E_2$

(2) *the* superconcentrator overlay *of an N node DAG G by* $\mathsf{superconc}(G) = \mathsf{overlay}(G, \mathsf{SC}_N, L_N)$, *where SC_N is a superconcentrator with N input (sources) and output (sinks) nodes and L_N is the line graph of N nodes*

(3) *the* grates overlay *of an N node DAG G by* $\mathsf{grates}_\epsilon(G) = \mathsf{overlay}(G, \mathsf{grates}_{N,\epsilon}, L_N)$.

An example of a graph overlay is displayed in Figure 2.

We describe a preliminary attempt at a ciMHF construction in Figure 3. At a high level, the construction consists of four components. The first component is a grates graph G_1 with N nodes. The second component is a superconcentrator overlay with $\mathcal{O}(N)$ nodes, including N input nodes and N output nodes, so that $G_2 = \mathsf{superconc}(G_1)$. The third component consists of a grates overlay with $\mathcal{O}(N)$ nodes including N output nodes, so that $G_3 = \mathsf{grates}_\epsilon(G_2)$. The N output nodes of G_3 are partitioned into $\frac{N}{k}$ blocks, each with k nodes, in preparation for a block partition extension in the final component. Namely, the fourth component consists of a k-restricted graph with N nodes, so that $G_4 = \mathsf{BlockPartition}_k(G_3)$.

Sampling Algorithm, for $k = \Omega(N^\epsilon)$:
(1) $G_1 = \mathsf{grates}_{N,\epsilon}$
(2) $G_2 = \mathsf{superconc}(G_1)$
(3) $G_3 = \mathsf{grates}_\epsilon(G_2)$
(4) $G_4 \sim \mathsf{BlockPartition}_k(G_3)$

Figure 3 First attempt at ciMHF. Each parent $r(i)$ is randomly chosen from the labels in specific block corresponding to i.

The intuition for the $\Omega(N^2)$ cumulative pebbling complexity is as follows. Suppose there exists a time t_{bad} with a "small" number of pebbles on the graph. Then with high probability, walking a pebble $s = \frac{N}{4k}$ steps on the final layer of the graph will require some number of output nodes of the grates graph to be repebbled. Again with high probability, repebbling one of these output nodes requires a large number of input nodes of the grates graph to be repebbled. These input nodes are the output nodes of the superconcentrator at the second layer. The superconcentrator property then implies that $\Omega(N)$ nodes of the grates graph

on the first layer will need to be repebbled. For a grates graph that is $(\Omega(N), \Omega(N^{1-\epsilon}))$-depth robust, this cost is at least $\Omega(N^{2-\epsilon})$ every s steps. Thus, the total cost is at least $\min\left(\Omega(N^2), k\Omega(N^{2-\epsilon})\right)$, which is just $\Omega(N^2)$ for $k = \Omega(N^\epsilon)$.

We now show that our construction in Figure 3 has cumulative memory complexity $\Omega(N^2)$.

▶ **Theorem 17.** *Let G be drawn from the distribution of k-restricted graphs in Figure 3, for $k = \Omega(N^\epsilon)$. There exist constants $c_1 > 0$ and $c_2 \in (0,1)$ such that for any dynamic pebbling strategy S,*

$$\Pr_{\mathbb{G}}\left[\mathsf{cc}(S,G) > c_1 N^2\right] \geq 1 - c_2^{N/k}.$$

Proof. Let αN be the total number of nodes in G_3 so that the total number of nodes in G is $(\alpha+1)N$. Let $x, y \in (0,1)$ be constants such that the grates graph G_1 is $(xN, yN^{1-\epsilon})$-depth robust. By Theorem 15, there exist constants $0 < c < \frac{x}{2}$ and $0 < \gamma < c$ such that for any set S with $|S| \leq \gamma N$, at least cN nodes in the output nodes of G_3 are c-good with respect to S. For each node i, let t_i be the first time that node i is pebbled. Suppose there exists a time t_{bad} with $t_i \leq t_{\mathsf{bad}} < t_{i+1}$ such that there are $|P_{t_{\mathsf{bad}}}| < \frac{\gamma N}{4}$ pebbles on the graph.

For a node j in the output set $[(\alpha-1)N, \alpha N]$ of G_3, we call an index j a *costly index* if j is c-good with respect to $P_{t_{\mathsf{bad}}}$ and let COSTLY be the set of costly indices. Note that if a node $i \in \mathsf{COSTLY}$, then by definition $i \notin P_{t_{\mathsf{bad}}}$. By Theorem 15 and the observation that $|P_{t_{\mathsf{bad}}}| < \frac{\gamma N}{4}$, there are at least cN nodes in the output set of G_3 are c-good with respect to $P_{t_{\mathsf{bad}}}$, i.e., $|\mathsf{COSTLY}| > cN$. Then for $s := \frac{N}{k}$, we call $j \in [i, i+s]$ a *missed costly index* if $r(j) \notin P_{t_{\mathsf{bad}}}$ and let $r(j) \in \mathsf{COSTLY}$.

For each $j \in \left[\frac{N}{k}\right]$, let c_j be the number of costly indices in block j of the output set of G_3, i.e., $c_j := |\mathsf{COSTLY} \cap [(\alpha-1)N + (j-1)k + 1, (\alpha-1)N + jk]|$. Since the parents of $[i, i+s]$ are exactly one random node from each of the $\frac{N}{k}$ blocks, then the probability p that no parent of $[i, i+s]$ is a missed costly index is

$$p := \prod_{j=1}^{N/k}\left(1 - \frac{c_j}{k}\right) \leq \left(\frac{k}{N}\sum_{j=1}^{N/k}\left(1 - \frac{c_j}{k}\right)\right)^{N/k},$$

where the inequality holds by the Arithmetic Mean-Geometric Mean Inequality. Since $\sum c_j = cN$, then

$$p \leq \left(1 - \frac{k}{N}\sum_{j=1}^{N/k}\frac{c_j}{k}\right)^{N/k} \leq (1-c)^{N/k}.$$

Thus with high probability, there will be some missed costly index.

By Lemma 14 and the definition of c-good, any missed costly index requires cN nodes in the input set of G_3 to be repebbled. Since the input set of G_3 is connected by a superconcentrator to the output set of G_1, the cN nodes in the input set of G_3 that need to be repebbled have at least $N - cN$ ancestors in the output set of G_1. Thus, at least $N - cN - |P_{t_{\mathsf{bad}}}|$ nodes in G_1 must be repebbled.

Because G_1 is $(xN, yN^{1-\epsilon})$-depth robust, then $G_1 - S$ is $(xN - |S|, yN^{1-\epsilon})$-depth robust for any set S. Moreover, the cost to pebble $G_1 - S$ is at least $(xN - |S|)(yN^{1-\epsilon})$. In particular, if $G_1 - S$ is the set of nodes in G_1 must be repebbled, then it costs at least $(xN - cN - |P_{t_{\mathsf{bad}}}|)(yN^{1-\epsilon})$ to repebble $G_1 - S$. Since $c < \frac{x}{2}$ and $|P_{t_{\mathsf{bad}}}| < \frac{\gamma N}{4} < \frac{cN}{4} < \frac{xN}{8}$, then the cost is at least $\frac{3xy}{8}N^{2-\epsilon}$.

Hence to pebble an interval $[i, i+s]$ with $s = \frac{N}{k}$, either $\frac{\gamma N}{4}$ pebbles are kept on the graph for all $s = \frac{N}{k}$ steps or if we at any point in time $j \in [i, i+s]$ we have $|P_j| \leq \gamma N/4$ then (whp) an the pebbling algorithm incurs cost $\frac{3xy}{8}N^{2-\epsilon}$ to repebble the graph during the next s steps $[j, j+s]$. By partitioning the last N nodes of the graph G into k disjoint intervals of length $\frac{N}{k}$, it follows that the total cost is at least $\min\left(\frac{\gamma N^2}{4}, \frac{3}{16}xykN^{2-\epsilon}\right)$. Thus for $k = \Omega(N^{\epsilon})$, the total cost is $\Omega(N^2)$ with high probability. ◀

▶ **Corollary 18.** *Let G be drawn from the distribution of k-restricted graphs in Figure 3, for $k = \Omega(N^{\epsilon})$. Then $\mathsf{cc}(G) = \Omega(N^2)$.*

5 k-Restricted Graphs: Amenable to Shuffling

In this section, we introduce a useful property for certain dynamic graphs: amenable to shuffling. In Section 6, we will describe computationally data-independent evaluation algorithms for evaluating memory hard function based on dynamic graphs that are amenable to shuffling.

5.1 Characterization of Dynamic Graphs Amenable to Shuffling

We first describe the properties of dynamic graphs that are amenable to shuffling. Recall that for a node i, we define $\mathsf{PotentialParents}(i)$ to be set Y_i of minimal size such that $\mathbf{Pr}[r(i) \in Y_i] = 1$, where $r(i) < i - 1$ is randomly chosen so that the directed edge $(r(i), i)$ is in the dynamic graph.

▶ **Definition 19** (Amenable to Shuffling)**.** *Let G be a DAG with αN nodes for some constant $\alpha > 1$ and let L be the last N nodes of G. Suppose that L can be partitioned into $\frac{N}{k}$ groups $G_1, \ldots, G_{\frac{N}{k}}$ such that*

(1) *Uniform Size of Groups: $|G_i| = k$ for all $i \in \left[\frac{N}{k}\right]$.*
(2) *Large Number of Potential Parents: For each $v \in L$, $|\mathsf{PotentialParents}(u)| = k$.*
(3) *Potential Parents not in L: For each $v \in L$, $\mathsf{PotentialParents}(u) \subseteq [(\alpha - 1)N]$*
(4) *Same Potential Parents for Each Group: For all $i \in \left[\frac{N}{k}\right]$ and $u, v \in G_i$, $\mathsf{PotentialParents}(u) = \mathsf{PotentialParents}(v)$.*
(5) *Different Potential Parents for Different Groups: For all $i, j \in \left[\frac{N}{k}\right]$ with $i \neq j$, let $u \in G_i$ and $v \in G_j$. Then $\mathsf{PotentialParents}(u) \cap \mathsf{PotentialParents}(v) = \emptyset$.*
(6) *No Collision for Parents: For each $i \in \left[\frac{N}{k}\right]$, define the event UNIQUE_i to be the event that $r(u) \neq r(v)$ for all $u, v \in G_i$. Then $\mathbf{Pr}[\mathsf{UNIQUE}_i] = 1$ for all $i \in \left[\frac{N}{k}\right]$.*
(7) *Data-Independency: The subgraph induced by the first $(\alpha - 1)N$ nodes is a static graph.*

Then we call G amenable to shuffling.

We shall show in Theorem 24 in Section 6 that dynamic graphs that are amenable to shuffling can be used for memory hard functions with computationally data-independent evaluation algorithms. We now describe a version of Figure 3 that is amenable to shuffling.

5.2 Version of Construction Amenable to Shuffling

To ensure that there does not exist $i \neq j$ such that $r(i) = r(j)$, we slightly modify the construction of block partition extensions to the concept of a collision-resistant block partition extension. For the sake of presentation, note that we use $2N$ output nodes in G in the following definition.

▶ **Definition 20** (Collision-Resistant Block Partition Extension). *Given a DAG $G = (V = [\alpha N], E)$ with αN nodes containing a set of $O = [(\alpha-2)N+1, \alpha N]$ output nodes of size $2N$ and a parameter k, let $O_i = [(\alpha-2)N+1+2ik, (\alpha-2)N+2(i+1)k]$ for $i \in \left[\frac{N}{k}\right]$ so that $\{O_i\}$ forms a partition of O. We define the* collision-resistant block partition extension *of G, denoted $\mathsf{CR}-\mathsf{BlockPartition}_k(G)$, as a distribution of graphs $\mathbb{G}_{G,k}$. Each graph G' sampled from $\mathbb{G}$ has vertices $V' = [(\alpha+1)N]$ and edges $E' = E \cup F$, where F is defined as the edges $(i-1, i)$ and $(r(i), i)$ for each $i \in [\alpha N+1, (\alpha+1)N]$, where $r(i)$ is defined as follows:*

(1) *Let* Enc *be the family of all permutations of $[2k]$, so that for each fixed j,*

$$\{\mathsf{Enc}(j,\ell)\}_{\ell\in[2k]} = [2k]\,.$$

(2) *For each $i \in [\alpha N+1, (\alpha+1)N]$, let $j = i \mod \frac{N}{k}$ and define $1 \le p \le k$ to be the unique integer such that $i = \frac{N}{k}(p-1) + \alpha N i + j$. Then we define $r(i) = (\alpha-2)N + 1 + 2jk + \mathsf{Enc}(x \circ j, p)$, so that $r(i) \in O_j$.*

Observe that the collision-resistant block partition extension is a gadget that yields a dMHF, since the parent function $r(i)$ has the key $x \circ j$ to its permutation function Enc. Hence, the underlying dynamic graph differs across different input values x.

Then our construction of the ciMHF appears in Figure 4 and Figure 5. As before, the construction consists of four layers. The first layer consists of a grates graph with $2N$ nodes, $G_1 = \mathsf{grates}_{2N,\epsilon}$. The second layer consists of a superconcentrator overlay with $\mathcal{O}(N)$ nodes with $2N$ input nodes and $2N$ output nodes, so that $G_2 = \mathsf{superconc}(G_1)$. The third layer consists of a grates overlay with $\mathcal{O}(N)$ nodes including $2N$ output nodes, so that $G_3 = \mathsf{grates}_{\epsilon}(G_2)$. The $2N$ output nodes of G_3 are partitioned into $\frac{N}{k}$ blocks, each with $2k$ nodes, which allows the final layer to be a $2k$-restricted grpah. In particular, the fourth layer uses a collision-free block partition extension rather than the block partition extension of Figure 3.

Sampling algorithm, for $k = \Omega(N^{\epsilon})$:
(1) $G_1 = \mathsf{grates}_{2N,\epsilon}$
(2) $G_2 = \mathsf{superconc}(G_1)$
(3) $G_3 = \mathsf{grates}_{\epsilon}(G_2)$
(4) $G_4 \sim \mathsf{CR}-\mathsf{BlockPartition}_k(G_3)$

Figure 4 Second attempt at ciMHF. Each parent $r(i)$ is chosen by a permtuation of the labels in specific block corresponding to i. The underlying graph is visualized in Figure 5.

We now show that the construction of Figure 4 has cumulative cost $\Omega(N^2)$ with high probability. The proof is almost verbatim to Theorem 17 except that the graph overlays now have $2N$ input and output nodes.

▶ **Theorem 21.** *Let $0 < \epsilon < 1$ be a constant and $k = \Omega(N^{\epsilon})$. Let $\mathbb{G}$ be drawn from the distribution of $2k$-restricted graphs in Figure 4. There exist constants $c_1 > 0$ and $c_2 \in (0,1)$ such that for any dynamic pebbling strategy S,*

$$\Pr_{G\in\mathbb{G}}\left[\mathsf{cc}(S,G) > c_1 N^2\right] \ge 1 - c_2^{N/k}.$$

▶ **Corollary 22.** *Let $\mathbb{G}$ be drawn from the distribution of $2k$-restricted graphs in Figure 4, for $k = \Omega(N^{\epsilon})$. Then $\mathsf{cc}(\mathbb{G}) = \Omega(N^2)$.*

Finally, we observe that the construction of Figure 4 is amenable to shuffling since it satisfies the properties of Definition 19.

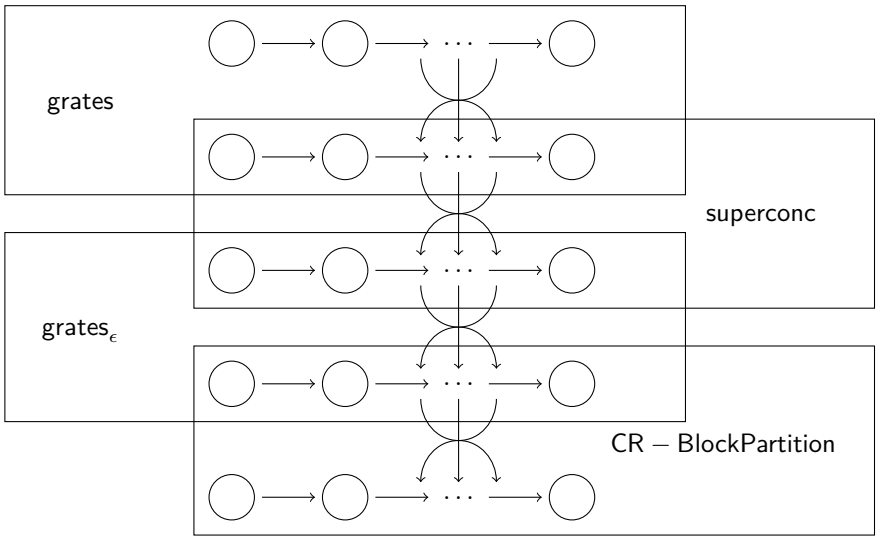


Figure 5 Final construction of Figure 4.

6 Implementation of ciMHF

In this section, we describe how to implement our construction in a way that is computationally data-independent. We first formalize the notion of computationally data-independent and then describe the system model we utilize.

6.1 Computationally Data-Independent MHF (ciMHF) and Systems Model

We define the security of a computationally data-independent memory hard function in terms of the following game: a side-channel attacker $\mathcal{A}$ selects two inputs x_0, x_1 and sends these inputs to an honest party $\mathcal{H}$. We first require the following definition of leakage patterns.

Leakage Pattern

We define the leakage pattern of an evaluation algorithm MHF.Eval by the sequence of request and store instructions made in each round. Specifically, in each round r, an attacker can observe from the leakage pattern the blocks of memory to be loaded into cache, as requested by MHF.Eval. Let $i = (i_1, \ldots, i_m)$ be the sequence of locations of all blocks requested by MHF.Eval in a particular round r through some command load(i). If i is completely contained in cache, then no events will be observed by the attacker. Otherwise, if i is not completely contained in cache, we use request$_r$ to denote the locations of the blocks in memory, as well as their sizes, requested by MHF.Eval in round r. Similarly, we use store$_r$ to denote the locations of the blocks, as well as their sizes, stored into memory by MHF.Eval in round r. We do not allow the attacker to observe the contents of the requested or stored blocks. Formally, the leakage pattern LP is the information $\{(\text{request}_r, \text{store}_r)\}_{r=1}^{t}$ and is dependent on the algorithm MHF.Eval, random oracle H, internal randomness R, and input value x.

Computationally Data-Independency Game

$\mathcal{H}$ runs a (randomized) evaluation algorithm MHF.Eval on both inputs x_0 and x_1, yielding two leakage patterns LP_0 and LP_1, where LP_i for $i \in \{0,1\}$ depends on both the input x_i and the random coins selected during the execution of MHF.Eval. $\mathcal{H}$ then picks a random challenge bit $b \in \{0,1\}$ and sends $\mathsf{LP}_b, \mathsf{LP}_{1-b}$ to $\mathcal{A}$ to simulate a side-channel. The goal of $\mathcal{A}$ is to predict b i.e., match each input with the corresponding leakage pattern. For a secure ciMHF we guarantee that *any* PPT side-channel attacker $\mathcal{A}$ wins the game with only negligible advantage over random guessing.

Formally, the game consists of three phases **setup**, **challenge**, and **guess**, which are described as follows.

Data independency game for ciMHF:

setup In this phase, $\mathcal{A}$ selects the security parameter λ and two challenge messages x_0 and x_1 and sends them to $\mathcal{H}$. Here we assume without loss of generality that the runtime of the evaluation algorithm MHF.Eval on x_0 and x_1 are the same.

challenge In this phase, $\mathcal{H}$ selects a random bit $b \in \{0,1\}$ and random coins $R_0, R_1 \in \{0,1\}^\lambda$ uniformly at random and then samples $\mathsf{lp}_0 \leftarrow \mathsf{LP}(\mathsf{MHF.Eval}(x_0; R_0))$ and $\mathsf{lp}_1 \leftarrow \mathsf{LP}(\mathsf{MHF.Eval}(x_1; R_1))$. $\mathcal{H}$ sends the ordered pair $(\mathsf{lp}_b, \mathsf{lp}_{1-b})$ to $\mathcal{A}$.

guess After receiving $(\mathsf{lp}_b, \mathsf{lp}_{1-b})$, the adversary $\mathcal{A}$ outputs b' as a guess for b. The adversary wins the game if $b = b'$.

The advantage of the adversary to win the game of computationally data independency of the given MHF is defined as

$$Adv^{\mathsf{ind-lp-iMHF}}_{\mathcal{A},\mathsf{MHF}} = \left| \frac{1}{2} - \Pr[\mathcal{A}(x_0, x_1, \mathsf{lp}_b, \mathsf{lp}_{1-b}) = b' : b = b')] \right|,$$

where $\mathsf{lp}_i = \mathsf{LP}(x_i; \mathsf{MHF.Eval}(x_i; R_i))$.

▶ **Definition 23** (Computational data independency)**.** *An evaluation algorithm* MHF.Eval *is computationally data independent if for all non-uniform circuits $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda \in \mathbb{N}}$, there is a negligible function* $\mathsf{negl}(\cdot)$ *such that* $Adv^{\mathsf{ind-lp-iMHF}}_{\mathcal{A},\mathsf{MHF}} < \mathsf{negl}(\lambda)$.

With the proper random coins, a memory-hard function with an evaluation algorithm that satisfies the above definition reveals only a negligible amount of information through its leakage patterns, and we thus call such a function a computationally data-independent memory hard function.

On the Definition of Computational Data Independency

In section Section 6.2.2 we show that the definition is equivalent to a multi round version of the game in which the attacker can adaptively select the challenge $x_{i,0}, x_{i,1}$ in each round $i \leq r$ after observing $\mathsf{lp}_{i-1,b}, \mathsf{lp}_{i-1,1-b}$ – the memory access patterns from the last round. We also prove that the two security notions are asymptotically equivalent when the attacker runs in polynomial time – in terms of concrete security parameters we lose a factor of r (number of challenge rounds) in the reduction.

We are primarily motivated by the password hashing application where the inputs x_0 and x_1 come from a small domain, as user selected passwords tend to have low entropy [18]. In practice it is reasonable to assume that r is polynomial i.e., if the user only authenticates

poly(λ) times then there are at most $r = \text{poly}(\lambda)$ memory access patterns for the attacker to observe. Assuming that the input domain has size poly(λ) a brute-force attacker cannot use the leaked memory access pattern on input x to eliminate any candidate password x' with high probability, otherwise the attacker could have used the pair x and x' to win the data independency game.

However, in settings where the input domain is very large and r is super-polynomial it will be better to adopt a concrete security definition (see Section 6.2.2). The asymptotic definition in Definition 23 does not definitively rule out the possibility that an attacker can substantially narrow the search space after many (super-polynomial) side channel attacks. For example, suppose that the attacker gets to observe $\mathsf{lp}_i \leftarrow \mathsf{LP}(\mathsf{MHF.Eval}(x; R_i))$ for $i = 1, \ldots, 2^\lambda$, i.e., 2^λ independent evaluations of MHF on secret input x. Supposing that $Adv^{\mathsf{ind-lp-iMHF}}_{\mathcal{A},\mathsf{MHF}} = 2^{-\lambda}$ and that the input domain for MHF has size $2^{2\lambda}$, it is possible that each lp_i allows the attacker to eliminate a random subset of $Adv^{\mathsf{ind-lp-iMHF}}_{\mathcal{A},\mathsf{MHF}} \times 2^{2\lambda} = 2^\lambda$ candidate inputs, allowing the attacker to find x after just $\mathcal{O}\left(\lambda 2^\lambda\right)$ examples. However, in practice it will usually be reasonable to assume that the attacker gets to observe lp_i a polynomial number of times i.e., the honest party will execute $\mathsf{LP}(\mathsf{MHF.Eval}(x; R_i))$ at most poly(λ) times.

Memory Architecture Assumptions

We consider a tiered random access memory architecture with main memory (RAM) and working memory (cache). We assume that main memory (RAM) is a shared resource with other untrusted processes, each of which have their own cache. Although the operating system kernel will enforce memory separation, i.e., only our program has some region of memory and that other processes cannot read/write to this block, it is also possible that an untrusted process will be able to infer the memory address of read/write operations in RAM (due to side-channel effects).

Formally, the system allows programs access to two operations $\mathsf{Write}(i, x)$, which takes an address i within the memory allocated to the program and writes the value x at address i, and $\mathsf{Read}(i)$ which loads the data at location i. When an operation requests memory at location i, there are two possible outcomes. Either the data item is already in cache or the data item is not in cache. In the second case, the location of the item in memory is revealed through the leakage pattern. Hence, the leakage pattern is either $\perp$, if the data item is already in cache, or i, if the data item is not in cache.

Cache Replacement Policy

We now show that our implementation of the dynamic pebbling construction with cumulative memory cost $\Omega(N^2)$ is computationally data-independent. In particular, we provide an evaluation algorithm whose leakage pattern is computationally indistinguishable under each of the following cache replacement policies:

Least recently used (LRU) This policy tracks the most recent time each item in cache was used and discards the least recently used items first when cache is full and items need to be replaced.

First in first out (FIFO) This policy evicts the first item that was loaded into cache, ignoring how recent or how often it has been accessed.

6.2 ciMHF Implementation

Recall from the definition of a graph labeling in Definition 6 that given a function H and a distribution of dynamic graphs $\mathbb{G}$, the goal is to compute $f_{\mathbb{G},H}(x)$ for some input x, which is equivalent to $f_{G,H}(x)$ once the graph G has been determined by the choice of H and x. In this section, we describe a computationally data-independent implementation of the construction of Section 5.2.

We implement the first three layers as data-independent components. Namely, the grates graph G_1, its superconcentrator overlay G_2 and the subsequent grates overlay G_3 can be implemented deterministically. Observe that G_3 has αN nodes, including $2N$ output nodes O that are partitioned into $\frac{N}{k}$ blocks of size $2k$ each. Specifically $O = O_1 \cup \ldots \cup O_{\frac{N}{k}}$, where $O_j = [(\alpha-2)N + 1 + 2jk, (\alpha-2)N + 2(j+1)k]$ for $j \in \left[\frac{N}{k}\right]$.

As stated in Figure 4, the collision-resistant block partition extension G_4 is actually data-independent, since for each $i \in [(\alpha-1)N, \alpha N]$, each parent $r(i)$ of i is chosen uniformly at random from k possible nodes, but the random procedure is independent of the input x. Hence, the challenge is to implement a computationally data-independent version of the collision-resistant block partition extension. We demand the input of a key K for each computation of $f_{G,H}(x)$. The value of $f_{G,H}(x)$ remains the same across all keys K but the leakage pattern is different for each K.

Data-Dependent Dynamic Graph

For each $i \in [\alpha N + 1, (\alpha+1)N]$, let $j = i \mod \frac{N}{k}$. To implement a computationally data-independent version of G_4 from Figure 4, we use the value of $L_{G,H,x}(i-1)$ to select a previously "unused" node of O_j as the parent $r(i)$ for i that is not $i-1$. Here, we say a node $v \in O_j$ is unused if it is not the parent of any node besides $v+1$ and i.

Since $L_{G,H,x}(i-1)$ can be viewed as a random integer modulo $2k$, we can use $L_{G,H,x}(i-1) \pmod{2k}$ as an input to a permutation with key K to randomly choose the parent of i from O_j. Observe that $i = \alpha N + j$ is the first time a parent will be selected from O_j. Moreover, observe that $m = L_{G,H,x}(i-1) \pmod{2k} + 1$ can be viewed as a random number from $[2k]$ so we set $r(i) = m + (\alpha-2)N + 1 + 2jk$ as the m^{th} entry of O_j.

Now if $L_{G,H,x}(i-1) \pmod{2k}$ were all unique across the values of $\{i | i \mod \frac{N}{k} \equiv j\}$, then there would be no collisions among selections of parents in O_j and we would be done. However, since these values are not unique, we must do a little more work to avoid collisions, which would reveal information through leakage patterns about the parents of two nodes being the same. To ensure there are no collisions in O_j among parents, we store an array U_j of size $2k$ for each block O_j. For each $1 \le \ell \le 2k$, we initialize $U_j[\ell] = \ell$. The purpose of the U_j array is to ensure that the nodes of O_j that have already appeared as parents are at the end of U_j. In the above example when $r(i) = m + (\alpha-2)N + 1 + 2jk$, we then set $U_j[m] = 2k$ and $U_j[2k] = m$. Then in the next round of selecting a parent from O_j, we choose uniformly at random from the first $2k-1$ entries of U_j and in general, for the s^{th} round of selecting a parent from O_j, we choose uniformly at random from the first $2k-s+1$ entries of U_j.

Specifically for some $2 \le s \le k$, consider the s^{th} iteration in which a parent is selected from O_j. That is, for $i = \alpha N + j + (s-1)\frac{N}{k}$, the parent $r(i)$ is the s^{th} parent among the nodes of O_j. Observe that $m = L_{G,H,x}(i-1) \pmod{2k-s+1}$ can be viewed as a random number from $[2k-s+1]$ and so $U_j[m]$ is a random entry among the unselected $2k-s+1$ nodes of O_j. We then swap the values of $U_j[m]$ and $U_j[2k-s+1]$ so that if $U_j[m] = a$ and $U_j[2k-s+1] = b$ previously then we set $U_j[m] = b$ and $U_j[2k-s+1] = a$. Hence, the invariant remains that the first $2k-s$ locations of U_j have been unused.

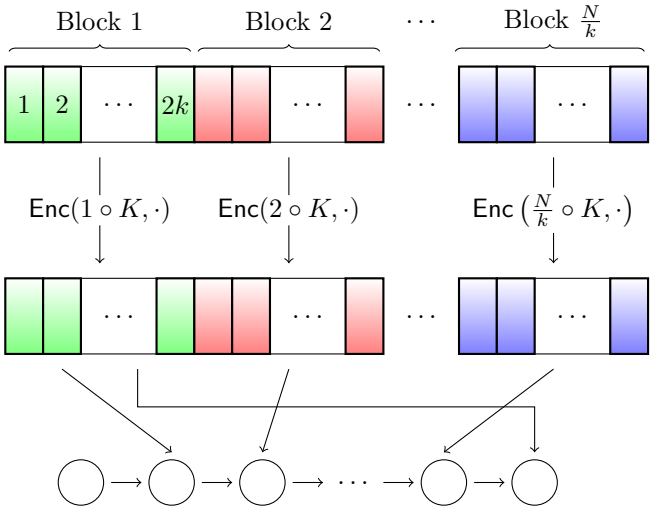


Figure 6 Parent $r(i)$ is drawn uniformly at random from the nodes partitioned to each block.

Shuffling Leakage Patterns

Finally, we point out that the leakage pattern across all computations of $f_{G,H}(x)$ is still the same, since we have not actually incorporated the key K in any of the above details. In summary, the above the description ensures a collision-resistant block partition extension that is data-dependent, but is still vulnerable to side-channel attacks. Hence, we add a final element to our implementation that shuffles the locations of each node $p \in O_j$ inside O_j. That is, for each $1 \leq j \leq \frac{N}{k}$, we use the keyed permutation Enc to store the label of $p \in O_j$ in the location that corresponds to $\mathsf{Enc}(j \circ K, p)$ instead. Thus if $r(i) = p \in O_j$ for some node i, the algorithm must look at the location associated with $\mathsf{Enc}(j \circ K, p)$ to learn the value of $L_{G,H,x}(p)$. Therefore, the underlying graph G is a dynamic graph that is data-dependent but the leakage pattern across each computation of $f_{G,H}(x)$ is different due to the choice of K that shuffles the locations of all labels in each block O_j. A high level example of this shuffling is shown in Figure 6.

Observe that this shuffling must be done completely in the cache to avoid leaking locations of labels during the shuffling. Hence, we require cache eviction policies such as the least recently used (LRU) or first in first out (FIFO) cache eviction policies to ensure that the entire block O_j will remain in cache as the shuffling is performed. We describe the implementation in full in Figure 7.

A Note on Oblivious RAM

The complications with the cache eviction policies and shuffling leakage patterns originate from the necessity of not divulging information in the data-independency game. One reasonable question is whether these complications can be avoided with other implementations that conceal the leakage patterns. For example, an algorithm using oblivious RAM (ORAM), introduced by Goldreich and Ostrovsky [23], reveals no information through the memory access patterns about the underlying operations performed. Thus, an algorithm using an ORAM data structure to evaluate a memory hard function would induce a computationally independent memory hard function, regardless of whether the underlying function is data-

Computationally data-independent sequential evaluation algorithm $\mathsf{MHF.Eval}(x; R)$ to compute $f_{G,H}(x)$ for any k-restricted dynamic graph G that is amenable to shuffling.

(1) Data-independent phase:
- **a.** Let G be a k-restricted dynamic graph with αN nodes for some constant $\alpha > 1$ that is amenable to shuffling, L be the last N nodes of G, and H be an arbitrary hash function.
- **b.** Let $K = \mathsf{Setup}(1^\lambda; R)$ be a hidden random permutation key for each computation of $f_{G,H}(x)$, given the security parameter λ and random bits R.
- **c.** Recall that the subgraph induced by the first $(\alpha - 1)N$ nodes is a static graph. Compute the label $L_{G,H,x}(v)$ for each node $v \in (G - L)$.

(2) Shuffling phase:
- **a.** Since G is amenable to shuffling, L can be partitioned into groups $G_1, \ldots, G_{\frac{N}{k}}$ that satisfy the definition of Definition 19. For each $j \in \left[\frac{N}{k}\right]$, let $O_j = \mathsf{PotentialParents}(G_j)$.
- **b.** For each $j \in \left[\frac{N}{k}\right]$, shuffle the contents of O_j:
  - **i.** Let $v_1, \ldots, v_k$ be the vertices in O_j.
  - **ii.** Load the labels $L_{G,H,x}(v_1), \ldots, L_{G,H,x}(v_k)$ into cache.
  - **iii.** Shuffle the positions of $L_{G,H,x}(v_1), \ldots, L_{G,H,x}(v_k)$ so that for each $p \in [k]$, $L_{G,H,x}(v_p)$ is in the location that previously corresponded to $L_{G,H,x}(v_q)$, where $q = \mathsf{Enc}(j \circ K, p)$, where Enc is a keyed pseudorandom permutation of k.

(3) Data-dependent phase:
- **a.** For each $j \in \left[\frac{N}{k}\right]$, initialize an array U_j such that for all $1 \le \ell \le k$, $U_j[\ell] = \ell$.
- **b.** For each $i = \alpha N + 1$ to $(\alpha + 1)N$:
  - **i.** Let j and s be defined so that $1 \le s \le k$ and $1 \le j \le \frac{N}{k}$ given $i = \alpha N + j + (s-1)\frac{N}{k}$ and let $m = L_{G,H,x}(i-1) \pmod{k - s + 1} + 1$.
  - **ii.** Set $r(i) = U_j[m] + (\alpha - 1)N + 1 + jk$ so that $r(i) \in O_j$ and load $L_{G,H,x}(r(i))$. (Recall that the label of $r(i)$ is actually located at the position where the label of node $\mathsf{Enc}(j \circ K, U[m])$ was previously located prior to the shuffling.)
  - **iii.** Load $L_{G,H,x}(r(i))$ and $L_{G,H,x}(i-1)$ and compute $L_{G,H,x}(i) = H(i \circ L_{G,H,x}(r(i)) \circ L_{G,H,x}(i-1))$.
  - **iv.** Let $U_j[U_j[m]] = a$ and $U_j[k - s + 1] = b$. Then swap the values of U_j at $U_j[m]$ and $k - s + 1$ so that $U_j[U_j[m]] = b$ and $U_j[k - s + 1] = a$.

Figure 7 Description of evaluation algorithm for k-restricted graphs that are amenable to shuffling. Note that each computation of $f_{G,H}(x)$ requires as input random bits R to generate the leakage patterns.

dependent or data-independent. [23] describe an oblivious RAM simulator that transforms any program in the standard RAM model into a program in the oblivious RAM model, where the leakage pattern is information theoretically hidden, which is ideal for the data-independency game.

Existing constructions of ORAM protocols such as Path ORAM [31] require amortized $\Omega(\log N)$ bandwidth overhead. Hence given any dMHF and evaluation algorithm running in sequential time M, we can use ORAM to develop a new evaluation algorithm with a concealed leakage pattern, running in sequential time $N = M \log M$. However, this is not ideal because the cumulative memory complexity of the dMHF is $\mathcal{O}\left(M^2\right) = \mathcal{O}\left(\frac{N^2}{\log^2 N}\right)$. Viewed in this way, the ciMHF construction is worse than known iMHF constructions that achieve CMC $\Omega\left(\frac{N^2}{\log N}\right)$ such as DRSample [4, 11]. In fact, even for k-restricted graphs,

we still obtain a blow-up of $\Omega(\log^2 K)$, which is $\Omega\left(\frac{\log^2 N}{\log^2 \log N}\right)$ when $k = \Omega(N^{1/\log\log N})$. Otherwise for $k = o(N^{1/\log\log N})$, our dynamic pebbling attack in Corollary 11 shows that the CMC is at most $o(N^2)$.

Although Boyle and Naor [20] proposed the notion of *online* ORAM, where the operations to be performed arrive in an online manner, and observe that the lower bounds of [23] do not hold for online ORAM, Larsen and Nielsen [25] answer this open question by proving an amortized $\Omega(\log N)$ bandwidth overhead lower bound on the bandwidth of any online ORAM. Therefore, it does not seem obvious how to use ORAM in the implementations of maximally hard ciMHFs.

6.2.1 Implementation and Analysis

Hybrid:

(1) Data-independent phase:
- **a.** Let G be a k-restricted dynamic graph with αN nodes for some constant $\alpha > 1$ that is amenable to shuffling, L be the last N nodes of G, and H be an arbitrary hash function.
- **b.** Let $K = \mathsf{Setup}(1^\lambda; R)$ be a hidden random permutation key for each computation of $f_{G,H}(x)$, given the security parameter λ and random bits R.
- **c.** Recall that the subgraph induced by the first $(\alpha - 1)N$ nodes is a static graph. Compute the label $L_{G,H,x}(v)$ for each node $v \in (G - L)$.

(2) Shuffling phase:
- **a.** Since G is amenable to shuffling, L can be partitioned into groups $G_1, \ldots, G_{\frac{N}{k}}$ that satisfy the definition of Definition 19. For each $j \in \left[\frac{N}{k}\right]$, let $O_j = \mathsf{PotentialParents}(G_j)$.
- **b.** For each $j \in \left[\frac{N}{k}\right]$, shuffle the contents of O_j:
  - **i.** Let $v_1, \ldots, v_k$ be the vertices in O_j.
  - **ii.** Load the labels $L_{G,H,x}(v_1), \ldots, L_{G,H,x}(v_k)$ into cache.
  - **iii.** Shuffle the positions of $L_{G,H,x}(v_1), \ldots, L_{G,H,x}(v_k)$ so that for each $p \in [k]$, $L_{G,H,x}(v_p)$ is in the location that previously corresponded to $L_{G,H,x}(v_q)$, where $q = \mathsf{Enc}(j \circ K, p)$, where Enc is a keyed truly random permutation of k.

(3) Data-dependent phase:
- **a.** For each $j \in \left[\frac{N}{k}\right]$, initialize an array U_j such that for all $1 \leq \ell \leq k$, $U_j[\ell] = \ell$.
- **b.** For each $i = \alpha N + 1$ to $(\alpha + 1)N$:
  - **i.** Let j and s be defined so that $1 \leq s \leq k$ and $1 \leq j \leq \frac{N}{k}$ given $i = \alpha N + j + (s-1)\frac{N}{k}$ and let $m = L_{G,H,x}(i-1) \pmod{k - s + 1} + 1$.
  - **ii.** Set $r(i) = U_j[m] + (\alpha - 1)N + 1 + jk$ so that $r(i) \in O_j$ and load $L_{G,H,x}(r(i))$. (Recall that the label of $r(i)$ is actually located at the position where the label of node $\mathsf{Enc}(j \circ K, U[m])$ was previously located prior to the shuffling.)
  - **iii.** Load $L_{G,H,x}(r(i))$ and $L_{G,H,x}(i-1)$ and compute $L_{G,H,x}(i) = H(i \circ L_{G,H,x}(r(i)) \circ L_{G,H,x}(i-1))$.
  - **iv.** Let $U_j[U_j[m]] = a$ and $U_j[k - s + 1] = b$. Then swap the values of U_j at $U_j[m]$ and $k - s + 1$ so that $U_j[U_j[m]] = b$ and $U_j[k - s + 1] = a$.

Figure 8 Description of hybrid. Differs from Figure 7 in that the hidden input key is used to index into the entire family of random permutations, rather than a pseudorandom permutation.

We require the hybrid in Figure 8 to argue that our implementation of Figure 4 is a ciMHF. The hybrid in Figure 8 differs from the implementation of Figure 4 in Figure 7 in that the hidden input key is used to index from the entire family of random permutations, rather than a pseudorandom permutation. Thus the only way an adversary can distinguish between the hybrid and the real world sampler is by distinguishing between a random permutation and a pseudorandom permutation. On the other hand, if an adversary fails to distinguish between the hybrid and the real world sampler, then the cumulative memory complexity of the implementation requires $\Omega(N^2)$ since the leakage pattern of the hybrid is statistically equivalent to the dMHF construction in Figure 4, where each parent is chosen a priori using a permutation drawn uniformly at random.

▶ **Theorem 24.** *For each DAG G that is amenable to shuffling, there exists a computationally data-independent sequential evaluation algorithm* $\mathsf{MHF.Eval}(x; R)$ *computing the function $f_{\mathbb{G},H}$ in time $\mathcal{O}(N)$.*

Proof. Consider the evaluation function in Figure 7. Observe that the hybrid in Figure 8 has the same distribution of leakage patterns as the dMHF of Figure 4. Moreover, under the least recently used (LRU) or first in first out (FIFO) cache eviction policies, if k is less than the size of the cache, then all the shuffling can be performed so an attacker observing the leakage patterns of the hybrid has no advantage in the data-independency game. Furthermore, the ciMHF implementation in Figure 7 only differs from the hybrid in Figure 8 in the implementation of Enc as a pseudorandom permutation compared to a truly random permutation. Therefore, an attacker observing leakage patterns from the implementation in Figure 7 only obtains a negligible advantage $\mathsf{negl}(\lambda)$ in the security parameter λ, in the data-independency game. Hence, the implementation of Figure 7 is a ciMHF. ◀

We now show that the evaluation function in Figure 7 of the dMHF in Figure 4 is a maximally hard ciMHF.

▶ **Theorem 25.** *Let $0 < \epsilon < 1$ be a constant and $k = \Omega(N^\epsilon)$. Then there exists a family $\mathbb{G}$ of k-restricted graphs with $\mathsf{cc}(\mathbb{G}) = \Omega(N^2)$ that is amenable to shuffling. Moreover, there exists a negligible value $\delta = \mathsf{negl}(N)$ such that $\mathsf{cc}_\delta(\mathbb{G}) = \Omega(N^2)$.*

Proof. Consider the evaluation function in Figure 7 of the dMHF in Figure 4. For the sake of completeness, the full implementation is also shown in Figure 9. Since the construction of Figure 4 is amenable to shuffling, then the evaluation algorithm is a ciMHF by Theorem 24. Finally by Theorem 21, $\mathsf{cc}(\mathbb{G}) = \Omega(N^2)$.

In fact, Theorem 21 implies that for $G \in \mathbb{G}$ drawn uniformly at random and any pebbling strategy S, not only is $\mathop{\mathbb{E}}_{G\sim\mathbb{G}}[\mathsf{cc}(S, G)] = \Omega(N^2)$, but also $\mathsf{cc}(S, G) = \Omega(N^2)$ with probability at least $1-c^{N/k}$ for some constant $0 < c < 1$. Thus for $\delta = 1-c^{N/k}$, we have $\mathsf{cc}(S, \mathbb{G}, \delta) = \Omega(N^2)$ for any pebbling strategy S and so $\mathsf{cc}_\delta(\mathbb{G}) = \Omega(N^2)$. ◀

For the sake of completeness, we give the evaluation algorithm for the maximally hard ciMHF in Figure 9.

Computationally data-independent sequential evaluation algorithm $\mathsf{MHF.Eval}(x; R)$

(1) Let H be an arbitrary hash function and $K = \mathsf{Setup}(1^\lambda; R)$ be a hidden random permutation key for each computation of $f_{\mathbb{G},H}(x)$, given the security parameter λ and random bits R. Let $k = \Omega(N^\epsilon)$.

(2) Data-independent phase:

- **a.** $G_1 = \mathsf{grates}_{2N,\epsilon}$
- **b.** $G_2 = \mathsf{superconc}(G_1)$
- **c.** $G_3 = \mathsf{grates}_\epsilon(G_2)$, which has αN total nodes, including $2N$ output nodes $O = O_1 \cup \ldots \cup O_{\frac{N}{k}}$, where $O_j = [(\alpha-2)N+1+2ik, (\alpha-2)N+2(j+1)k]$ for $j \in \left[\frac{N}{k}\right]$.
- **d.** Compute the label $L_{G_3,H,x}(v)$ for each node $v \in G_3$.

(3) Shuffling phase:

- **a.** Let G be the graph G_3 appended with N additional nodes, so that $V = [(\alpha+1)N]$, and edges $(i-1, i)$ for each $\alpha N + 1 \leq i \leq (\alpha+1)N$.
- **b.** For each $j \in \left[\frac{N}{k}\right]$, shuffle the contents of O_j:
  - **i.** Let $v_1, \ldots, v_{2k}$ be the vertices in O_j.
  - **ii.** Load the labels $L_{G,H,x}(v_1), \ldots, L_{G,H,x}(v_{2k})$ into cache.
  - **iii.** Shuffle the positions of $L_{G,H,x}(v_1), \ldots, L_{G,H,x}(v_{2k})$ so that for each $p \in [2k]$, $L_{G,H,x}(v_p)$ is in the location that previously corresponded to $L_{G,H,x}(v_q)$, where $q = \mathsf{Enc}(j \circ K, p)$, where Enc is a keyed pseudorandom permutation of $2k$.

(4) Data-dependent phase:

- **a.** For each $j \in \left[\frac{N}{k}\right]$, initialize an array U_j such that for all $1 \leq \ell \leq 2k$, $U_j[\ell] = \ell$.
- **b.** For each $i = \alpha N + 1$ to $(\alpha+1)N$:
  - **i.** Let j and s be defined so that $1 \leq s \leq k$ and $1 \leq j \leq \frac{N}{k}$ given $i = \alpha N + j + (s-1)\frac{N}{k}$ and let $m = L_{G,H,x}(i-1) \pmod{2k-s+1} + 1$.
  - **ii.** Set $r(i) = U_j[m] + (\alpha-2)N + 1 + 2jk$ so that $r(i) \in O_j$ and load $L_{G,H,x}(r(i))$. (Recall that the label of $r(i)$ is actually located at the position where the label of node $\mathsf{Enc}(j \circ K, U[m])$ was previously located prior to the shuffling.)
  - **iii.** Load $L_{G,H,x}(r(i))$ and $L_{G,H,x}(i-1)$ and compute $L_{G,H,x}(i) = H(i \circ L_{G,H,x}(r(i)) \circ L_{G,H,x}(i-1))$.
  - **iv.** Let $U_j[U_j[m]] = a$ and $U_j[2k-s+1] = b$. Then swap the values of U_j at $U_j[m]$ and $2k-s+1$ so that $U_j[U_j[m]] = b$ and $U_j[2k-s+1] = a$.

Figure 9 Description of implementation of *maximally hard* ciMHF. Again note that each computation of $f_{G,H}(x)$ requires as input random bits R to generate the leakage pattern.

6.2.2 Extension to Multiple Rounds

Finally, we show that our ciMHF implementation is robust to multiple rounds of leakage by considering a data independency game where an adversary is allowed to submit and observe multiple adaptive queries before outputting a guess for the hidden challenge bit b. The game again consists of the phases **setup**, **challenge**, and **guess**, which are described as follows.

Adaptive data independency game for ciMHF:

setup In this phase, $\mathcal{A}$ selects the security parameter λ and sends it to $\mathcal{H}$. $\mathcal{H}$ then selects a random bit $b \in \{0,1\}$.

challenge For each round $i = 1, 2, \ldots$, $\mathcal{A}$ chooses two adaptive query messages $x_{i,0}$ and $x_{i,1}$ and sends the query messages to $\mathcal{H}$. $\mathcal{H}$ selects random coins $R_{i,0}, R_{i,1} \in \{0,1\}^\lambda$ uniformly at random, samples $\mathsf{lp}_{i,0} \leftarrow \mathsf{LP}(\mathsf{MHF.Eval}(x_{i,0}; R_{i,0}))$ and $\mathsf{lp}_{i,1} \leftarrow \mathsf{LP}(\mathsf{MHF.Eval}(x_{i,1}; R_{i,1}))$, and sends the ordered pair $(\mathsf{lp}_{i,b}, \mathsf{lp}_{i,1-b})$ to $\mathcal{A}$. Note: $\mathcal{A}$ can pick $x_{i+1,0}$ and $x_{i+1,1}$ adaptively after observing the response $(\mathsf{lp}_{i,b}, \mathsf{lp}_{i,1-b})$.

guess The game ends when the adversary $\mathcal{A}$ outputs b' as a guess for b. $\mathcal{A}$ wins the game if $b = b'$.

As before, the advantage of the adversary to win the adaptive data independency game for ciMHF is:

$$Adv^{\mathsf{ind-mult-lp-iMHF}}_{\mathcal{A},\mathsf{MHF}} = \left| \frac{1}{2} - \mathbf{Pr}\left[\mathcal{A}(\mathcal{T}) = b' : b = b')\right] \right|,$$

where $\mathcal{T}$ is the transcript $\{x_{i,0}, x_{i,1}, \mathsf{lp}_{i,b}, \mathsf{lp}_{i,1-b}\}$ and $\mathsf{lp}_{i,j} = \mathsf{LP}(x_{i,j}; \mathsf{MHF.Eval}(x_{i,j}; R_{i,j}))$.

▶ **Definition 26.** *We say an evaluation algorithm* MHF.Eval *has (t,ϵ)-single security if any attacker running in time t has at most advantage ϵ in the data independency game. Similarly, we say an evaluation algorithm* MHF.Eval *has (t,r,ϵ)-adaptive security if any attacker running in time t and making r queries has at most advantage ϵ in the adaptive data independency game.*

We conclude by noting the following relationship between single security and adaptive security, thus implying the security of our evaluation function in Figure 7 of the dMHF in Figure 4 with respect to the adaptive data independency game.

▶ **Theorem 27.** *(t,ϵ)-single security implies $(t - \mathcal{O}(r \cdot \mathsf{time}(\mathsf{MHF.Eval})), r, r\epsilon)$-adaptive security.*

Proof. Suppose that $\mathsf{A}_{adaptive}$ violates $(t - \mathcal{O}(r \cdot \mathsf{time}(\mathsf{MHF.Eval})), r, r\epsilon)$-adaptive security for the sake of contradiction. Without loss of generality we will assume that $\mathsf{A}_{adaptive}$ outputs $b' = b$ with probability greater than $\frac{1}{2} + r\epsilon$. We will use $\mathsf{A}_{adaptive}$ to construct an attacker A_{single} that violates (t,ϵ)-single security.

We first define a sequence of r hybrids in the adaptive data-independency game. In Hybrid i, the challenger $\mathcal{H}$ picks bits $b, b_1, \ldots, b_{i-1}$ uniformly at random and sets $b_i = b, b_{i+1} = b, \ldots, b_r = b$. In round j when the attacker $\mathsf{A}_{adaptive}$ submits two strings $x_{j,0}$ and $x_{j,1}$, the challenger $\mathcal{H}$ samples $\mathsf{lp}_{j,0} \leftarrow \mathsf{LP}(\mathsf{MHF.Eval}(x_{j,0}; R_{j,0}))$ and $\mathsf{lp}_{j,1} \leftarrow \mathsf{LP}(\mathsf{MHF.Eval}(x_{j,1}; R_{j,1}))$ and then responds with $\mathsf{lp}_{j,b_i}, \mathsf{lp}_{j,1-b_i}$ instead of $\mathsf{lp}_{j,b}$ and $\mathsf{lp}_{j,1-b}$ i.e., the bit b_i is used to permute the order of the responses in round i instead of b.

Observe that in Hybrid 1, $b_1 = \ldots = b_r = b$, so that Hybrid 1 is equivalent to the actual adaptive independency game. Similarly, in Hybrid r, the bits $b_1, \ldots, b_r$ are all picked independently so that $\mathsf{A}_{adaptive}$ working in Hybrid r has no advantage i.e., the attacker guesses $b' = b$ correctly with probability at most $\mathbf{Pr}[b' = b \mid \text{Hybrid } r] = \frac{1}{2}$. We observe that the advantage of the attacker is

$$\mathbf{Pr}[b' = b \mid \text{Hybrid } 1] - \frac{1}{2} = \mathbf{Pr}[b' = b \mid \text{Hybrid } 1] - \mathbf{Pr}[b' = b \mid \text{Hybrid } r] = \Delta_2 + \ldots + \Delta_r\,,$$

where $\Delta_i = \mathbf{Pr}[b' = b \mid \text{Hybrid } i-1] - \mathbf{Pr}[b' = b \mid \text{Hybrid } i]$. By an averaging argument, we must have $\Delta_{i+1} > \epsilon$ for some $i < r$. The following observation will also be useful:

$$\Delta_{i+1} = \frac{1}{2}\mathbf{Pr}[b' = b \mid \text{Hybrid } i] - \frac{1}{2}\mathbf{Pr}[b = b' \mid \text{Hybrid } i+1, b_i \neq b]\,.$$

Reduction

We now define A_{single} as follows: (1) A_{single} simulates $\mathsf{A}_{adaptive}$ along with the adaptive challenger $\mathcal{H}_{adaptive}$. A_{single} generates random bits $b_1, \ldots, b_{i-1}$ and sets $b_{i+1} = \ldots = b_r = b''$ for another random bit b''. In each round $j \neq i$, when $\mathsf{A}_{adaptive}$ outputs a query $x_{j,0}, x_{j,1}$, our attacker A_{single} simply computes $\mathsf{lp}_{i,0} \leftarrow \mathsf{LP}(\mathsf{MHF.Eval}(x_{i,0}; R_{i,0}))$ and $\mathsf{lp}_{i,1} \leftarrow \mathsf{LP}(\mathsf{MHF.Eval}(x_{i,1}; R_{i,1}))$ and responds with $\mathsf{lp}_{i,b_i}, \mathsf{lp}_{i,1-b_i}$. When $\mathsf{A}_{adaptive}$ outputs the query $x_{i,0}, x_{i,1}$ in round i, A_{single} forwards this query to the challenger for the single stage challenger $\mathcal{H}_{single}$ and receives back $\mathsf{lp}_{i,b}, \mathsf{lp}_{i,1-b}$ for an unknown bit b selected by $\mathcal{H}_{single}$. Finally, when $\mathsf{A}_{adaptive}$ outputs a guess b' (for b'') A_{single} outputs the same guess b' (for b).

Analysis

Notice that since b'' is just a bit selected uniformly at random and independent from b, then $\mathbf{Pr}[b' = b'' \,|b'' = b] = \mathbf{Pr}[b' = b'' \mid \text{Hybrid } i]$. Then from the above observation, we have $\mathbf{Pr}[b' = b'' \,|b'' \neq b] = \mathbf{Pr}[b' = b'' \mid \text{Hybrid } i+1, b_i \neq b''] = \mathbf{Pr}[b' = b''| \text{ Hybrid } i] - 2\Delta_{i+1}$. It follows that $\mathbf{Pr}[b' = b'' \mid b'' = b] - \mathbf{Pr}[b' = b'' \,|b'' \neq b] = 2\Delta_{i+1}$. Thus, the probability that A_{single} wins is

$$\begin{aligned}\mathbf{Pr}[b' = b] &= \mathbf{Pr}[b'' = b]\,\mathbf{Pr}[b' = b'' \mid b'' = b] + \mathbf{Pr}[b'' \neq b]\,(1 - \mathbf{Pr}[b' = b'' \,|b'' \neq b]) \\ &= \frac{1}{2} + \Delta_{i+1} > \epsilon.\end{aligned}$$

Furthermore, the running time of A_{single} is at most t. This contradicts the assumption that the evaluation algorithm $\mathsf{MHF.Eval}$ has (t, ϵ)-single security. Therefore, (t, ϵ)-single security implies $(t - \mathcal{O}(r \cdot \mathsf{time}(\mathsf{MHF.Eval})), r, r\epsilon)$-adaptive security. ◀

References

1 Martín Abadi, Michael Burrows, Mark S. Manasse, and Ted Wobber. Moderately hard, memory-bound functions. *ACM Trans. Internet Techn.*, 5(2):299–327, 2005.

2 Joël Alwen and Jeremiah Blocki. Efficiently Computing Data-Independent Memory-Hard Functions. In *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Proceedings, Part II*, pages 241–271, 2016.

3 Joël Alwen and Jeremiah Blocki. Towards Practical Attacks on Argon2i and Balloon Hashing. In *2017 IEEE European Symposium on Security and Privacy, EuroS&P*, pages 142–157, 2017.

4 Joël Alwen, Jeremiah Blocki, and Benjamin Harsha. Practical Graphs for Optimal Side-Channel Resistant Memory-Hard Functions. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security, CCS*, pages 1001–1017, 2017.

5 Joël Alwen, Jeremiah Blocki, and Krzysztof Pietrzak. Depth-Robust Graphs and Their Cumulative Memory Complexity. In *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Proceedings, Part III*, pages 3–32, 2017.

6 Joël Alwen, Binyi Chen, Krzysztof Pietrzak, Leonid Reyzin, and Stefano Tessaro. Scrypt Is Maximally Memory-Hard. In *Advances in Cryptology - EUROCRYPT - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Proceedings, Part III*, pages 33–62, 2017.

7 Joël Alwen and Vladimir Serbinenko. High Parallel Complexity Graphs and Memory-Hard Functions. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC*, 2015.

8 Daniel J Bernstein. Cache-timing attacks on AES, 2005.

9 Alex Biryukov, Daniel Dinu, and Dmitry Khovratovich. Fast and Tradeoff-Resilient Memory-Hard Functions for Cryptocurrencies and Password Hashing. *IACR Cryptology ePrint Archive*, 2015:430, 2015.

10 Alex Biryukov, Daniel Dinu, and Dmitry Khovratovich. Argon2: New Generation of Memory-Hard Functions for Password Hashing and Other Applications. In *IEEE European Symposium on Security and Privacy, EuroS&P*, pages 292–302, 2016.

11 Jeremiah Blocki, Benjamin Harsha, Siteng Kang, Seunghoon Lee, Lu Xing, and Samson Zhou. Data-Independent Memory Hard Functions: New Attacks and Stronger Constructions. In *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Proceedings, Part II*, pages 573–607, 2019.

12 Jeremiah Blocki, Benjamin Harsha, and Samson Zhou. On the Economics of Offline Password Cracking. In *2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings*, pages 853–871, 2018.

13 Jeremiah Blocki, Seunghoon Lee, and Samson Zhou. Approximating Cumulative Pebbling Cost is Unique Games Hard. *CoRR*, abs/1904.08078, 2019. `arXiv:1904.08078`.

14 Jeremiah Blocki, Ling Ren, and Samson Zhou. Bandwidth-Hard Functions: Reductions and Lower Bounds. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security, CCS*, pages 1820–1836, 2018.

15 Jeremiah Blocki and Samson Zhou. On the Depth-Robustness and Cumulative Pebbling Cost of Argon2i. In *Theory of Cryptography - 15th International Conference, TCC, Proceedings, Part I*, pages 445–465, 2017.

16 Jeremiah Blocki and Samson Zhou. On the Computational Complexity of Minimal Cumulative Cost Graph Pebbling. In *Financial Cryptography and Data Security - 22nd International Conference, FC, Revised Selected Papers*, pages 329–346, 2018.

17 Dan Boneh, Henry Corrigan-Gibbs, and Stuart E. Schechter. Balloon Hashing: A Memory-Hard Function Providing Provable Protection Against Sequential Attacks. In *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Proceedings, Part I*, pages 220–248, 2016.

18 Joseph Bonneau. The Science of Guessing: Analyzing an Anonymized Corpus of 70 Million Passwords. In *2012 IEEE Symposium on Security and Privacy*, pages 538–552, San Francisco, CA, USA, May 21–23 2012. IEEE Computer Society Press. `doi:10.1109/SP.2012.49`.

19 Xavier Boyen. Halting Password Puzzles: Hard-to-break Encryption from Human-memorable Keys. In *Proceedings of the 16th USENIX Security Symposium*, 2007.

20 Elette Boyle and Moni Naor. Is There an Oblivious RAM Lower Bound? In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science*, pages 357–368, 2016.

21 Cynthia Dwork and Moni Naor. Pricing via Processing or Combatting Junk Mail. In *Advances in Cryptology - CRYPTO '92, 12th Annual International Cryptology Conference, Proceedings*, pages 139–147, 1992.

22 Christian Forler, Stefan Lucks, and Jakob Wenzel. Memory-Demanding Password Scrambling. In *Advances in Cryptology - ASIACRYPT - 20th International Conference on the Theory and Application of Cryptology and Information Security, Proceedings, Part II*, pages 289–305, 2014.

23 Oded Goldreich and Rafail Ostrovsky. Software Protection and Simulation on Oblivious RAMs. *J. ACM*, 43(3):431–473, 1996.

24 Marcos A. Simplício Jr., Leonardo C. Almeida, Ewerton R. Andrade, Paulo C. F. dos Santos, and Paulo S. L. M. Barreto. Lyra2: Password Hashing Scheme with improved security against time-memory trade-offs. *IACR Cryptology ePrint Archive*, page 136, 2015.

25 Kasper Green Larsen and Jesper Buus Nielsen. Yes, There is an Oblivious RAM Lower Bound! In *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Proceedings, Part II*, pages 523–542, 2018.

26 Thomas Lengauer and Robert Endre Tarjan. Asymptotically tight bounds on time-space trade-offs in a pebble game. *J. ACM*, 29(4):1087–1130, 1982.

27 Colin Percival. Stronger key derivation via sequential memory-hard functions, 2009.

28 Nicholas Pippenger. Superconcentrators. *SIAM J. Comput.*, 6(2):298–304, 1977.

29 Ling Ren and Srinivas Devadas. Bandwidth Hard Functions for ASIC Resistance. In *Theory of Cryptography - 15th International Conference, TCC 2017, Proceedings, Part I*, pages 466–492, 2017.

30 Georg Schnitger. On Depth-Reduction and Grates. In *24th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 323–328, 1983.

31 Emil Stefanov, Marten van Dijk, Elaine Shi, T.-H. Hubert Chan, Christopher W. Fletcher, Ling Ren, Xiangyao Yu, and Srinivas Devadas. Path ORAM: An Extremely Simple Oblivious RAM Protocol. *J. ACM*, 65(4):18:1–18:26, 2018.

32 Leslie G. Valiant. Graph-Theoretic Arguments in Low-Level Complexity. In *Mathematical Foundations of Computer Science 1977, 6th Symposium, Proceedings*, pages 162–176, 1977.

Learning and Testing Variable Partitions

Andrej Bogdanov
Department of Computer Science and Engineering
Institute of Theoretical Computer Science and Communications
The Chinese University of Hong Kong
https://www.cse.cuhk.edu.hk/~andrejb/
andrejb@cse.cuhk.edu.hk

Baoxiang Wang[1]
Department of Computer Science and Engineering, The Chinese University of Hong Kong
https://www.cse.cuhk.edu.hk/~bxwang/
bxwang@cse.cuhk.edu.hk

Abstract

Let F be a multivariate function from a product set Σ^n to an Abelian group G. A k-partition of F with cost δ is a partition of the set of variables $\boldsymbol{V}$ into k non-empty subsets $(\boldsymbol{X}_1, \ldots, \boldsymbol{X}_k)$ such that $F(\boldsymbol{V})$ is δ-close to $F_1(\boldsymbol{X}_1) + \cdots + F_k(\boldsymbol{X}_k)$ for some $F_1, \ldots, F_k$ with respect to a given error metric. We study algorithms for agnostically learning k partitions and testing k-partitionability over various groups and error metrics given query access to F. In particular we show that

1. Given a function that has a k-partition of cost δ, a partition of cost $O(kn^2)(\delta + \varepsilon)$ can be learned in time $\tilde{O}(n^2 \operatorname{poly} 1/\varepsilon)$ for any $\varepsilon > 0$. In contrast, for $k = 2$ and $n = 3$ learning a partition of cost $\delta + \varepsilon$ is NP-hard.
2. When F is real-valued and the error metric is the 2-norm, a 2-partition of cost $\sqrt{\delta^2 + \varepsilon}$ can be learned in time $\tilde{O}(n^5/\varepsilon^2)$.
3. When F is $\mathbb{Z}_q$-valued and the error metric is Hamming weight, k-partitionability is testable with one-sided error and $O(kn^3/\varepsilon)$ non-adaptive queries. We also show that even two-sided testers require $\Omega(n)$ queries when $k = 2$.

This work was motivated by reinforcement learning control tasks in which the set of control variables can be partitioned. The partitioning reduces the task into multiple lower-dimensional ones that are relatively easier to learn. Our second algorithm empirically increases the scores attained over previous heuristic partitioning methods applied in this context.

2012 ACM Subject Classification Theory of computation → Streaming, sublinear and near linear time algorithms; Theory of computation → Approximation algorithms analysis; Theory of computation → Machine learning theory; Theory of computation → Reinforcement learning; Computing methodologies → Reinforcement learning

Keywords and phrases partitioning, agnostic learning, property testing, sublinear-time algorithms, hypergraph cut, reinforcement learning

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.37

Funding Work funded by Hong Kong RGC GRF grant CUHK14209417.

Acknowledgements We would like to thank Arnab Bhattacharyya and Guy Kindler for helpful discussions on our work and its connection to learning and testing juntas, and Jiajin Li for pointing out that variable partitioning for reinforcement learning in fact reduces the variance of its policy gradient estimator.

[1] Authors are listed in alphabetical order.

1 Introduction

Divide-and-conquer methods rely on the ability to identify independent sub-instances of a given instance, such as connected components of graphs and hypergraphs. When these are not available one looks for partitions into loosely related parts like small or sparse cuts. These classic problems and their variants remain at the forefront of algorithmic research [6, 7, 16, 20, 24, 29].

We study the related problem of function decomposition: Given a multivariate function $F(\boldsymbol{V})$ over n variables $\boldsymbol{V} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$, we seek to partition the variables into k groups $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_k$ so that F decomposes into a sum $F_1(\boldsymbol{X}_1) + \cdots + F_k(\boldsymbol{X}_k)$. In case an exact decomposition of this type is unavailable, we seek an approximate one under a suitable error metric. This algebraic partitioning question can be sensibly asked for any Abelian group. While some of our results are quite general, two particular cases of interest are addition over $\mathbb{Z}_2$ with respect to the Hamming metric and addition over reals with respect to the 2-norm.

As a multivariate function is an exponentially large object, it is sensible to model the input F to the partitioning problem as an oracle and allow query access to it. This departs from the common setup in (hyper)graph partitioning problems, where an explicit representation of the input is assumed to be available. While variable partitioning of real-valued functions under the 2-norm turns out to be closely related to hypergraph partitioning, the difference in input access models renders certain techniques developed for the latter (e.g., random contractions) inapplicable to our setting.

Our work is motivated by learning control variables in high-dimensional reinforcement learning control [25, 33, 34]. If the *advantage function* of the control variables can be partitioned into multiple lower-dimensional subsets, then these subsets of variables can be learned independently with a relatively easier Monte-Carlo sampling. This advantage function involves the estimates of a dynamic system, which is complex enough to not have an explicit representation available. The function is thus treated as an oracle as is in our access model. Sometimes it is natural to assume that the function should be almost decomposable; for example, if we seek to control two robots jointly performing a task, the variables controlling the respective robots are almost independent. (The robots may be collaborating so the decomposition might not be perfect.) In general, the dependencies are not known in advance but need to be learned from observed behavior. Some heuristic methods have been applied to control variable partitioning [23, 36] but not rigorously analyzed.

Our contributions

Our main results are algorithmic: We show that variable partitions can be learned agnostically.

Let $F(\boldsymbol{V})$ be a function from some product set to an Abelian group G. A direct sum decomposition of F is a partition $(\boldsymbol{X}_1, \ldots, \boldsymbol{X}_k)$ of the set of variables $\boldsymbol{V}$ such that $F(\boldsymbol{V})$ is $F_1(\boldsymbol{X}_1) + \cdots + F_k(\boldsymbol{X}_k)$ for some functions $F_1, \ldots, F_k$. When the decomposition is imperfect, the decomposition error is measured by

$$\delta(\boldsymbol{X}_1, \ldots, \boldsymbol{X}_k) = \min_{F_1,\ldots,F_k} \|F(X_1, \ldots, X_k) - F_1(X_1) - \cdots - F_k(X_k)\|, \tag{1}$$

where $\|\cdot\| \colon G \to \mathbb{R}^+$ is a partial norm. The definition is given in Section 2; the main examples of interest are $G = \mathbb{Z}_q$ under the Hamming metric $\|F\| = \Pr[F(V) = 0]$ and $G = \mathbb{R}$ under the p-norm $\|F\|_p = \mathbb{E}\big[|F(V)|^p\big]^{1/p}$ for any $p \geq 1$ under some product measure. We seek an approximation of the best-possible partition, which minimizes the objective

$$\delta_2(F) = \min_{\boldsymbol{X}} \delta(\boldsymbol{X}, \overline{\boldsymbol{X}}), \tag{2}$$

Notation	Meaning	Notation	Meaning
$\boldsymbol{x}, \boldsymbol{y} \in \boldsymbol{V}$	variables	$\delta_k(F)$	optimal k-partition error
$\boldsymbol{X}, \boldsymbol{Y}, \overline{\boldsymbol{X}} \subseteq \boldsymbol{V}$	sets of variables	$D_F(\boldsymbol{X}, \boldsymbol{Y})$	dependence score
x, y, X, Y	(random) assignment	$\|\cdot\|, \|\cdot\|_{\mathbb{R},p}$	partial norm and p-norm

for bipartition and

$$\delta_k(F) = \min_{\boldsymbol{X}_1,\ldots,\boldsymbol{X}_k} \delta(\boldsymbol{X}_1, \ldots, \boldsymbol{X}_k). \tag{3}$$

for k-partition. (For p-norms over $\mathbb{R}$ we use the notations $\|\cdot\|_{\mathbb{R},p}$, $\delta_{\mathbb{R},2}(F)$, and $\delta_{\mathbb{R},k}(F)$.)

▶ **Theorem 1.** *Let $\|\cdot\|$ be either 1) $\|\cdot\|_{\mathbb{R},p}$ assuming $\|F\|_{\mathbb{R},2p} = O(1)$, or 2) the Hamming metric over $\mathbb{Z}_q$. There is an algorithm that given parameters n, k, ε, γ, and oracle access to $F\colon \Sigma^n \to \mathbb{R}$ outputs a k-partition $\mathcal{P}$ such that $\delta(\mathcal{P}) \le O(kn^2)(\delta_k(F) + \varepsilon)$ with probability at least $1 - \gamma$. The algorithm makes $O(K^p n^2 \log(n/\gamma)/\varepsilon^{2p})$ queries to F and runs in time linear in the number of queries, for an absolute constant K.*

This algorithm is closely related to the heuristic ones used in the aforementioned empirical studies. However, it only guarantees optimality up to an $O(kn^2)$ approximation factor. While we do not know if an approximation factor of this magnitude is inevitable, in Proposition 17 we show that obtaining a solution with additive error is NP-hard. The proofs are given in Section 4.

In contrast, our second algorithm obtains an additive error for bipartitions of real-valued functions under the 2-norm:

▶ **Theorem 2.** *Let $F\colon \Sigma^n \to \mathbb{R}$ be a function with $\|F\|_{\mathbb{R},4} \le 1$. There is an algorithm that given inputs n, ε, γ, and oracle access to F, runs in time $O(n^5 \log(n/\gamma)/\varepsilon^2)$ and outputs a bipartition $(\boldsymbol{X}, \overline{\boldsymbol{X}})$ such that $\delta_{\mathbb{R},2}(\boldsymbol{X}, \overline{\boldsymbol{X}})^2 \le \delta_{\mathbb{R},2}(F)^2 + \varepsilon$ with probability at least $1 - \gamma$.*

More generally, we give evidence that it may be possible to output a $\sqrt{2 - 2/k}$-approximate k-partition in time $\mathrm{poly}(n^k, k, 1/\varepsilon)$ (Corollary 21). For unbounded k finding a good approximation is ETH hard (Corollary 19).

Theorem 2 and Corollary 19 are based on an equivalence between variable partitioning under the 2-norm and hypergraph partitioning given in Proposition 18. The results are described and proved in Section 5.

As a consequence of Theorem 1, the property of being close to a k-partition is testable with $\tilde{O}(k^{2p} n^{4p+2}/\varepsilon^{2p})$ queries. The query complexity of the tester can be somewhat improved:

▶ **Theorem 3.** *k-partitionability is testable with one-sided error and $O(kn^3/\varepsilon)$ non-adaptive queries with respect to Hamming weight over $\mathbb{Z}_q$, and with $O(k^{2p} n^3/\varepsilon^{2p})$ non-adaptive queries with respect to the p-norm over $\mathbb{R}$ assuming $\|F\|_{2p} \le 1$.*

In Section 6 we prove Theorem 3 and show that $\Omega(n - k)$ queries are necessary even for two-sided error testers.

Ideas and techniques

Our Theorem 1 is inspired by algebraic property testing techniques. The starting point is the dual characterization of partitionability into sets $(\boldsymbol{X}, \overline{\boldsymbol{X}})$ by the constraints $D_F(\boldsymbol{X}, \boldsymbol{Y}) = 0$, where $D_F = F(X, Y) - F(X', Y) - F(X, Y') + F(X', Y')$, for all assignments X, X' to $\boldsymbol{X}$ and

Y, Y' to $\boldsymbol{Y}$. David et al. [10] apply this relation to random inputs towards testing whether a $\mathbb{Z}_2$-valued function F tensors decomposes into a direct sum. The acceptance probability of this test approximates the best decomposition to within a factor of 4 (Proposition 4).

Our partitioning algorithm estimates the *dependence score* $\|D_F(\boldsymbol{x}, \boldsymbol{y})\|$ on every pair of variables $\boldsymbol{x}, \boldsymbol{y}$ (keeping the rest fixed) to decide whether they should be partitioned or not. Here, $\|D_F\|$ is the probability that the test $D_F = 0$ fails for discrete groups like $\mathbb{Z}_2$. In general, it can represent any error metric satisfying the axioms in Section 2. The proof of Theorem 1 amounts to showing that a collection of single variable partitions $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{P}$ for which the local scores $\|D_F(\boldsymbol{x}, \boldsymbol{y})\|$ are small can be glued together into a single k-partition $\mathcal{P}$ with a small global score.

When F is real-valued and error is measured under the 2-norm, variable partitioning has a natural geometric interpretation. Functions that depend on different coordinates are orthogonal modulo their constant term, so the optimal decomposition with respect to a fixed partition $(\boldsymbol{X}_1, \ldots, \boldsymbol{X}_k)$ is given by the projection of F onto the respective subspaces of functions. This yields an equality between the distance and the dependence score (6) for bipartitions and a generalization to k-partitions (Proposition 8). Variable partitioning for functions is then equivalent to hypergraph partitioning of their orthogonal decompositions (Proposition 18), with the cost of cut $(\boldsymbol{X}, \overline{\boldsymbol{X}})$ given by $\frac{1}{4}\|D_F(\boldsymbol{x}, \boldsymbol{y})\|^2$.

This connection suggests the application of hypergraph partitioning algorithms that can be implemented with access to an *approximate* cut oracle[2], leading to Theorem 2. On the negative side it reveals that approximately optimal partitions into a large number of components are hard to find (Corollary 19).

Application to reinforcement learning control

We plug our partitioning algorithm back to reinforcement learning control. In this setting, the oracle is real-valued and as we adapt the 2-norm we use the submodularity cut algorithm described in Theorem 2.

We compare empirically with three previous approaches: The baseline that does not involve partitioning [25,35]; the baseline that trivially partitions n variables into n subsets [22,36]; the work that partitions the variables heuristically [23]. The way [23] partitions the variables is to calculate the discrete estimate of the Hessian of the oracle. Then they remove from Hessian the elements with lowest absolute values, until it forms at least k connected components if the Hessian matrix is treated as the adjacency matrix.

The scores we attained on the tasks in the physics simulator are improved over these approaches, which is demonstrated in Section 7.

Relation to other learning and testing problems

A j-junta is a function that depends on at most j of its n variables. The problems of learning and testing juntas have been extensively studied [2,4,8,9,13,26,30]. While a j-junta is always $(n - j + 1)$-partitionable, the two problems are technically incomparable. Moreover, juntas are usually studied in the regime where the junta size j is significantly smaller than the number of variables n and are therefore partitionable into many (mostly trivial) components.

[2]Several state-of-the-art algorithms for cuts in graphs and hypergraphs rely on random contractions [6, 17, 18]. In particular, Rubinstein et al. [29] showed that $\tilde{O}(n)$ queries to an *exact* cut oracle and similar running time are sufficient to find the minimum cut. We do not know if comparable efficiency can be obtained with an approximate oracle.

In this work we are mostly interested in partitions into two or a small number of components. Nevertheless, this connection between juntas and partitionable functions is used to prove the testing lower bound in Section 6.

Dinur and Golubev [12] showed that the existence of decomposition with respect to a fixed k-partition (given as input) is testable with four queries and soundness error $\Omega(\delta)$. The case $k = 2$ was already analyzed by David et al. [10] (see Section 3).

2 Some additional definitions

Let $F(\boldsymbol{V})$ be a function from some product set to an Abelian group G. In general we will assume that the variables $\boldsymbol{V}$ take values in some set Σ endowed with a product measure which is efficiently sampleable. The quality of the partition $(\boldsymbol{X}_1, \ldots, \boldsymbol{X}_k)$ of $\boldsymbol{V}$ is measured by $\delta(\boldsymbol{X}_1, \ldots, \boldsymbol{X}_k)$ given in (1), where $\|\cdot\| \colon G \to \mathbb{R}^+$ can be any functional satisfying the following three axioms:

1. $\|0\| = 0$;
2. $\|F_1 + F_2\| \leq \|F_1\| + \|F_2\|$;
3. $\mathbb{E}[\|F(X, \cdot)\| \mid X] \leq \|F\|$ for any set of variables X of F.

Our goal is to approximately optimize $\delta_2(F)$ in (2) and $\delta_k(F)$ in (3).

Our algorithms are based on the following dependence estimator inspired by the rank-1 test of [10]. Let $\boldsymbol{X}$ and $\boldsymbol{Y}$ be two disjoint sets of variables. The dependence estimator $D_F(\boldsymbol{X}, \boldsymbol{Y})$ is the random variable

$$D_F = F(X, Y, Z) + F(X', Y', Z) - F(X', Y, Z) - F(X, Y', Z)$$

where X, X' are independent samples of the $\boldsymbol{X}$ variable, Y, Y' are independent samples of the $\boldsymbol{Y}$ variable, and Z is a random sample of the remaining variables. If F decomposes into a direct sum that partitions the $\boldsymbol{X}$ and $\boldsymbol{Y}$ variables then D_F equals zero. Conversely, $\|D_F\|$ measures the quality of the approximation.

In the analysis it will be convenient to use the notation $F \approx_\delta G$ for $\|F - G\|_p \leq \delta$. The following two facts are immediate consequences of axioms 2 and 3:

Triangle inequality: If $F \approx_\delta G$ and $G \approx_{\delta'} H$ then $F \approx_{\delta+\delta'} H$.

Fixing: If $F(X, Z) \approx_\delta G(X, Z)$ then $F(\underline{X}, Z) \approx_\delta G(\underline{X}, Z)$ for some fixed value $\underline{X}$.

3 Estimating the quality of a partition

In this section we show that $\|D_F(\boldsymbol{X}, \boldsymbol{Y})\|$ is an approximate estimator for the quality $\delta(\boldsymbol{X}, \boldsymbol{Y})$ of a decomposition, namely

$$\delta(\boldsymbol{X}, \boldsymbol{Y}) \leq \|D_F(\boldsymbol{X}, \boldsymbol{Y})\| \leq 4 \cdot \delta(\boldsymbol{X}, \boldsymbol{Y}). \tag{4}$$

The proof is given in Claims 5 and 6 below. As $\|D_F(\boldsymbol{X}, \boldsymbol{Y})\|$ can be estimated efficiently from oracle access to F (Claim 7), we obtain an algorithm for estimating the quality of a partition to within a factor of 4 in general, and exactly for the 2-norm over $\mathbb{R}$.

▶ **Proposition 4.** *Let $\|\cdot\|$ be either 1) $\|\cdot\|_{\mathbb{R},p}$ assuming $\|F\|_{\mathbb{R},2p} = O(1)$, or 2) the Hamming metric over $\mathbb{Z}_q$. There is an algorithm that given a bipartition $\boldsymbol{X}, \boldsymbol{Y}$ of the variables and parameters $\varepsilon, \gamma > 0$, outputs a value $\hat{\delta}$ such that*

$$\delta(\boldsymbol{X}, \boldsymbol{Y}) \leq \hat{\delta} \leq 4 \cdot \delta(\boldsymbol{X}, \boldsymbol{Y}) + \varepsilon,$$

with probability at least $1 - \gamma$ from $K^p \log(1/\gamma)/\epsilon^{2p}$ queries to F in time linear in the number of queries, for an absolute constant K.

The value of $\delta(\boldsymbol{X}, \boldsymbol{Y})$ is known to be NP-hard to calculate exactly over $\mathbb{Z}_2$ under the Hamming metric given explicit access to the truth-table of f [28]. Therefore some approximation factor is unavoidable for algorithms running in time polynomial in n and $1/\varepsilon$ unless BPP is in NP. On the positive side Karpinski and Schudy [19] give a fully polynomial-time randomized approximation scheme for this special case. Their algorithm requires at least linear time but it is plausible that a sublinear-time variant can be obtained. However, it appears unrelated to the dependence score D_F which plays an essential role in the results to follow.

The analysis of D_F applies to any pair of disjoint subsets $\boldsymbol{X}$, $\boldsymbol{Y}$ that do not necessarily partition all the variables. In this more general setting distance is measured by the formula

$$\delta(\boldsymbol{X}, \boldsymbol{Y}) = \min_{A,B} \|F(X, Y, Z) - A(X, Z) - B(Y, Z)\|. \tag{5}$$

▷ Claim 5 (Completeness of D_F). For all disjoint $\boldsymbol{X}$, $\boldsymbol{Y}$, $\|D_F(\boldsymbol{X}, \boldsymbol{Y})\| \leq 4 \cdot \delta(\boldsymbol{X}, \boldsymbol{Y})$.

Proof. By definition of $\delta(\boldsymbol{X}, \boldsymbol{Y})$ there exists a decomposition of the form

$$F(X, Y, Z) = A(X, Z) + B(Y, Z) + D(X, Y, Z),$$

where $\|D(X, Y, Z)\| = \delta(\boldsymbol{X}, \boldsymbol{Y})$. In the expansion of D_F all the A and B terms cancel out, leaving

$$\begin{aligned} \|D_F(\boldsymbol{X}, \boldsymbol{Y})\| &= \|D(X, Y, Z) + D(X', Y', Z) - D(X, Y', Z) - D(X', Y, Z)\| \\ &\leq \|D(X, Y, Z)\| + \|D(X', Y', Z)\| + \|D(X, Y', Z)\| + \|D(X', Y, Z)\| \\ &= 4\delta(\boldsymbol{X}, \boldsymbol{Y}). \end{aligned}$$

◁

Soundness for Boolean functions under the uniform measure was proved by David et al. [10]. We reproduce their proof under a more general setting.

▷ Claim 6 (Soundness of D_F). For all disjoint $\boldsymbol{X}$, $\boldsymbol{Y}$, $\delta(\boldsymbol{X}, \boldsymbol{Y}) \leq \|D_F(\boldsymbol{X}, \boldsymbol{Y})\|$.

Proof. Let $\varepsilon = \|D_F(\boldsymbol{X}, \boldsymbol{Y})\|$. Then

$$F(X, Y, Z) \approx_\varepsilon F(X, Y', Z) - F(X', Y, Z) - F(X', Y', Z).$$

We can fix values $\underline{X}'$ and $\underline{Y}'$ for which

$$F(X, Y, Z) \approx_\varepsilon F(\underline{X}', \underline{Y}', Z) - F(X, \underline{Y}', Z) - F(\underline{X}', Y, Z) = A(X, Z) + B(Y, Z),$$

where $A(X, Z) = F(\underline{X}', \underline{Y}', Z) - F(X, \underline{Y}', Z)$ and $B(Y, Z) = F(\underline{X}', Y, Z)$. ◁

Proposition 4 now follows from inequality (4) and the following claim, which states that $\|D_F\|$ can be estimated by sampling in the cases of interest. See Appendix A for the proof.

▷ Claim 7. Assuming $\|F\|_{\mathbb{R},2p} \leq 1$, the value $\|F\|_{\mathbb{R},p}$ can be estimated within ε from $K^p \log(1/\gamma)/\epsilon^{2p}$ (random) queries to F in linear time with probability $1-\gamma$ for some absolute constant K.

3.1 Exact partitioning under the 2-norm

Since computing the optimal partition is in general NP-complete, we do not expect to replace the inequalities in (4) with an equality. However, in the special case of real-valued functions with ℓ^2-norm, the estimate becomes exact:

$$\|D_F(\boldsymbol{X}, \boldsymbol{Y})\|_{\mathbb{R},2} = 2 \cdot \delta_{\mathbb{R},2}(\boldsymbol{X}, \boldsymbol{Y}). \tag{6}$$

This equality is a consequence of the following characterization of $\delta_{\mathbb{R},2}$, which applies more generally to k-partitions:

▶ **Proposition 8.** *Assuming* $\mathbb{E}[F] = 0$*, the* k*-partition* $F_i(X_i) = \mathbb{E}[F|X_i]$ *achieves the minimum for* $\delta_{\mathbb{R},2}(\boldsymbol{X}_1, \ldots, \boldsymbol{X}_k)$*.*

In particular it follows that $\delta_{\mathbb{R},2}$ takes the value

$$\delta_{\mathbb{R},2}(\boldsymbol{X}_1, \ldots, \boldsymbol{X}_k) = \mathbb{E}[(\overline{F} - \mathbb{E}[\overline{F}|X_1] - \cdots - \mathbb{E}[\overline{F}|X_k])^2], \tag{7}$$

where $\overline{F} = F - \mathbb{E}[F]$. To derive identity (6) it remains to verify that when $k = 2$, the right-hand side of (7) is a quarter of $\|D_F\|^2$:

▶ **Fact 9.** $\|D_F(\boldsymbol{X}, \boldsymbol{Y})\|^2_{\mathbb{R},2} = 4 \cdot \mathbb{E}[(\overline{F} - \mathbb{E}[\overline{F}|X] - \mathbb{E}[\overline{F}|Y])^2]$.

Armed with this fact we prove the proposition.

Proof of Proposition 8. First assume $F(\boldsymbol{X}, \boldsymbol{Y})$ is bivariate. Let $A(\boldsymbol{X})$ be any function. The inequality $\mathbb{E}[(\mathbb{E}[F|X] - A(X))^2] \geq 0$ can be rewritten as

$$\mathbb{E}[(F - \mathbb{E}[F|X])^2] \leq \mathbb{E}[(F - A(X))^2], \tag{8}$$

stating that the orthogonal projection of F onto the subspace of functions that depend only on $\boldsymbol{X}$ in 2-norm is $\mathbb{E}[F|X]$.

Now let $F(\boldsymbol{X}_1, \ldots, \boldsymbol{X}_k)$ be k-variate. Assume $\mathbb{E}[F] = 0$ and $\mathbb{E}[F_i(X_i)] = 0$ for all i. Then $\mathbb{E}[F_i(X_i)|X_j] = 0$ for all $i \neq j$. Applying inequality (8) for k times in succession together with this fact, we obtain

$$\begin{aligned}
&\mathbb{E}[(F - F_1(X_1) - \cdots - F_{k-1}(X_{k-1}) - F_k(X_k))^2] \\
&\quad \geq \mathbb{E}[(F - F_1(X_1) - \cdots - F_{k-1}(X_{k-1}) - \mathbb{E}[F - F_1(X_1) - \cdots - F_{k-1}(X_{k-1})|X_k])^2] \\
&\quad = \mathbb{E}[(F - F_1(X_1) - \cdots - F_{k-1}(X_{k-1}) - \mathbb{E}[F|X_k])^2] \\
&\quad \vdots \\
&\quad \geq \mathbb{E}[(F - \mathbb{E}[F|X_1] - \cdots - \mathbb{E}[F|X_k])^2]
\end{aligned}$$

as desired. Finally, by orthogonality the optimal decomposition must satisfy $\sum \mathbb{E}[F_i(X_i)] = 0$ so the assumption $\mathbb{E}[F_i(X_i)] = 0$ can be made without loss of generality. ◀

By orthogonality, equation (7) can also be written in the following forms:

$$\begin{aligned}
\delta_{\mathbb{R},2}(\boldsymbol{X}_1, \ldots, \boldsymbol{X}_k) &= \mathbb{E}\left[\overline{F}^2\right] - \sum\nolimits_{i=1}^{k} \mathbb{E}\left[\mathbb{E}[\overline{F}|X_i]^2\right] \\
&= \mathbb{E}_X\left[\overline{F}(X)^2\right] - \sum_{i=1}^{k} \mathbb{E}_{X,X'}\left[\overline{F}(X_{-i}, X_i)\overline{F}(X'_{-i}, X_i)\right],
\end{aligned} \tag{9}$$

where (X_{-i}, X_i) is the input whose i-th variable takes value X_i and j-th variable takes value X'_j for $j \neq i$. As all these terms can be efficiently estimated, we obtain the following algorithm for estimating the quality of a given k-partition:

▶ **Proposition 10.** *There is an algorithm that given a* k*-partition* $\boldsymbol{X}_1, \cdots, \boldsymbol{X}_k$ *of the variables and parameters* $\varepsilon, \gamma > 0$*, outputs a value* $\hat{\delta}$ *such that*

$$|\hat{\delta}^2 - \delta_{\mathbb{R},2}(\boldsymbol{X}_1, \ldots, \boldsymbol{X}_k)^2| \leq \varepsilon,$$

with probability at least $1 - \gamma$ *from* $O(k \log(k/\gamma)/\epsilon^4)$ *queries to* F *in time linear in the number of queries.*

4 Variable partitioning over general groups

In this section we present our first partitioning algorithm, which is general enough to work on any normed group G assuming it is possible to efficiently estimate the quantity $\|D_F(\{\boldsymbol{x}\},\{\boldsymbol{y}\})\|$.

The algorithm is based on the pairwise estimates of dependency over sets of single variables. The intuition behind the algorithm is that if the dependency between $\boldsymbol{x}$ and $\boldsymbol{y}$ is low, then these two variables should be assigned to different partitions. Therefore the algorithm keeps asserting such "in different partitions" for the pairs with lowest dependency estimates, until the k-partitioning can be clearly observed from the assertions.

This idea of the algorithm has been used in previous works in reinforcement learning control [23, 36] in a heuristic way. Theorem 1 below shows that the algorithm outputs an $O(kn^2)$ approximation to the optimal partition in time polynomial in n, k, and $1/\varepsilon$.

Algorithm 1 Approximate partitioning via pairwise estimates.

Input: number of partitions k
Output: partition $\mathcal{P}$
For every pair of variables $\boldsymbol{x},\boldsymbol{y} \in \boldsymbol{V}$, find estimate $\hat{e}(\boldsymbol{x},\boldsymbol{y})$ for $e(\boldsymbol{x},\boldsymbol{y}) = \|D_F(\{\boldsymbol{x}\},\{\boldsymbol{y}\})\|$;
Create a weighted graph with vertices $\boldsymbol{V}$ and weights $\hat{e}(\boldsymbol{x},\boldsymbol{y})$;
Order the edges in increasing weight;
repeat
    Remove the edge with the smallest weight;
until The graph has exactly k connected components

▶ **Proposition 11.** *Assuming $e(\boldsymbol{x},\boldsymbol{y}) \le \hat{e}(\boldsymbol{x},\boldsymbol{y}) \le e(\boldsymbol{x},\boldsymbol{y}) + \varepsilon$ for all $\boldsymbol{x}$ and $\boldsymbol{y}$,*

$$\delta(\mathcal{P}) \le (8k-10)n^2(4\delta_k(F)+\varepsilon). \tag{10}$$

If the estimates $\hat{e}(\boldsymbol{x},\boldsymbol{y})$ are obtained by empirical averaging, we obtain Theorem 1.

4.1 Proof of Proposition 11

For a partition $\mathcal{P}$ of the variables, let $\Delta(\mathcal{P}) = \sum \delta(\{\boldsymbol{x}\},\{\boldsymbol{y}\})$, where the sum is taken over all pairs that cross the partition. We will deduce Theorem 1 from the following bound on $\delta(\mathcal{P})$.

▷ Claim 12. For every k-partition $\mathcal{P}$, $\delta(\mathcal{P}) \le (16k-20)\Delta(\mathcal{P})$.

The following fact is immediate from the definitions of δ. The proof of this claim is delayed to the end of this section.

▶ **Fact 13.** *For any partition $(\boldsymbol{U},\overline{\boldsymbol{U}})$ such that $\boldsymbol{X} \subseteq \boldsymbol{U}$ and $\boldsymbol{Y} \subseteq \overline{\boldsymbol{U}}$, $\delta(\boldsymbol{X},\boldsymbol{Y}) \le \delta(\boldsymbol{U},\overline{\boldsymbol{U}})$.*

Now we prove the theorem, assuming the correctness of Claim 12.

▶ **Theorem 1.** *Let $\|\cdot\|$ be either 1) $\|\cdot\|_{\mathbb{R},p}$ assuming $\|F\|_{\mathbb{R},2p} = O(1)$, or 2) the Hamming metric over $\mathbb{Z}_q$. There is an algorithm that given parameters n, k, ε, γ, and oracle access to $F\colon \Sigma^n \to \mathbb{R}$ outputs a k-partition $\mathcal{P}$ such that $\delta(\mathcal{P}) \le O(kn^2)(\delta_k(F)+\varepsilon)$ with probability at least $1-\gamma$. The algorithm makes $O(K^p n^2 \log(n/\gamma)/\varepsilon^{2p})$ queries to F and runs in time linear in the number of queries, for an absolute constant K.*

Proof of Theorem 1. By Claim 5 and Fact 13, all edges $(\boldsymbol{x}, \boldsymbol{y})$ in the optimal partition must satisfy $e(\boldsymbol{x}, \boldsymbol{y}) \leq 4\delta_2(F)$. By our assumption on the quality of the approximations,

$$\hat{e}(\boldsymbol{x}, \boldsymbol{y}) \leq 4\delta_2(F) + \varepsilon. \tag{11}$$

Since the algorithm removes edges in increasing order of weight, all the edges that cross the output partition $\mathcal{P}$ must also satisfy this inequality. Then

$$\begin{aligned}
\delta(\mathcal{P}) &\leq (16k-20)\Delta(\mathcal{P}) && \text{by Claim 12,}\\
&\leq (16k-20)\sum\nolimits_{\boldsymbol{x},\,\boldsymbol{y} \text{ cross } \mathcal{P}} e(\boldsymbol{x}, \boldsymbol{y}) && \text{by Claim 6,}\\
&\leq (16k-20)\sum\nolimits_{\boldsymbol{x},\,\boldsymbol{y} \text{ cross } \mathcal{P}} \hat{e}(\boldsymbol{x}, \boldsymbol{y})\\
&\leq (16k-20)\sum\nolimits_{\boldsymbol{x},\,\boldsymbol{y} \text{ cross } \mathcal{P}} 4\delta_2(F) + \varepsilon && \text{by (11),}\\
&\leq (8k-10)n^2 \cdot (4\delta_2(F) + \varepsilon).
\end{aligned}$$

The last inequality holds because there are at most $\binom{n}{2} \leq n^2/2$ pairs of variables crossing the partition. ◀

We first prove Claim 12 in the case $k = 2$ of bipartitions. This is Claim 15 below. We use $\boldsymbol{X}\boldsymbol{X}'$ to denote the union of the variable sets $\boldsymbol{X}$ and $\boldsymbol{X}'$.

▷ Claim 14. For disjoint sets of variables $\boldsymbol{X}, \boldsymbol{X'}, \boldsymbol{Y}$, $\delta(\boldsymbol{X}\boldsymbol{X'}, \boldsymbol{Y}) \leq \delta(\boldsymbol{X}, \boldsymbol{Y}) + 2\delta(\boldsymbol{X'}, \boldsymbol{Y})$.

Proof. Assume that

$$\begin{aligned}
&F(X, X', Y) \approx_\delta A(X, X') + B(X', Y) \qquad \text{and}\\
&F(X, X', Y) \approx_{\delta'} A'(X, X') + B'(X, Y).
\end{aligned}$$

By the triangle inequality,

$$A(X, X') + B(X', Y) \approx_{\delta+\delta'} A'(X, X') + B'(X, Y).$$

Fix $X'(Z) = \underline{X}'(Z)$. Writing $C(X) = A(X, \underline{X}') - A'(X, \underline{X}')$ and $D(Y') = B(\underline{X}', Y')$ we get that

$$B'(X, Y) \approx_{\delta+\delta'} C(X) + D(Y).$$

By the triangle inequality (with the second equation), we get that

$$F(X, X', Y) \approx_{\delta+2\delta'} A'(X, X') + C(X) + D(Y).$$ ◁

▷ Claim 15. For every bipartition $\boldsymbol{X}, \overline{\boldsymbol{X}}$ of the variables, $\delta(\boldsymbol{X}, \overline{\boldsymbol{X}}) \leq 4 \cdot \Delta(\boldsymbol{X}, \overline{\boldsymbol{X}})$.

Proof. By Claim 14,

$$\delta(\boldsymbol{X}'\{\boldsymbol{x}\}, \{\boldsymbol{y}\}) \leq \delta(\boldsymbol{X}', \{\boldsymbol{y}\}) + 2\delta(\{\boldsymbol{x}\}, \{\boldsymbol{y}\})$$

for all $\boldsymbol{X}' \subseteq \boldsymbol{X} \setminus \{x\}$ and $\boldsymbol{y}$. Applying this inequality iteratively we conclude that $\delta(\boldsymbol{X}, \{\boldsymbol{y}\}) \leq 2\sum_{\boldsymbol{x}\in\boldsymbol{X}} \delta(\{\boldsymbol{x}\}, \{\boldsymbol{y}\})$. Also by Claim 14

$$\delta(\boldsymbol{X}, \boldsymbol{Y}'\{\boldsymbol{y}\}) \leq \delta(\boldsymbol{X}, \boldsymbol{Y}') + 2\delta(\boldsymbol{X}, \boldsymbol{Y}'\{\boldsymbol{y}\}),$$

so $\delta(\boldsymbol{X}, \boldsymbol{Y}) \leq 2\sum_{\boldsymbol{y}\in\boldsymbol{Y}} \delta(\boldsymbol{X}, \{\boldsymbol{y}\})$. Combining the two inequalities we obtain the desired conclusion. ◁

To extend the proof to larger k and obtain Claim 12, we generalize the first inequality in this sequence to k-partitions.

▷ Claim 16. For every $2k$-partition $(\boldsymbol{Y}_1, \ldots, \boldsymbol{Y}_k, \boldsymbol{Z}_1, \ldots, \boldsymbol{Z}_k)$,

$$\delta(\boldsymbol{Y}_1, \ldots, \boldsymbol{Y}_k, \boldsymbol{Z}_1, \ldots, \boldsymbol{Z}_k) \leq 2\delta(\boldsymbol{Y}_1\boldsymbol{Z}_1, \ldots, \boldsymbol{Y}_k\boldsymbol{Z}_k) + 3\delta(\boldsymbol{Y}_1 \ldots \boldsymbol{Y}_k, \boldsymbol{Z}_1 \ldots \boldsymbol{Z}_k).$$

Proof. Assume that

$$F(V) \approx_\delta F_1(Y_1, Z_1) + \cdots + F_t(Y_t, Z_t)$$
$$F(V) \approx_{\delta'} A(Y_1, \ldots, Y_t) + B(Z_1, \ldots, Z_t).$$

By the triangle inequality

$$A(Y_1, \ldots, Y_t) + B(Z_1, \ldots, Z_t) \approx_{\delta+\delta'} F_1(Y_1, Z_1) + \cdots + F_t(Y_t, Z_t).$$

Fixing $Z_1, \ldots, Z_t$ to values $\underline{Z}_1, \ldots, \underline{Z}_t$ we get the decomposition

$$A(Y_1, \ldots, Y_t) \approx_{\delta+\delta'} F_1(Y_1, \underline{Z}_1) + \cdots + F_t(Y_t, \underline{Z}_t) - B(\underline{Z}_1, \ldots, \underline{Z}_t).$$

and similarly

$$B(Z_1, \ldots, Z_t) \approx_{\delta+\delta'} F_1(\underline{Y}_1, Z_1) + \cdots + F_t(\underline{Y}_t, Z_t) - A(\underline{Y}_1, \ldots, \underline{Y}_t).$$

Plugging these into the second equation gives the desired decomposition. ◁

Proof of Claim 12. We assume that k is a power of two and prove by induction that $\delta(\mathcal{P}) \leq c_k\Delta(\mathcal{P})$, where c_k is the sequence $c_{2k} = 2c_k + 12$, $c_2 = 4$. The base case $k = 2$ follows from Claim 15. Assume the claim holds for k and apply Claim 16 to $\mathcal{P}$. By inductive assumption and Claim 15,

$$\delta(\mathcal{P}) \leq 2 \cdot c_k\Delta(\boldsymbol{Y}_1\boldsymbol{Z}_1, \ldots, \boldsymbol{Y}_k\boldsymbol{Z}_k) + 3 \cdot 4\Delta(\boldsymbol{Y}_1 \ldots \boldsymbol{Y}_k, \boldsymbol{Z}_1 \ldots \boldsymbol{Z}_k).$$

Since $\mathcal{P}$ is a refinement of both these partitions, it follows that $\delta(\mathcal{P}) \leq (2c_k + 12)\Delta(\mathcal{P}) = c_{2k}\Delta(\mathcal{P})$, concluding the induction.

The recurrence solves to $c_k = 8k - 12$, proving the claim when k is a power of two. When it is not, the same reasoning applies to the closest power of two exceeding k (by taking some of the sets in the partition to be empty), which is at most $2k - 1$, proving the desired bound. ◁

4.2 Hardness of exact variable bipartitioning

In contrast to Theorem 1, finding the exact bipartition is NP-hard even when there are only three variables.

▶ **Proposition 17.** *For any $n \geq 3$ there is no algorithm that outputs a bipartition of cost $\delta_2(F) + \varepsilon$ over $\mathbb{Z}_2$ under Hamming distance in time polynomial in $|\Sigma|/\varepsilon$ with constant probability unless BPP is in NP.*

Proof. Assume such an algorithm $BIPARTITION$ exists. We show it can be used to solve the following problem: Given explicit functions $F_1, \ldots, F_{n-1} \colon \Sigma^2 \to G$, find i^* that minimizes $\delta_2(F_{i^*})$ assuming this i^* is unique, i.e. a function that has the smallest bipartition cost among the candidates.

Let $F(x_1, \ldots, x_{n-1}, y) = F_1(x_1, y) + \cdots + F_{n-1}(x_{n-1}, y)$. The cost of the bipartition that splits x_i from all other variables in F is at most the cost of F_i, so F has a bipartition of cost at least $\delta_2(F_{i^*})$. We now argue that the cost of all other bipartitions is greater. In fact, any other bipartition must split y from x_i for some $i \neq i^*$. Assuming the cost of this partition is δ, we must have

$$F(X, Y) \approx_\delta A(X) + B(Y),$$

where $x_i \in X$ and $y \in Y$. After fixing all variables except for x_i and y we obtain

$$F_i(x_i, y) \approx_\delta A(x_i, \underline{X_{-i}}) + B(y, \underline{Y_{-i}}) - \sum_{j \neq i} F_j(\underline{x_j}, y).$$

This is a bipartition for F_i so its cost is strictly greater than $\delta_2(F_{i^*})$. Therefore the output of $BIPARTITION$ with oracle access to F and $\varepsilon = 1/|\Sigma|^2$ has the desired property.

Roth and Viswanathan [28] give an efficient reduction R that maps a graph G into a function F such that if G has a larger max-cut than G' then $\delta_2(R(G)) < \delta_2(R(G'))$. By composing the two reductions we obtain an efficient algorithm for deciding which of two graphs has a larger maximum cut, which is an NP-hard problem. ◀

5 Partitioning real-valued functions under the 2-norm

The problem of partitioning real-valued functions under the 2-norm is closely related to the well-studied problem of hypergraph partitioning. To explain this connection we recall the Efron-Stein decomposition of real-valued functions over product sets. The Efron-Stein decomposition of a function $F\colon \Sigma^n \to \mathbb{R}$ (under some product measure) is the unique decomposition of the form

$$F(x) = \sum\nolimits_{S \subseteq [n]} \hat{F}_S \cdot F_S(x),$$

where $\hat{F}_S$ are real coefficients and F_S are functions satisfying the following properties:

1. F_S depends on the variables in S only;
2. $\mathbb{E}[F_S | x_{-i}] = 0$, where x_{-i} is a fixing of all variables except the i-th one;
3. $\mathbb{E}[F_S^2] = 1$.

In particular, properties 1 and 2 imply that $\mathbb{E}[F_S F_T] = 0$ when $S \neq T$, and so $\mathbb{E}[F^2] = \sum_S \hat{F}_S^2$ by property 3.

▶ **Proposition 18.** *Given $F\colon \Sigma^n \to \mathbb{R}$, let H be the hypergraph whose vertices are the variables of F and whose hyperedges S have weight $\hat{F}_S^2$ for every subset S. The cost of the k-cut $(\boldsymbol{X}_1, \ldots, \boldsymbol{X}_k)$ in H equals $\delta_{\mathbb{R},2}(\boldsymbol{X}_1, \ldots, \boldsymbol{X}_k)^2$.*

Proof. We may assume $\mathbb{E}[F] = 0$ and use expression (9) to evaluate $\delta_{\mathbb{R},2}$. The first term equals $\mathbb{E}[F^2] = \sum_S \hat{F}_S^2$. The rest of the terms have the form $\mathbb{E}[\mathbb{E}[F|X_I]^2] = \mathbb{E}[F(X_{-I}, X_I)F(X'_{-I}, X_I)]$ for subsets I of variables. Plugging in the Efron-Stein decomposition of F we have

$$\mathbb{E}[\mathbb{E}[F|X_I]^2] = \sum_{S,T} \hat{F}_S \hat{F}_T \mathbb{E}[F_S(X_{-I}, X_I) F_T(X'_{-I}, X_I)].$$

By property 2, the terms in the summation in which $S \neq T$ evaluate to zero. Among the rest, if the set S contains any variable i outside I then

$$\mathbb{E}[F_S(X_{-I}, X_I) F_T(X'_{-I}, X_I)] = \mathbb{E}\big[F_S(X_{-I}, X_I)\mathbb{E}[F_T(X'_{-I}, X_I)|X, X'_{-i}]\big]$$

and the inside expectation evaluates to zero by property 2. Therefore the only surviving terms are those where $S = T$ and $S \subseteq I$, from where

$$\mathbb{E}[\mathbb{E}[F|X_I]^2] = \sum\nolimits_{S \subseteq I} \hat{F}_S^2.$$

By (9),

$$\delta_{\mathbb{R},2}(\boldsymbol{X}_1, \ldots, \boldsymbol{X}_k) = \sum\nolimits_S \hat{F}_S^2 - \sum\nolimits_{i=1}^k \sum\nolimits_{S \subseteq \boldsymbol{X}_i} \hat{F}_S^2 = \sum\nolimits_{S \not\subseteq \boldsymbol{X}_i \text{ for any } i} \hat{F}_S^2.$$

The last quantity is the desired value of the cost of k-cut in H. ◀

When $\Sigma = \{-1, 1\}$ under uniform measure, the functions F_S do not depend on F and equal the Fourier characters $\chi_S(x) = \prod_{i \in S} x_i$, allowing us to embed instances of hypergraph partitioning into variable partitioning.

▶ **Corollary 19.** *Assume there is an algorithm A that given oracle access to $F\colon \{-1,1\}^n \to \mathbb{R}$ under uniform measure outputs a k-variable partition of cost at most $C \cdot \delta_2(F) + \varepsilon$ in time $t(n, k, \varepsilon)$. Then given a hypergraph with n vertices and m hyperedges with a k-cut of value opt, it is possible to output a k-cut of value $C \cdot opt$ in time $mnt(n, k, 1/m)$.*

Chekuri and Li [7] give a reduction from hypergraph k-cut to densest-k-subgraph. Manurangsi [24] shows that the latter is hard to approximate to within $O(n^{1/(\log\log n)^c})$ assuming the exponential-time hypothesis, implying inapproximability of the same order for $\delta_{\mathbb{R},2}(F)$.

On the positive side, Proposition 18 can be used to obtain variable partitioning algorithms from hypergraph partitioning ones. The conversion is not direct as hypergraph partitioning assumes explicit access to the hypergraph. Klimmek and Wagner [21] observed that submodularity of the hypergraph cut function $f(\boldsymbol{X}) = \delta_{\mathbb{R},2}(\boldsymbol{X}, \overline{\boldsymbol{X}})^2$ allows for efficient minimization from *exact* oracle access. To derive Theorem 2 we extend the analysis to approximate oracle access.

The following proposition is an analysis of Queyranne's symmetric submodular minimization algorithm [27] for an approximate input oracle. We say g is ε-submodular if $g(\boldsymbol{XYZ}) - g(\boldsymbol{XZ}) - g(\boldsymbol{YZ}) + g(\boldsymbol{Z}) \leq \varepsilon$ for all disjoint subsets $\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z}$.

▶ **Proposition 20** (Queyranne's algorithm with an approximate oracle)**.** *There is an algorithm that given oracle access to a symmetric ε-submodular g, makes $O(n^3)$ oracle queries and outputs a nontrivial subset $\boldsymbol{X}$ such that $g(\boldsymbol{X})$ is within $n\varepsilon/2$ of the minimum of g.*

▶ **Theorem 2.** *Let $F\colon \Sigma^n \to \mathbb{R}$ be a function with $\|F\|_{\mathbb{R},4} \leq 1$. There is an algorithm that given inputs n, ε, γ, and oracle access to F, runs in time $O(n^5 \log(n/\gamma)/\varepsilon^2)$ and outputs a bipartition $(\boldsymbol{X}, \overline{\boldsymbol{X}})$ such that $\delta_{\mathbb{R},2}(\boldsymbol{X}, \overline{\boldsymbol{X}})^2 \leq \delta_{\mathbb{R},2}(F)^2 + \varepsilon$ with probability at least $1 - \gamma$.*

Proof of Theorem 2. By Proposition 10, $\delta_{\mathbb{R},2}(\boldsymbol{X}, \overline{\boldsymbol{X}})^2$ can be estimated to within error ε/Kn with $O(\log(n/\gamma)n^2/\varepsilon^2)$ queries to F with probability $1 - K\gamma/n^3$ for any constant K. This estimator implements an $\varepsilon/2n$-approximate oracle to $\delta_{\mathbb{R},2}(\boldsymbol{X}, \overline{\boldsymbol{X}})^2$ with probability $1-\gamma$ with respect to an algorithm that makes at most Kn^3 queries. In particular, with probability $1 - \gamma$, the output of the oracle is $\varepsilon/4n$-close to the value of the submodular function $\delta_{\mathbb{R},2}^2$ at all points queried by Queyranne's algorithm and also at the minimum of $\delta_{\mathbb{R},2}^2$. Since from the algorithm's perspective it is interacting with a symmetric ε/n-submodular function g, it outputs a partition such that $g(\boldsymbol{X}, \overline{\boldsymbol{X}})$ is within $\varepsilon/2$ of the minimum of g. By the triangle inequality, $\delta_{\mathbb{R},2}(\boldsymbol{X}, \overline{\boldsymbol{X}})^2$ is within $\varepsilon/2 + 2\varepsilon/4n \leq \varepsilon$ close to the minimum of $\delta_{\mathbb{R},2}^2$. ◀

Saran and Vazirani's approximation algorithm for multiway k-cut (with fixed terminals) can be viewed as a reduction from multiway k-cut to multiway 2-cut. The reduction works given access to approximate s-t-cut oracles, where s and t are designated terminals that must be split by the cut. The corresponding cut function $\delta_{\mathbb{R},2}(\boldsymbol{s}\boldsymbol{X}, \boldsymbol{t}\overline{\boldsymbol{X}})$, where $(\boldsymbol{X}, \overline{\boldsymbol{X}})$ is now a partition of $\boldsymbol{V} - \boldsymbol{s}, \boldsymbol{t}$, is still submodular but no longer symmetric. We believe that an analogue of Proposition 20 for general (non-symmetric) submodular minimization [14, 15] should hold, but were unable to verify the claim. We state the extension of Theorem 2 to k-partitions as a conditional result.

▶ **Corollary 21.** *Assume that a ε-approximate minimum of an n-variate function f that is $\mathrm{poly}(\varepsilon/n)$-close to submodular on every input can be found in time $\mathrm{poly}(n/\varepsilon)$ given oracle access to it. Then there is an algorithm that given inputs k, ε and oracle access to a function F such that $\|F\|_{\mathbb{R},4} \le 1$ runs in time $k^2 n^k \,\mathrm{poly}(n/\varepsilon)\log(1/\gamma)$ and outputs a k-partition $\mathcal{P}$ such that $\delta_{\mathbb{R},2}(\mathcal{P})^2 \le (2 - 2/k)\delta_{\mathbb{R},2}(F)^2 + \varepsilon$ with probability $1 - \gamma$.*

It remains to prove Proposition 20.

▷ Claim 22. Let g be ε-submodular. Assume there exists $\boldsymbol{x} \in \boldsymbol{W}$ such that for all $\boldsymbol{Y} \subseteq \boldsymbol{W} \setminus \boldsymbol{x}$ and $\boldsymbol{u} \notin \boldsymbol{W}$,

$$g(\boldsymbol{W}) + g(\boldsymbol{u}) \le g(\boldsymbol{W} \setminus \boldsymbol{Y}) + g(\boldsymbol{Y}\boldsymbol{u}) + \delta.$$

If $\boldsymbol{x}'$ maximizes $g(\boldsymbol{W}\boldsymbol{u}) - g(\boldsymbol{u})$ among all $\boldsymbol{u} \notin \boldsymbol{W}$ then

$$g(\boldsymbol{W}\boldsymbol{x}') + g(\boldsymbol{u}) \le g(\boldsymbol{W}\boldsymbol{x}' \setminus \boldsymbol{Y}) + g(\boldsymbol{Y}\boldsymbol{u}) + (\delta + \varepsilon).$$

Proof. If $\boldsymbol{x} \notin \boldsymbol{Y}$ then

$$\begin{aligned} g(\boldsymbol{W}\boldsymbol{x}') + g(\boldsymbol{u}) &\le \big(g(\boldsymbol{W}) - g(\boldsymbol{W} \setminus \boldsymbol{Y}) + g(\boldsymbol{W}\boldsymbol{x}' \setminus \boldsymbol{Y})\big) + g(\boldsymbol{u}) + \varepsilon && \text{by } \varepsilon\text{-submodularity} \\ &= g(\boldsymbol{W}\boldsymbol{x}' \setminus \boldsymbol{Y}) + \big(g(\boldsymbol{W}) - g(\boldsymbol{W} \setminus \boldsymbol{Y}) + f(\boldsymbol{u})\big) + \varepsilon \\ &\le g(\boldsymbol{W}\boldsymbol{x}' \setminus \boldsymbol{Y}) + g(\boldsymbol{Y}\boldsymbol{u}) + (\delta + \varepsilon) && \text{by inductive hypothesis.} \end{aligned}$$

Otherwise, $\boldsymbol{x} \notin \boldsymbol{W} \setminus \boldsymbol{Y}$ and

$$\begin{aligned} g(\boldsymbol{W}\boldsymbol{x}') + g(\boldsymbol{u}) &\le g(\boldsymbol{W}\boldsymbol{u}) + g(\boldsymbol{x}') && \text{by optimality of } \boldsymbol{x}' \\ &\le \big(g(\boldsymbol{W}) - g(\boldsymbol{Y}) + g(\boldsymbol{Y}\boldsymbol{u})\big) + g(\boldsymbol{x}') + \varepsilon && \text{by } \varepsilon\text{-submodularity} \\ &= \big(g(\boldsymbol{W}) + g(\boldsymbol{x}') - g(\boldsymbol{Y})\big) + g(\boldsymbol{Y}\boldsymbol{u}) + \varepsilon \\ &\le g(\boldsymbol{W}\boldsymbol{x}' \setminus \boldsymbol{Y}) + g(\boldsymbol{Y}\boldsymbol{u}) + (\delta + \varepsilon) && \text{by inductive hypothesis.} \end{aligned}$$

◁

Proof of Proposition 20. Queyranne's algorithm Q^g is recursive. If $n = 2$ the unique partition is output. Otherwise, starting from an arbitrary singleton set $\boldsymbol{W}_1$, the algorithm sets $\boldsymbol{W}_{i+1} = \boldsymbol{W}_i\boldsymbol{x}_i$, where $\boldsymbol{x}_i$ maximizes $g(\boldsymbol{W}_i\boldsymbol{u}) - g(\boldsymbol{u})$ among all $\boldsymbol{u} \notin \boldsymbol{W}_i$. The algorithm then contracts the elements $\boldsymbol{x}_{n-1}$ and $\boldsymbol{x}_n$ into $\boldsymbol{x}_{n-1}\boldsymbol{x}_n$ and outputs the smaller value of $Q^g(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{n-2}, \boldsymbol{x}_{n-1}\boldsymbol{x}_n)$ and $g(\boldsymbol{x}_n)$.

We prove by induction on n that the output of Q^g is $(n-1)\varepsilon/2$-close to the minimum of g. The base case $n = 2$ is clear. Now assume this is true for inputs of size $n - 1$. If the minimum partition of g doesn't split $\boldsymbol{x}_{n-1}$ and $\boldsymbol{x}_n$ then the claim follows by inductive assumption.

Otherwise, we show that $g(\boldsymbol{x}_n)$ is within $(n-1)\varepsilon/2$-close to the minimum of g. Applying Claim 22 iteratively to the sets $\boldsymbol{W}_1, \ldots, \boldsymbol{W}_{n-1}$, we conclude that

$$g(\boldsymbol{W}_{n-1}) + g(\boldsymbol{x}_n) \le g(\boldsymbol{W}_{n-1} \setminus \boldsymbol{Y}) + g(\boldsymbol{Y}\boldsymbol{x}_n) + (n-1)\varepsilon$$

for all $\boldsymbol{Y}$ that do not contain $\boldsymbol{x}_n$ and $\boldsymbol{x}_{n-1}$. Applying symmetry this inequality can be rewritten as $g(\boldsymbol{x}_n) \leq g(\boldsymbol{x}_n Y) + (n-1)\varepsilon/2$. As $\boldsymbol{x}_{n-1}$ and $\boldsymbol{x}_n$ are split in the optimal solution it must be of type $\boldsymbol{x}_n Y$ for some Y excluding $\boldsymbol{x}_{n-1}$, so $g(\boldsymbol{x}_n)$ is $(n-1)/2\varepsilon$ close to the minimum as desired. ◀

6 Testing partitionability

As a consequence of Theorem 1, k-partitionability is testable with $\tilde{O}(k^{2p}n^{4p+2}/\varepsilon^{2p})$ queries. The yes instances are inputs with $\delta_k(F) = 0$; the no instances are inputs with $\delta_k(F) > \varepsilon$. The query complexity of Theorem 3 can be obtained by the improved analysis that follows.

To simplify notation we only prove the theorem for the Hamming weight over $\mathbb{Z}_q$ and describe the change necessary for p-norms over $\mathbb{R}$.

Algorithm 2 Tester for k-partitionability.

1: Create an empty undirected graph G with vertex set $\boldsymbol{V}$.
2: **for** $O(kn/\varepsilon)$ times **do**
3: **for** For every pair of distinct variables $\boldsymbol{x}, \boldsymbol{y} \in \boldsymbol{V}$ **do**
4: Choose random x, y, x', y', Z
5: If $F(x,y,Z) + F(x',y',Z) \neq F(x,y',Z) + F(x',y,Z)$ create the edge $\{\boldsymbol{x}, \boldsymbol{y}\}$ in G.
6: **end for**
7: **end for**
8: Accept if the graph has at least k connected components.

We will need the following fact:

▶ **Fact 23.** *If $p_1, \ldots, p_n$ are probabilities such that $\sum p_i \geq \varepsilon$ then $1 - \prod(1-p_i) \geq \varepsilon - O(\varepsilon^2)$.*

Proof. Using the inequality $1 - p \leq e^{-p}$ and the second-order estimate $e^{-x} = 1 - x + O(x^2)$, we have

$$\prod(1-p_i) \leq \prod e^{-p_i} = e^{-\sum p_i} \geq e^{-\varepsilon} = \varepsilon - O(\varepsilon^2).$$ ◀

▶ **Theorem 3.** *k-partitionability is testable with one-sided error and $O(kn^3/\varepsilon)$ non-adaptive queries with respect to Hamming weight over $\mathbb{Z}_q$, and with $O(k^{2p}n^3/\varepsilon^{2p})$ non-adaptive queries with respect to the p-norm over $\mathbb{R}$ assuming $\|F\|_{2p} \leq 1$.*

While in this work we are mainly interested in small values of k, in the extreme case when $k = n$ the partition is unique and the property is testable with $O(1/\varepsilon)$ queries by the result of Dinur and Golubev [12].

Proof of Theorem 3. The connected components of G are always contained in the partition components of F, so if F is k partitionable the tester always accepts. We argue that with constant probability, all bipartition of F satisfying $\delta(\boldsymbol{X}, \overline{\boldsymbol{X}}) \geq \varepsilon$ cross an edge in G.

If F is ε-far from k-partitionable, by Claim 12 $\Delta(\mathcal{P}) = \Omega(\varepsilon/k)$ for all k-partitions $\mathcal{P}$. As every k-cut can be coarsened into a 2-cut of at least half the weight, every k-partition can be coarsened into a bipartition such that $\Delta(\boldsymbol{X}, \overline{\boldsymbol{X}}) = \Omega(\varepsilon/k)$. We now argue that with constant probability, all such heavy bipartitions $(\boldsymbol{X}, \overline{\boldsymbol{X}})$ are crossed by an edge in G, so no k-partition is likely to survive in G.

Assume $\Delta(\boldsymbol{X}, \overline{\boldsymbol{X}}) = \sum_{\boldsymbol{x}\in\boldsymbol{X}, \boldsymbol{y}\in\boldsymbol{X}} \|D_F(\boldsymbol{x}, \boldsymbol{y})\| = \Omega(\varepsilon/k)$. As $\|D_F(\boldsymbol{x}, \boldsymbol{y})\|$ is the acceptance probability of the test in line 5, by Fact 23 in any given iteration of the outer loop 3 at least one of these edges will appear in G with probability $\Omega(\varepsilon/k)$. (For p-norms over $\mathbb{R}$,

$\|D_F(\boldsymbol{x}, \boldsymbol{y})\|$ is an expectation that takes $O((\varepsilon/k)^{2p})$ queries to estimate.) After $O(kn/\varepsilon)$ iterations the probability that the cut survives is less than 2^{-n}. By a union bound the probability that any heavy cut survives is at most half. ◀

It is not difficult to see that n queries are required for one-sided error testers when k equals 2 as the relevant constraints span an n-dimensional space. We show that a linear dependence on n is necessary for two-sided error testers as well. Proposition 24 shows a general $\Omega(n)$ lower bound for functions over finite domains (with uniform measure) valued over finite groups under the Hamming metric. Proposition 25 shows that $\Omega(n)$ *non-adaptive* queries are necessary for functions from $\mathbb{R}^n$ to $\mathbb{R}$ under the 2-norm.

▶ **Proposition 24.** *Testing 2-partitionability for functions $F\colon \mathbb{Z}_q^n \to G$ for a finite group G under uniform measure and Hamming metric requires $\Omega(n-k)$ queries even for constant ε.*

For simplicity of notation we present the proof in the case $q = 2$ and $G = \mathbb{Z}_2$. The proof is closely related to Chockler and Gutfreund's lower bound for testing juntas [9].

Proof. Let $R\colon \mathbb{Z}_2^n \to \mathbb{Z}_2$ be a random function and $P\colon \mathbb{Z}_2^{n-1} \to \mathbb{Z}_2$ be a function that depends on all but a random hidden input coordinate I. First we argue that $\delta_2(R) = \Omega(1)$ with high probability. For this it is sufficient to argue that $\|D_R(\boldsymbol{X}, \boldsymbol{Y})\| = \Omega(1)$ for every partition $(\boldsymbol{X}, \boldsymbol{Y})$. By definition $\|D_R(\boldsymbol{X}, \boldsymbol{Y})\|$ is the average value of $\Omega(2^{2n})$ indicator values for events of the type $R(x, y) + R(x', y') - R(x', y) - R(x, y') = 0$. These events have probability half each and are pairwise independent, so by Chebyshev's inequality the probability that the $\|D_R(\boldsymbol{X}, \boldsymbol{Y})\|$ is sub-constant is $\Omega(2^{-2n})$. Taking a union bound over all 2^n bipartitions it follows that $\|D_R(\boldsymbol{X}, \boldsymbol{Y})\| = \Omega(1)$ with probability at least $1 - \Omega(2^{-n})$.

To complete the proof, it is sufficient to argue that with high probability any Q queries to P reveal independent random bits. Consider the subspace of $\mathbb{Z}_2^n$ spanned by the Q queries (or the submodule of $\mathbb{Z}_q^n$ if q is not a prime). This vector space has dimension at most Q, so it can contain at most Q of the elementary basis vectors $e_1, \ldots, e_n$. However, unless it contains e_I for the hidden coordinate I, no two queries differ in a single coordinate and all answers are independent random bits. Since I is uniformly random the probability that P and R can be distinguished is at most Q/n. By a union bound the distinguishing advantage of the tester is at most $\Omega(2^{-n}) + Q/n$, which is subconstant unless $Q = \Omega(n)$. ◀

It was pointed out to us by Guy Kindler that the proof of Proposition 24 to functions from $\mathbb{R}^n$ to $\mathbb{R}$ say under Gaussian measure by considering the functions $R_{\mathbb{R}}$ and $P_{\mathbb{R}}$ given by $F_{\mathbb{R}}(x_1, \ldots, x_n) = F(\operatorname{sign} x_1, \ldots, \operatorname{sign} x_n)$ where the sign is interpreted as a Boolean value. This example is somewhat unnatural because the functions are discontinuous. The following proposition shows that testing still requires $\Omega(n)$ non-adaptive queries even for highly smooth functions.

▶ **Proposition 25.** *Testing 2-partitionability non-adaptively for quadratic functions from $\mathbb{R}^n$ to $\mathbb{R}$ under the 2-norm requires $\Omega(n)$ queries under any measure with zero mean and unit variance and bounded third and fourth moments.*

We need the following claim about distinguishing linear functions of normal random variables.

▷ Claim 26. Let $Z_1, \ldots, Z_n$ be independent standard normal random variables, $F(x) = \sum_{i=1}^n Z_i x_i$, and $F'(x) = \sum_{i \in S} Z_i x_i$ where $S \subseteq [n]$ is a random subset of size s. For any q queries $x^1, \ldots, x^q \in \mathbb{R}^n$, $(F(x^1), \ldots, F(x^q))$ and $(F'(x^1), \ldots, F'(x^q))$ are $O(qs/(n-s+1))$-statistically close.

Proof. It is sufficient to prove the claim for $|S| = n - 1$ and apply the triangle inequality. By convexity it is sufficient to upper bound the expected statistical distance averaged over the choice of the index i that is omitted from S. For fixed i, since the queried functions are linear, without loss of generality we may assume that the queries $x^1, \ldots, x^q$ are orthonormal. Let X be the $q \times n$ matrix whose rows are the queries $x^1, \ldots, x^n$, and X_{-i} be the submatrix obtained by removing the i-th column. The desired statistical distance is then within a constant factor of $\|(X^T X)^{-1}(X_{-i}^T X_{-i}) - I\|_F$, where $\|\cdot\|_F$ is the Frobenius norm [1]. By orthonormality we obtain that $\|(X^T X)^{-1}(X_{-i}^T X_{-i}) - I\|_F = \|x^i\|_2^2$. Averaging over i, the desired statistical distance is at most $O(\mathbb{E}_i[\|x^i\|_2^2]) = O(q/n)$. ◁

Proof of Proposition 25. Let $F(x) = n^{-1} \sum_{j \neq k} Z_{jk} x_j x_k$, where Z_{jk} are independent standard normal random variables. Let $F'(x) = n^{-1} \sum_{j,\, k,\, i \text{ distinct}} Z_{jk} x_j x_k$ where i is chosen at random from $[n]$. By standard concentration inequalities both $\|F\|_4$ and $\|F'\|_4$ are constant with high probability. By Claim 26, the answers to any q non-adaptive queries to F and F' are $O(q/n)$-statistically close.

It remains to argue that F is $\Omega(1)$-far from 2-partitionable. For a fixed bipartition (S, T) of $[n]$, by Claim 26 the cost of F is $n^{-1} \sum_{j \in S, k \in T} Z_{jk}^2$. Therefore the average cost (over the randomness of F) is $|S||T|/n$. By Chernoff bound the cost is at least $\Omega(|S||T|/n) = \Omega(1)$ with probability $1 - \exp(-\Omega(|S||T|))$. Taking a union bound over all 2^{n-1} possible bipartitions we conclude that F is $\Omega(1)$-far from 2-partitionable with probability $1 - \exp(-\Omega(n))$. ◀

If Claim 26 extends to adaptive queries, so would Proposition 25.

7 Applications to reinforcement learning control

In this section we discuss the application of variable partitioning algorithms given real-valued oracle and the 2-norm measure. The set $\boldsymbol{X}$ of variables to be partitioned corresponds to the set a of control variables (the action), while the oracle F corresponds to the advantage function A. While the control task is achieved by a series of actions, the advantage function describes how much one single action in the series can affect the final objective. In general, this function is complex enough so explicit representation is not available. Instead, it is usually estimated by Monte-Carlo sampling of the action series or by function approximation, where in either case it is sensible to treat the function as an oracle.

In a reinforcement learning control task, the objective is to control the action a so as to maximize the expected cumulative reward over time t. The advantage function $A(\cdot, a)$ describes the marginal gain of such an objective of an action $a = a_t$ at time t. This function can be estimated by Monte-Carlo sampling of the actions $a_t, a_{t+1}, \ldots$, or by function approximation. In either of the cases it is sensible to treat the function as an oracle when using it to partition the variables.

We compare empirically with three previous approaches. The first approach is a standard approach proposed by Williams [33, 35] and later improved by Mnih et al. [25] and Schulman et al. [32]. These approaches learn reinforcement learning control without considering the possible partitioning of the advantage function. The second approach is to trivially partition n variables into n subsets [22, 36]. This causes a large partition error which induces bias in the learning update. The third baseline partitions the variables heuristically [23]. In their method the Hessian matrix of the advantage function is first calculated using a discrete gradient method. Then this Hessian matrix is treated as an adjacency matrix of a graph, by the heuristic that two independent variables have a zero element in Hessian. Then elements are removed from Hessian, from those with the lowest absolute values, until the graph has at least k connected components. This algorithm shares some similar intuition with our first algorithm.

The rest of this section will introduce the preliminaries of how this partition may be used in reinforcement learning, and then demonstrate the comparison of scores attained in the experiments.

7.1 Reinforcement learning control and policy gradient

We consider a reinforcement learning task described by a discrete-time Markov decision process (MDP), denoted as the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, r, \rho_0, \beta)$. That includes $\mathcal{S} \in \mathbb{R}^m$ the m dimensional state space, $\mathcal{A} \in \mathbb{R}^n$ the n dimensional action space, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}^+$ the environment transition probability function, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ the reward function, ρ_0 the initial state distribution and $\beta \in [0, 1)$ the unnormalized discount factor. Here n is the number of the control variables, which is consistent with the dimension of the input oracle. A (stochastic) policy is a function $\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}_+$ that outputs a distribution over $\mathcal{A}$ on any given state $s \in \mathcal{S}$. The objective of reinforcement learning is to learn a policy π such that the expected cumulative reward $J(\theta) = \mathbb{E}_{s\sim\rho_\pi, a\sim\pi}[\sum_{t=0}^{\infty} \beta^t r(s_t, a_t)]$, is maximized, where $\rho_\pi(s) = \sum_{t=1}^{\infty} \beta^{t-1}\mathbb{P}(s_t = s)$. Since π is in a functional space, the problem is commonly relaxed to find over the space of parameterized functions the policy, such as the space of neural networks. When the policy is parameterized we denote it as π_θ.

Advantage actor-critic (A2C), a standard approach in policy optimization [25, 32], estimates the gradient of the policy $\nabla_\theta J(\theta)$. According to the policy gradient theorem [35], this gradient can be estimated by $\nabla_\theta J(\theta) = \mathbb{E}_{\pi(a|s)}[\nabla_\theta \log \pi(a|s) A^\pi(s, a)]$, where $A^\pi(s, a)$ is the advantage function of s,a, and policy π. Here $A^\pi(s, a)$ is defined as $A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s)$, where $Q^\pi(s, a) = \mathbb{E}_\pi[\sum_{t'\geq t}^{\infty} \beta^{t'-t} r(s_{t'}, a_{t'})|s = s_t, a = a_t, \pi]$ is the action-state value function and $V^\pi(s) = \mathbb{E}_{a\sim\pi(a|s)}[Q^\pi(s, a)]$ the state-value function.

It is shown later in [36] and [23], that an alternative estimator

$$\nabla_\theta J(\theta) = \sum_{j=1}^{k} \mathbb{E}_{\pi(a_{(j)}|s)}[\nabla_\theta \log \pi(a_{(j)}|s)(A^\pi(s, a_{(j)})], \tag{12}$$

may induces a lower variance. The condition that this estimator holds is that the advantage function can be approximately partitioned into k parts correspondingly:

$$A^\pi(s, a) = A^\pi_1(s, a_{(1)}) + \cdots + A^\pi_k(s, a_{(k)}) + U(s, a)$$

for some state s the estimation takes place, where $U(s, a)$ the partition error is expected to be small for the estimator to be accurate.

The learning is an iterative process that takes N updates by the gradient $\nabla_\theta J(\theta)$ while the k-partition is computed every N/N_1 iterations. Every run of the partitioning algorithm outputs the disjoint subsets $a_{(1)}, \ldots, a_{(k)}$, which is then used by (12) for N_1 iterations. It is worth note that our algorithm has a complexity of $\mathcal{O}(N_1 n^5)$, which is negligible in reinforcement learning. As the Monte-Carlo estimation of $\nabla_\theta J(\theta)$ requires a complete trial of the task (for example, play a game for an entire episode), which involves the interaction of a complex system.

7.2 Experiments

We compare our first algorithm (called pairwise estimates - PE) and our second algorithm (called submodular minimization - SM) with the aforementioned existing approaches. A2C [25] is the baseline approach in reinforcement learning who does not leverage variable partition. It uses control variates (CV) as the primary variance reduction technique. Other methods [23,36]

partition the control variable so as to reduce the variance in the Monte-Carlo estimation by Rao-Blackwellization (RB) [5]. For the discussion on variance reduction we refer the readers to the paper cited above. The comparisons are summarized below

Table 1 Comparisons of our algorithms with previous ones.

PG estimator	Variance reduction	Heuristics	Partitioning	Guarantees	Limits
A2C [25]	CV	-	-	-	-
Wu et al. [36]	CV & RB	yes	fully	no	$k = n$
Li and Wang [23]	CV & RB	yes	greedy	no	**no**
PE (our first)	CV & RB	**no**	greedy	factor-$\mathcal{O}(kn^2)$	**no**
SM (our second)	CV & RB	**no**	**optimal**	**almost opt**	**no**

Now we study the performance in terms of both the correctness and the optimality on graph cuts on weighted graphs. Correctness notes the number of times the algorithm outputs exactly the optimal partition, while optimality describes the average of the ratio of the partition error and the optimal partition error, over all the independent runs. This will illustrate the difference between greedy-based algorithms like [23] and our first algorithm, and submodular minimization based algorithms like our second algorithm. Note that submodular minimization always finds the optimal partition.

Table 2 Performance of the greedy algorithm on variable partition.

#Nodes n	$n = 5$	$n = 10$	$n = 20$	$n = 40$	$n = 100$
Submodular	-	-	-	-	-
Greedy (correctness)	7753	6271	4226	2380	1101
Greedy (optimality)	1.060	1.203	1.408	1.352	1.250

Then Table 3 compares the partitioning algorithms when the oracle is a quadratic function $a^T H_0 a$ for some random H_0. In this case our second algorithm SM also incurs an error per Theorem 2, but the error in practice is shown to be small enough. It has constantly the best empirical performance in both correctness and optimality.

Since we only replaced heuristic partitioning with our partitioning algorithm in reinforcement learning, it is reasonable that our more accurate partitions will improve reinforcement learning.

Table 3 Comparisons of the algorithms on variable partition.

#Nodes n	$n = 5$	$n = 10$	$n = 20$	$n = 40$	$n = 100$
Li and Wang [23] (correctness)	7553	5651	2929	1251	400
PE (correctness)	7709	6108	4001	2020	918
SM (correctness)	**9896**	**9630**	**9243**	**8193**	**6802**
Li and Wang [23] (optimality)	1.150	1.281	1.508	1.501	1.290
Wu et al. [36] (optimality)	9.049	13.54	20.96	34.42	72.55
PE (optimality)	1.075	1.277	1.452	1.400	1.281
SM (optimality)	**1.020**	**1.028**	**1.101**	**1.110**	**1.025**

Finally we plug our algorithms into reinforcement learning control, replacing the partitioning steps in [23]. The tasks we are testing on are standard tasks in reinforcement learning by the MuJoCo physics simulator. This includes training a simplified model of ant, cheetah, or human to run as fast as possible. The score is the cumulative reward over time, where the reward is the speed less the energy cost (which is $0.001\|a\|_2^2$). The control variables a are the forces applied to the joints. We refer to [3] for the exact simulator settings.

We have conducted experiments on all eight environments from MuJoCo that has the action dimensional higher than one, shown in Figure 1 below. In the figure the x-axis is the number of Monte-Carlo sample updates, which can be regarded as the time elapsed on the training, while the y-axis is the score attained by the model. Our second algorithm (SM) has achieved the highest score among most of these tasks, which agrees with our theoretical finding.

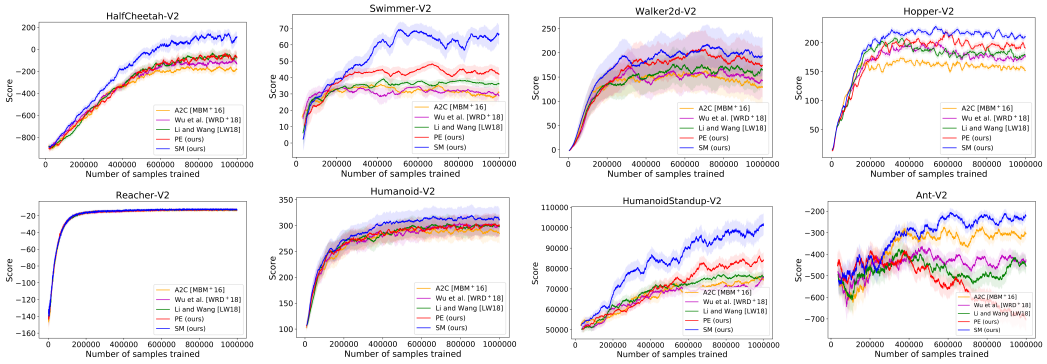


Figure 1 Empirical comparisons on MuJoCo high-dimensional control tasks. Each curve is averaged over 10 independent experiments.

References

1 S.S. Barsov and Vladimir Ulyanov. Estimates of the proximity of Gaussian measures. *Doklady Mathematics*, 34:462–, January 1987.

2 Eric Blais. Testing juntas nearly optimally. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 151–158, 2009. doi:10.1145/1536414.1536437.

3 Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint*, 2016. arXiv:1606.01540.

4 Nader H. Bshouty. Almost Optimal Distribution-Free Junta Testing. In *34th Computational Complexity Conference, CCC 2019, July 18-20, 2019, New Brunswick, NJ, USA*, pages 2:1–2:13, 2019. doi:10.4230/LIPIcs.CCC.2019.2.

5 George Casella and Christian P Robert. Rao-Blackwellisation of sampling schemes. *Biometrika*, 83(1):81–94, 1996.

6 Karthekeyan Chandrasekaran, Chao Xu, and Xilin Yu. Hypergraph K-cut in Randomized Polynomial Time. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '18, pages 1426–1438, Philadelphia, PA, USA, 2018. Society for Industrial and Applied Mathematics. URL: http://dl.acm.org/citation.cfm?id=3174304.3175399.

7 Chandra Chekuri and Shi Li. A note on the hardness of approximating the k-way hypergraph cut problem, 2015.

8 Xi Chen, Rocco A. Servedio, Li-Yang Tan, Erik Waingarten, and Jinyu Xie. Settling the Query Complexity of Non-adaptive Junta Testing. *J. ACM*, 65(6):40:1–40:18, November 2018. doi:10.1145/3213772.

9 Hana Chockler and Dan Gutfreund. A lower bound for testing juntas. *Inf. Process. Lett.*, 90(6):301–305, 2004. doi:10.1016/j.ipl.2004.01.023.

10 Roee David, Irit Dinur, Elazar Goldenberg, Guy Kindler, and Igor Shinkar. Direct Sum Testing. *SIAM Journal on Computing*, 46(4):1336–1369, 2017. doi:10.1137/16M1061655.

11 Thomas Degris, Martha White, and Richard S Sutton. Off-policy actor-critic. *arXiv preprint*, 2012. arXiv:1205.4839.

12 Irit Dinur and Konstantin Golubev. Direct Sum Testing: The General Case. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2019, September 20-22, 2019, Massachusetts Institute of Technology, Cambridge, MA, USA*, pages 40:1–40:11, 2019. doi:10.4230/LIPIcs.APPROX-RANDOM.2019.40.

13 Eldar Fischer, Guy Kindler, Dana Ron, Shmuel Safra, and Alex Samorodnitsky. Testing juntas. *Journal of Computer and System Sciences*, 68(4):753–787, 2004.

14 Martin Grötschel, László Lovász, and Alexander Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2):169–197, 1981.

15 Satoru Iwata and James Orlin. A Simple Combinatorial Algorithm for Submodular Function Minimization. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1230–1237, January 2009. doi:10.1145/1496770.1496903.

16 David Karger and Matthew S. Levine. Fast Augmenting Paths by Random Sampling from Residual Graphs. *SIAM Journal on Computing*, 44:320–339, March 2015. doi:10.1137/070705994.

17 David R. Karger. Minimum Cuts in Near-linear Time. *J. ACM*, 47(1):46–76, January 2000. doi:10.1145/331605.331608.

18 David R. Karger and Clifford Stein. A New Approach to the Minimum Cut Problem. *J. ACM*, 43(4):601–640, July 1996. doi:10.1145/234533.234534.

19 Marek Karpinski and Warren Schudy. Linear Time Approximation Schemes for the Gale-Berlekamp Game and Related Minimization Problems. In *Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing*, STOC '09, pages 313–322, New York, NY, USA, 2009. ACM. doi:10.1145/1536414.1536458.

20 Ken-ichi Kawarabayashi and Mikkel Thorup. Deterministic Global Minimum Cut of a Simple Graph in Near-Linear Time. In *Proceedings of the Forty-seventh Annual ACM Symposium on Theory of Computing*, STOC '15, pages 665–674, New York, NY, USA, 2015. ACM. doi:10.1145/2746539.2746588.

21 Regina Klimmek and Frank Wagner. A Simple Hypergraph Min Cut Algorithm, 1996. Technical Report B.

22 Ilya Kostrikov. PyTorch Implementations of Reinforcement Learning Algorithms. https://github.com/ikostrikov/pytorch-a2c-ppo-acktr, 2018.

23 Jiajin Li and Baoxiang Wang. Policy Optimization with Second-Order Advantage Information. *arXiv preprint*, 2018. arXiv:1805.03586.

24 Pasin Manurangsi. Almost-polynomial Ratio ETH-hardness of Approximating Densest k-subgraph. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2017, pages 954–961, New York, NY, USA, 2017. ACM. doi:10.1145/3055399.3055412.

25 Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy P Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, 2016.

26 Elchanan Mossel, Ryan O'Donnell, and Rocco P Servedio. Learning juntas. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 206–212. ACM, 2003.

27 Maurice Queyranne. Minimizing symmetric submodular functions. *Mathematical Programming*, 82(1-2):3–12, 1998.

28 R. M. Roth and K. Viswanathan. On the Hardness of Decoding the Gale-Berlekamp Code. In *2007 IEEE International Symposium on Information Theory*, pages 1356–1360, June 2007. doi:10.1109/ISIT.2007.4557411.

29 Aviad Rubinstein, Tselil Schramm, and S. Matthew Weinberg. Computing Exact Minimum Cuts Without Knowing the Graph. In Anna R. Karlin, editor, *9th Innovations in Theoretical Computer Science Conference (ITCS 2018)*, volume 94 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 39:1–39:16, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.ITCS.2018.39.

30 Mert Saglam. Near Log-Convexity of Measured Heat in (Discrete) Time and Consequences. In *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 967–978, 2018. doi:10.1109/FOCS.2018.00095.

31 John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint*, 2015. arXiv:1506.02438.

32 John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint*, 2017. arXiv:1707.06347.

33 Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

34 Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063, 2000.

35 Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

36 Cathy Wu, Aravind Rajeswaran, Yan Duan, Vikash Kumar, Alexandre M.Bayen, Sham Kakade, Igor Mordatch, and Pieter Abbeel. Variance Reduction for Policy Gradient with Action-Dependent Factorized Baselines. In *International Conference on Learning Representations*, 2018.

A Statistical claims

▷ Claim 27. Assume $t, \hat{t} \geq 0$. If $t^p \leq \hat{t}^p \leq t^p + (\varepsilon/2)^p$ then $t \leq \hat{t} \leq t + \varepsilon$.

Proof. The left-hand inequalities are immediate. For the right-hand ones we start we consider two cases. If $t \leq \varepsilon/2$, then $\hat{t}^p \leq 2(\varepsilon/2)^p \leq \varepsilon \leq t + \varepsilon$. If $t > \varepsilon/2$ then

$$\hat{t} - t \leq \frac{\hat{t}^p - t^p}{t^{p-1}} \leq \frac{(\varepsilon/2)^p}{(\varepsilon/2)^{p-1}} \leq \varepsilon. \qquad \triangleleft$$

▷ Claim 7. Assuming $\|F\|_{\mathbb{R},2p} \leq 1$, the value $\|F\|^p_{\mathbb{R},p}$ can be estimated within ε^p, and $\|F\|_{\mathbb{R},p}$ can be estimated within ε, from $K^p \log(1/\gamma)/\epsilon^{2p}$ queries to F in linear time with probability $1 - \gamma$ for some absolute constant K.

Proof. By Chebyshev's inequality, $\mathbb{E}[|F|^p]$ can be estimated within an additive error of $(\varepsilon/2)^p$ by averaging $(2/\varepsilon)^{2p}$ samples with probability $3/4$. The error can be improved to $1 - \gamma$ by taking the median value of $O(\log 1/\gamma)$ runs. The second bound follows from Claim 27. $\triangleleft$

B Details in the experiments

The exact reinforcement learning control algorithm we used is described below. The algorithm is based on proximal policy optimization [32] and generalized advantage estimator [11, 31] in reinforcement learning.

Algorithm 3 Policy optimization with variable partitions.

1: **Input:** Total number of samples T, batch size B, partition frequency M_p, number of value iterations M_w, initial policy parameter θ, initial value and advantage parameters w and μ;
2: **Output:** Optimized policy π_θ;
3: **for** each iteration j in $[T/B]$ **do**
4: Collect a batch of trajectory data $\{s_t^{(i)}, a_t^{(i)}, r_t^{(i)}\}_{i=1}^B$;
5: **for** M_θ iterations **do**
6: Update θ by one gradient descent step using proximal policy gradient with the gradient estimator (12);
7: **end for**
8: **for** M_w iterations **do**
9: Update w and μ by minimizing $\|V^w(s_t) - R_t\|_2^2$ and $\|\hat{A} - A^\mu(s_t, a_t)\|_2^2$ in one step;
10: **end for**
11: Estimate $\hat{A}(s_t, a_t)$ using $V^w(s_t)$ by generalized advantage estimator;
12: **if** $j \equiv 0 \pmod{M_p}$ **then**
13: Define estimation $f(\boldsymbol{X}) = \mathbb{E}[D_F(\boldsymbol{X}, \overline{\boldsymbol{X}})^2]$;
14: Run submodular minimization on $f(\boldsymbol{X})$;
15: Assign $\boldsymbol{X}$ and $\bar{\boldsymbol{X}}$ to $a_{(1)}$ and $a_{(-1)}$ in (12), respectively;
16: **end if**
17: **end for**

Variable Partitioning

The differences between our algorithm and proximal policy gradient [32] have been highlighted: Line 6 uses the estimator with partitions on the control variables. Line 12-16 find the near-optimal variable partition using submodular minimization, by Theorem 2.

We use three neural networks as function approximations: a policy network π_θ and a value network V^w as is in the baseline methods, and an advantage network A^μ solely used in the partition algorithm. The networks have the same architecture as is in the previous line of works [25, 32].

In our MuJoCo experiments, the tasks have been slightly modified (the physics simulator keeps intact). As the number of control variables of the original tasks is relatively low, we augment such dimensions by letting the agent controls two independent instances of the tasks at the same time. The scores and the reinforcement signals are then the additions of the scores of the two sub-tasks. Correspondingly, we use $k = 2$ in [23] and our algorithms.

Linear Time Subgraph Counting, Graph Degeneracy, and the Chasm at Size Six

Suman K. Bera
University of California, Santa Cruz, CA 95064, USA
sbera@ucsc.edu

Noujan Pashanasangi
University of California, Santa Cruz, CA 95064, USA
npashana@ucsc.edu

C. Seshadhri
University of California, Santa Cruz, CA 95064, USA
sesh@ucsc.edu

Abstract

We consider the problem of counting all k-vertex subgraphs in an input graph, for any constant k. This problem (denoted SUB-CNT$_k$) has been studied extensively in both theory and practice. In a classic result, Chiba and Nishizeki (SICOMP 85) gave linear time algorithms for clique and 4-cycle counting for *bounded degeneracy graphs*. This is a rich class of sparse graphs that contains, for example, all minor-free families and preferential attachment graphs. The techniques from this result have inspired a number of recent practical algorithms for SUB-CNT$_k$. Towards a better understanding of the limits of these techniques, we ask: for what values of k can SUB-CNT$_k$ be solved in linear time?

We discover a chasm at $k = 6$. Specifically, we prove that for $k < 6$, SUB-CNT$_k$ can be solved in linear time. Assuming a standard conjecture in fine-grained complexity, we prove that for all $k \geqslant 6$, SUB-CNT$_k$ cannot be solved even in near-linear time.

2012 ACM Subject Classification Mathematics of computing → Graph algorithms; Theory of computation → Graph algorithms analysis

Keywords and phrases Subgraph counting, bounded degeneracy graphs, fine-grained complexity

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.38

Funding All authors are supported by NSF TRIPODS grant CCF-1740850 NSF CCF-1813165, and ARO Award W911NF1910294.

Acknowledgements We would like to thank David Helmbold for insightful discussions. In particular, David pointed out that the lower bound for counting 8-cycles does not follow from our construction.

1 Introduction

The subgraph counting problem asks for the number of occurrences of a (typically connected) "pattern" subgraph H in a connected input graph G. It is a fundamental algorithmic problem with a rich theory [34, 51, 16, 53, 5, 24, 69, 19], and widely used in practice [32, 18, 58, 15, 60, 33, 59, 40, 68, 62, 67, 8]. With the explosion of network science, subgraph counting is now a fundamental tool used for analyzing real-world graphs. Thus, the search for fast algorithms for subgraph counting is not just a theoretical problem, but one that has many applications in bioinformatics, social sciences, and computer science.

Especially for the many of the practical applications, a common version of subgraph counting is to count the frequency of *all connected subgraphs* with k vertices [57, 54, 2, 25, 26, 31, 47, 61, 37, 71, 71]. We will denote this problem as SUB-CNT$_k$. Even in the theory literature, it is common to parametrize running time by n (vertices in G) and k, so it is natural to study SUB-CNT$_k$. There is a rich line of theoretical work on getting $n^{\mu k}$ time

algorithms, for $\mu < 1$, using matrix multiplication and tree decomposition methods [34, 53, 5, 13, 44, 70, 45, 20, 14, 19]. Unfortunately, SUB-CNT$_k$ is (a generalization of) the canonical #W[1]-hard problem, and it is not believed that there exist $f(k) \cdot n^{o(k)}$ algorithms for SUB-CNT$_k$. From an application standpoint, these algorithms are typically not practical, and do not provide algorithmic guidance. Real-world graphs are massive, and one typically desires linear-time algorithms.

An alternate perspective is to look for faster algorithms for restricted graph classes, and hope that these classes correspond to real-world graphs. A seminal result of Chiba-Nishizeki gave $O(m\kappa^{k-2})$ algorithms for k-clique counting and an $O(m\kappa)$ algorithm for 4-cycle counting, where m is the number of edges in G and κ is the *graph degeneracy* [16]. We leave the technical definitions for later; but κ can be thought of as the maximum average degree of any subgraph of G. Chiba-Nishizeki implicitly prove linear-time algorithms for SUB-CNT$_k$ for $k = 3, 4$ (explicitly shown in [57, 54]). The class of bounded (constant) degeneracy graphs is immensely rich: it contains all minor-closed families, preferential attachment graphs, and bounded expansion graphs. The graph degeneracy appears heavily in network science, and real-world graphs have typically low degeneracy (though maybe not constant).

But most importantly for subgraph counting, the techniques from Chiba-Nishizeki have inspired a number of recent practical subgraph counting algorithms [57, 54, 37, 35].

The problems of SUB-CNT$_k$ for $k \leqslant 5$ have been successfully tackled in practice using these approaches. These algorithms are often tailored for k (using, for example, specific tricks to count individual 4-vertex subgraphs) and it is not clear how far they will extend for larger k.

Towards a better theoretical understanding, we pose the following question.

For what values k, does the SUB-CNT$_k$ *problem admit a linear time algorithm in bounded degeneracy graphs?*

1.1 Our Results

The question above has a surprisingly clean resolution, assuming conjectures from fine-grained complexity. For simplicity, we assume that the input graph G is connected. We assume Las Vegas randomized algorithms, so we talk of expected running times.

Our main theorem asserts linear time algorithms for counting (up to) 5-vertex subgraphs in bounded degeneracy graphs. For counting 6-vertex subgraphs and beyond, it is unlikely that even near-linear time algorithms exists.

▶ **Theorem 1** (The chasm at size 6)**.** *For $k \leqslant 5$, there is an expected $O(m\kappa^{k-2})$ time algorithm for* SUB-CNT$_k$.

Assume the Triangle Detection Conjecture *(Conj. 2). There exists an absolute constant $\gamma > 0$ such that the following holds. For any $k \geqslant 6$ and any function $f : \mathbb{N} \to \mathbb{N}$, there is no (expected) $o(m^{1+\gamma} f(\kappa))$ algorithm for* SUB-CNT$_k$.

The Triangle Detection Conjecture was first stated by Abboud and Williams [1]. They proved many lower bounds for the dynamic version of many well known graph problems such as bipartite perfect matching, single source reachability etc. It is actually believed that the constant γ could be as large as 1/3.

▶ **Conjecture 2** (Triangle Detection Conjecture [1])**.** There exists a constant $\gamma > 0$ such that in the word RAM model of $O(\log n)$ bits, any algorithm to detect whether an input graph on m edges has a triangle requires $\Omega(m^{1+\gamma})$ time in expectation.

1.2 Main Ideas

Conditional Lower Bounds

It is instructive to look at the conditional lower bounds. The reduction of triangle detection to subgraph counting in bounded degeneracy graphs is actually quite simple. Suppose we want to detect (or even count) triangles in an input graph G. Get graph G' by subdividing each edge into two, so a triangle in G becomes a $\mathcal{C}_6$ (6-cycle) in G'. But the degeneracy of G' is just 2! (In any induced subgraph of G', the minimum degree is at most 2, proving the bound.) Thus, if there exists $o(f(\kappa)m^{1+\gamma})$ time algorithms for counting 6-cycles, that would violate the Triangle Detection Conjecture.

It is fairly straightforward to generalize this idea for larger cycles, by replacing edges in G by short paths. Assuming Triangle Detection Conjecture, for all $k \geqslant 6$ and $k \neq 8$, we can rule out linear time algorithms for counting $\mathcal{C}_k$ in bounded degeneracy graphs. Our reduction does not work for $\mathcal{C}_8$; instead we consider a different subgraph for the case of $k = 8$ ($\mathcal{C}_7$ with a tail). We give the details in Section 5.

This reduction fails for counting 5-cycles and in general, it does not work for counting any 5-vertex subgraph. For good reason, as we discovered an efficient algorithm for this problem. This is the more technical part of our paper.

Algorithmic Framework

We present an algorithmic framework for solving the SUB-CNT$_k$ problem, that generalizes the core idea of Chiba and Nishizeki [16]. It is known from past work that their ideas basically provide an $O(m\kappa^{k-2})$ algorithm for SUB-CNT$_k$, for $k = 3, 4$. The main challenge is to get such an algorithm for $k = 5$, thereby nailing down the chasm of Theorem 1. This leads to new results for counting various 5-vertex subgraphs. Perhaps more than these individual results, our main contribution lies in identifying structural decompositions of the pattern subgraphs that allows for efficient algorithms. This decomposition also sheds light on why certain k-vertex subgraphs, for $k \geqslant 6$, does not seem to have any efficient algorithms in bounded arboricity graphs. We give an outline of our framework next, and present it formally in Section 4.

The key idea that comes from Chiba-Nishizeki is to perform subgraph counting on $G^{\rightarrow}$, an acyclic orientation of G where the out degree of each vertex is bounded by $O(\kappa)$[1]. The classic clique and 4-cycle counting algorithms enumerate directed stars and directed paths of length 2 to count subgraphs. We note that the algorithm does *not* enumerate 4-cycles, since there can be $\Omega(n^2)$ 4-cycles. It requires clever indexing to solve this problem, which we generalize in our algorithm.

The crucial generalization of this idea is to enumerate directed rooted trees. Specifically, we count occurrences of a connected pattern H by counting occurrences of all possible acyclic orientations (up to isomorphism) $H^{\rightarrow}$ of H in $G^{\rightarrow}$. The main idea is to find the largest directed rooted tree in $H^{\rightarrow}$, with edges directed away from the root. Call this tree T. Since outdegrees in $G^{\rightarrow}$ are bounded, we can efficiently enumerate all copies of T. Any copy of $H^{\rightarrow}$ in $G^{\rightarrow}$ is formed by extending a copy of T, but $H^{\rightarrow}$ may contain vertices that are not in T. Thus, the extensions could be expensive to compute. But when H has at most 5 vertices, we can prove that $H^{\rightarrow} \setminus T$ is itself either a collection of rooted stars or paths. We can create hash tables that store information about the occurrences of the latter. The final count of $H^{\rightarrow}$ is obtained by enumerating T and carefully combining counts from the hash tables.

[1] Technically, this is *not* the idea of Chiba-Nishizeki, who use the degree orientation. But it was somewhat of a folklore result that it is easy to get the same result using the degeneracy orientation. Arguably the first such reference is Schank-Wagener [63].

2 Related Work

Subgraph counting problems has a long and rich history. More than three decades ago, Itai and Rodeh [34] gave the first non-trivial algorithm for the triangle detection and counting problems with $O(m^{3/2})$ runtime. Subsequently, Chiba and Nishizeki [16] gave an elegant algorithm based on the degree based vertex ordering that solves triangle counting, 4-cycle counting and ℓ-clique counting with running times of $O(m\kappa)$, $O(m\kappa)$, and $O(m\kappa^{\ell-2})$ respectively (κ denotes the degeneracy). In comparison, our algorithm exploits the *degeneracy ordering* of the vertices (see Section 3 for a formal definition); this enables us to create a uniform framework for any k-vertex subgraph for $k \in \{4, 5\}$. In dense graphs, the best bounds for the clique counting problem are achieved by fast matrix multiplications based algorithms [53, 24]; Vassilevska [69] gave combinatorial algorithm with significantly reduced space requirement. For general subgraphs, there is a rich line of research based on matrix multiplication, tree decomposition and vertex cover methods [34, 53, 5, 13, 44, 70, 45, 20, 14, 19] – these works focus on getting $n^{\mu k}$ time algorithmis, for $\mu < 1$.

Subgraph counting problems, specifically triangle counting, clique counting and cycle counting problems, has also been studied extensively in various Big Data models such as property testing model [22, 23, 6], MapReduce settings [17, 66, 42], and streaming model [7, 38, 46, 39, 3, 36, 56, 49, 9]. Most of these work focuses on an approximate count, rather than an exact count. In the applied world, there are many efficient algorithms that are based on clever sampling techniques [11, 10, 33, 75, 61, 73, 72, 37, 71]. Exact counting has also been studied extensively in the applied world [2, 57, 54, 12, 28, 50, 65, 47, 29, 31, 30, 25, 26]. In particular, Ahmed et al. [2] presented an algorithmic framework for solving the SUB-CNT$_4$ problem, called PGD (Parametrized Graphlet Decomposition), which scales to graphs with tens of millions of edges. Pinar et al. [57] studied the SUB-CNT$_5$ problem, and gave the current state of the art ESCAPE library based on degree ordering techniques. However, the provable runtime of their algorithm for certain 5-vertex subgraphs is quadratic, $O(n^2)$. For a deeper exploration of related applied work, refer to the tutorial on subgraph counting by Seshadhri and Tirthapura [64].

The subgraph detection problem, which asks whether an input graph has a copy of the subgraph, is a well-studied problem [34, 51, 4, 5, 41, 45, 74]. For the triangle detection problem, the best known algorithm is based on fast matrix multiplication and it runs in time $O(\min\{n^\omega, m^{2\omega/(\omega+1)}\})$ [5]. If $\omega = 2$, this would give us $O(\min\{n^2, m^{4/3}\})$ algorithm for the triangle detection problem. Hence, to falsify the Triangle Detection Conjecture, it would require a major breakthrough result in the algorithmic graph theory world. For a more detailed discussion on the Triangle Detection Conjecture and its implications, refer to the paper by Abboud and Williams [1].

In the subgraph enumeration problem, the goal is to output each occurrences of the target subgraph. Chiba and Nishizeki [16] showed that it is possible to enumerate all the triangles in a graph along with counting the total number of triangles in $O(m\kappa)$ time. For enumerating all the triangles, $O(m\kappa)$ time is effectively optimal assuming the 3SUM Conjecture [55, 43]. Eppstein [27] studied the bipartite subgraph enumaration problem in bounded arboricity graphs.

3 Preliminaries

In this paper, we study the SUB-CNT$_k$ problem which asks for the number of occurrences of each k-vertex subgraph H, in an input graph G with n vertices and m edges. We consider k to be a constant. For a fixed subgraph H, we use SUB-CNT$_H$ to denote the problem of

counting all occurrences of H in the input graph G. When H is the triangle subgraph, we denote the corresponding counting problem as TRI-CNT. In the context of the SUB-CNT$_k$ problem, we always use G to denote the input graph and H to denote the subgraph to be counted. Both G and H are simple, connected, undirected and unweighted.

In our algorithmic framework, directed graphs play a crucial role. We use $N^+_G(u)$ and $N^-_G(u)$ to denote the out-neighborhood and in-neighborhood of a vertex u in a directed graph G, respectively. We define $d^+_G(u) = |N^+_G(u)|$ and $d^-_G(u) = |N^-_G(u)|$. If the graph is clear from the context, we drop the subscript G.

A graph G is k-degenerate if every subgraph of G has a vertex of degree at most k. The *degeneracy* of a graph G (also called coloring number, refer to Sec. 5.2 of [21]), denoted as $\kappa(G)$, is the smallest integer k such that G is k-degenerate. The *arboricity* of a graph G, denoted as $\alpha(G)$, is the smallest integer k such that the edge set $E(G)$ can be partitioned into k forests. When the graph G is clear from the context, we simply write κ, and α, instead of $\kappa(G)$ and $\alpha(G)$. A classic theorem of Nash-Williams shows that the degeneracy and arboricity are closely related. All our results can be stated in terms of either of the parameters.

▶ **Theorem 3** (Nash-Williams [52]). *In every graph G, $\alpha(G) \leqslant \kappa(G) \leqslant 2\alpha(G) - 1$.*

Vertex ordering is central to many subgraph counting algorithms. In this paper, we work with the *degeneracy ordering* of G, which is defined as follows.

▶ **Definition 4.** Degeneracy ordering *of a graph G, denoted by $\lhd$, is obtained by repeatedly removing the vertex with minimum degree. The ordering is defined by the removal time.*

For example, if $u \lhd v$, then u is removed before v according to the above process. *Degeneracy ordering* can be found in linear time [48].

Using any vertex ordering $\prec$ of an undirected graph G, we construct a directed graph $G^{\rightarrow}_{\prec}$ as follows: for each edge $\{u, v\} \in E(G)$, direct the edge from u to v iff $u \prec v$. We denote this directed edge as (u, v). Observe that $G^{\rightarrow}_{\prec}$ is necessarily acyclic. We denote the directed graph obtained from *degeneracy ordering* $\lhd$ as $G^{\rightarrow}_{\lhd}$. The following two are folklore results about vertex ordering and degeneracy, and can be derived from Prop. 5.2.2 of [21].

▶ **Lemma 5.** *For each vertex $v \in G^{\rightarrow}_{\lhd}$, $d^+(v) \leqslant \kappa$.*

▶ **Lemma 6.** *If there exists a vertex ordering $\prec$ of G such that in the corresponding directed graph $G^{\rightarrow}_{\prec}$, $d^+(v) \leqslant k$ for each vertex v, then $\kappa(G) \leqslant k$.*

Next, we formally define a match (occurrence) of the target subgraph H in the input graph G. We also define a match in the context of directed graphs H' and G'.

▶ **Definition 7.** *A match of H in G is a bijection $\pi : S \to V(H)$ where $S \subseteq V(G)$ and for any two vertices u and v in S, $\{u, v\} \in E(G)$ if $\{\pi(u), \pi(v)\} \in E(H)$.*

▶ **Definition 8.** *A match of H' in G' is a bijection $\pi : S \to V(H')$ where $S \subseteq V(G')$ and for any ordered pair of vertices (u, v) where u and v are in S, $(u, v) \in E(G')$ if $(\pi(u), \pi(v)) \in E(H')$.*

Our algorithm counts matches of H in G by counting matches of all possible acyclic orientations $H^{\rightarrow}$ of H in $G^{\rightarrow}_{\lhd}$. In general, whenever we use '$\rightarrow$' to denote a directed graph, such as in $G^{\rightarrow}_{\lhd}$ and $H^{\rightarrow}$, the directed graph is a DAG.

We denote the number of matches of H in G by $\mathrm{M}(G, H)$. An incomplete match of H in G is an injection $\pi : S \to V(H)$ (so $|S| < |V(H)|$), that has the same properties of a match except being surjective. Consider two incomplete matches (injections) of H, $\pi_1 : S_1 \to V(H)$, and $\pi_2 : S_2 \to V(H)$. Let $V_{\pi_1} = \{\pi_1(u) \mid u \in S_1\}$ and $V_{\pi_2} = \{\pi_2(u) \mid u \in S_2\}$. We say that π_2 completes π_1 to be a match of H, when $V(H) = V_{\pi_1} \cup V_{\pi_2}$ (surjective), $V_{\pi_1} \cap V_{\pi_2} = \emptyset$ (injective), and for any two vertices $u \in S_1$ and $v \in S_2$, $\{u, v\} \in E(G)$ if $\{\pi_1(u), \pi_2(v)\} \in E(H)$. In case of directed graphs, it should hold that $(u, v) \in E(G')$ if $(\pi_1(u), \pi_2(v)) \in E(H')$ and $(v, u) \in E(G')$ if $(\pi_2(v), \pi_1(u)) \in E(H')$.

Two matches are distinct if they are not authomorphims of a match. In other words, two matches π_1 and π_2 of H are equivalent, if they map two automorphisms of the exact same subgraph of G to H. We denote the number of distinct matches of H in G by $\mathrm{DM}(G, H)$. In the SUB-CNT$_k$ problem, we are interested in $\mathrm{DM}(G, H)$ for all k-vertex subgraphs H.

4 Subgraph Counting Through Orientation and Directed Trees

In this section, we discuss our algorithmic framework for solving the SUB-CNT$_k$ problem. Instead of directly counting the number of occurrences of a k-vertex subgraph H in the input graph G, we count the occurrences of all possible DAG $H^{\rightarrow}$ (up to isomorphism) of H in the graph $G^{\rightarrow}_{\triangleleft}$. To achieve this, our main idea is to find the largest directed tree of $H^{\rightarrow}$, enumerate all matches of this tree, and then count matches of the remaining vertices using structures we save in a hash table. In Section 4.1, we show that our framework solves the SUB-CNT$_5$ problem in expected $O(m\kappa^3)$ time. In Section 4.2, we demonstrate the limitation of our framework as it fails to solve the SUB-CNT$_{\mathcal{C}_6}$ problem *efficiently*.

Algorithm 1 Counting distinct matches of all 5-vertex subgraphs in G (SUB-CNT$_5$).

procedure COUNT-ALL-5(G)
  Derive $G^{\rightarrow}_{\triangleleft}$ by orienting $E(G)$ with respect to degeneracy ordering.
  for all connected 5-vertex subgraphs H except 4-star **do**
    Run COUNT-MATCH($G^{\rightarrow}_{\triangleleft}, H$) and save the result for H.
  Save $\sum_{u \in V(G)} \binom{d(u)}{4}$ for 4-star.

Algorithm 2 Counting distinct matches of H in G (SUB-CNT$_H$).

procedure COUNT-MATCH($G^{\rightarrow}_{\triangleleft}, H$)
  $\mathrm{DM}(G, H) \leftarrow 0$
  for all possible DAGs (up to isomorphism) $H^{\rightarrow}$ of H **do**
    $\mathrm{M}(G^{\rightarrow}_{\triangleleft}, H^{\rightarrow}) \leftarrow 0$
    Find one of the largest DRTSs in $H^{\rightarrow}$, and call it $T_{\max}$.
    for all match π of $T_{\max}$ in $G^{\rightarrow}_{\triangleleft}$ **do**
      if π is a match of $H^{\rightarrow}$ **then** ▷ $V(T_{\max}) = V(H^{\rightarrow})$. Lemma 14
        $\mathrm{M}(G^{\rightarrow}_{\triangleleft}, H^{\rightarrow}) \leftarrow \mathrm{M}(G^{\rightarrow}_{\triangleleft}, H^{\rightarrow}) + 1$
      else if π is an incomplete match of $H^{\rightarrow}$ **then** ▷ Lemma 14
        $k \leftarrow$ number of ways to complete π to a match of $H^{\rightarrow}$. ▷ Lemma 16
        $\mathrm{M}(G^{\rightarrow}_{\triangleleft}, H^{\rightarrow}) \leftarrow \mathrm{M}(G^{\rightarrow}_{\triangleleft}, H^{\rightarrow}) + k$
    $\mathrm{DM}(G, H) \leftarrow \mathrm{M}(G^{\rightarrow}_{\triangleleft}, H^{\rightarrow}) / |Aut(H^{\rightarrow})|$
  return $\mathrm{DM}(G, H)$

4.1 5-vertex Subgraph Counting

Our main algorithmic result is given in the following theorem.

▶ **Theorem 9.** *There is an algorithm that solves the* SUB-CNT$_5$ *problem in* $O(m\kappa^3)$ *time.*

Our strategy is to count matches of all possible DAGs (up to isomorphism) $H^{\rightarrow}$ of H in $G^{\rightarrow}_{\triangleleft}$, to obtain the number of distinct matches of H in G. Alg. 2 demonstrates this subroutine of our algorithm for SUB-CNT$_5$, which is shown in Alg. 1. First, we find one of the largest directed rooted tree subgraphs (*DRTS*), which we define as follows, in $H^{\rightarrow}$.

▶ **Definition 10.** *Given any directed graph D, a directed rooted tree subgraph (DRTS) of D, is a subgraph T of D, where the underlying undirected graph of T is a rooted tree, and edges are oriented away from the root in T.*

The following lemma shows that we can find all matches of any DRTS in $H^{\rightarrow}$ in the desired time.

▶ **Lemma 11.** *Let T be a directed tree with k vertices. All matches of T in $G^{\rightarrow}_{\triangleleft}$ can be enumerated in $O(m\kappa^{k-2})$.*

Proof. Let $t_1, \ldots, t_k$ be a BFS ordering of T starting at the root t_1. Fix an edge $(u, v) \in E(G^{\rightarrow}_{\triangleleft})$ and map u to t_1 and v to t_2. There are m possible matches for (t_1, t_2), which we can find by enumerating the edges of $G^{\rightarrow}_{\triangleleft}$. Now, we will choose vertices to map to $t_3, \ldots, t_k$, one by one, in this order. Since the out-degree of each vertex in $G^{\rightarrow}_{\triangleleft}$ is at most κ, if we have already mapped vertices to $t_1, \ldots, t_i$, there are at most κ vertices that could be mapped to t_{i+1}. Therefor $\mathrm{M}(G^{\rightarrow}_{\triangleleft}, T) = O(m\kappa^{k-2})$, and we can enumerate all of them by first choosing (u, v) to map to (t_1, t_2) and then choosing vertices to map to $t_3, \ldots, t_k$, in this order and one by one. ◀

▶ **Observation 12.** *Call a vertex v of a directed graph a* source *vertex, if $d^-(v) = 0$. Consider T to be one of the largest DRTSs of a DAG D. T has to have a source vertex of D as the root, otherwise the root has an in-neighbor v, which is not in T as it would create a cycle. Adding v to T creates a new DRTS which has one more vertex than T. This contradicts the fact that T is one of the largest DRTSs of D. Hence, the root of T has to be a source vertex of D.*

Given a 5-vertex DAG $H^{\rightarrow}$, we can find a DRTS that has the most number of vertices among all DRTSs of $H^{\rightarrow}$ in constant time. First, find all source vertices, and then apply a Breath First Search (BFS) starting from each of these vertices and pick a BFS tree with the most number of vertices among all. The following lemma shows that the largest DRTS has at least 3 vertices for a 5-vertex connected subgraphs, except 4-star. Notice that, the largest DRTS of a 4-star with all the edges oriented towards the center has two vertices.

▶ **Lemma 13.** *Let H be a connected undirected 5-vertex graph that is not a 4-star. Each largest DRTS of any DAG $H^{\rightarrow}$, which is an acyclic orientation of H, has at least three vertices.*

Proof. We prove this lemma by contradiction. Assume that any DRTS of $H^{\rightarrow}$ has at most two vertices. A directed 2-path, or any vertex with at least two outgoing edges result in a DRTS with three vertices. Therefore,

(a) $H^{\rightarrow}$ does not have a 2-path,

(b) each vertex in $H^{\rightarrow}$ has at most one outgoing edges.

Notice that, since $H^{\rightarrow}$ is a DAG, it has at least one source vertex. Consider a source vertex u. Since H is connected, u has at least one neighbor, and by (b) it should have exactly one neighbor. Let $N^+(u) = \{v\}$, then $N^+(v) = \emptyset$, by (a). So, v should have at least one incoming neighbor w. By (a), w has no incoming edges, and it has no outgoing edges by (b). Call the other two vertices x and y. As H is connected, there should be a connection between $\{u, v, w\}$ and $\{x, y\}$. u and w cannot have any neighbor other than v, so x and y could only be connected to v. Since H is not a star, there should be an edge between x and y. Without loss of generality, let (x, y) be that edge. By (a), $(y, v) \notin E(H^{\rightarrow})$ and by (b) $(x, v) \notin E(H^{\rightarrow})$. So, $\{u, v, w\}$ is not connected to $\{x, y\}$, and H is disconnected, which is a contradiction. Thus, the assumption that any DRTS of $H^{\rightarrow}$ has at most two vertices is wrong, and each largest DRTS of $H^{\rightarrow}$ has at least three vertices. ◀

So far, we know that we can find one of the largest DRTSs of H, which has at least 3 vertices. We use $T_{\max}$ to denote this DRTS. By Lemma 11, we can enumerate all matches of $T_{\max}$ in $G^{\rightarrow}_{\triangleleft}$ in $O(m\kappa^3)$ time. For each such match, we need to validate whether it is a (incomplete) match of $H^{\rightarrow}$ or not. If it is not, then it could not be completed to a match of $H^{\rightarrow}$. The following lemma shows that we can perform this validation efficiently. In the remaining part of this section, "constanct expected time", refers to constant amortized time access to hash maps that we use.

▶ **Lemma 14.** *Let T be a DRTS of a DAG $H^{\rightarrow}$ of a connected k-vertex graph H. Assume edges of $G^{\rightarrow}_{\triangleleft}$ are saved in a hash table. For each match π of T in $G^{\rightarrow}_{\triangleleft}$, it takes $O(|E(H^{\rightarrow})|)$ expected time to validate whether π is a (incomplete) match of $H^{\rightarrow}$ or not.*

Proof. Since π is a bijection, it has an inverse which we denote by π^{-1}. Let $H^{\rightarrow}[V(T)]$ denote the subgraph of $H^{\rightarrow}$ induced on $V(T)$. Observe that, there could be edges in $H^{\rightarrow}[V(T)]$ not present in T. For π to be a match (if $V(T) = V(H^{\rightarrow})$) or incomplete match of $H^{\rightarrow}$, these edges have to be present between corresponding vertices in $G^{\rightarrow}_{\triangleleft}$ mapped to T by π. Formally, consider all ordered pairs of vertices $(a, b) \in V(T) \times V(T)$ such that $(a, b) \in E(H^{\rightarrow})$ and $(a, b) \notin E(T)$, π is a match or incomplete match of $H^{\rightarrow}$ iff $(\pi^{-1}(a), \pi^{-1}(b)) \in E(G^{\rightarrow}_{\triangleleft})$ for all such pairs of vertices. To validate this, we enumerate all edges (a, b) of $H^{\rightarrow}[V(T)]$ which are not present in T, and search for $(\pi^{-1}(a), \pi^{-1}(b))$ in hashed edges of $G^{\rightarrow}_{\triangleleft}$ in expected constant time. So this only requires $O(|E(H^{\rightarrow})|)$ expected time. ◀

If $V(T_{\max}) = V(H^{\rightarrow})$, then a match of $T_{\max}$ could be a match of $H^{\rightarrow}$ too, which could be verified as explained. If there is a vertex in $H^{\rightarrow}$ which is not present in $T_{\max}$, then after validating that a match of $T_{\max}$ is an incomplete match of $H^{\rightarrow}$, we need to find the number of ways to complete it to a match of $H^{\rightarrow}$. For this we need to count matches of each possible structures that $T_{\max}$ does not cover in $H^{\rightarrow}$. We save the count of these structures in $G^{\rightarrow}_{\triangleleft}$, in hash tables. The following lemma shows that this can be done efficiently.

▶ **Lemma 15.** *In $O(m\kappa^3)$ time and space, we can save all the following key and value pairs in hash maps $\mathcal{HM}_1$, $\mathcal{HM}_2$, and $\mathcal{HM}_3$.*

1. $\mathcal{HM}_1 : ((u, v), 1)$ *where* $(u, v) \in E(G^{\rightarrow}_{\triangleleft})$
2. $\mathcal{HM}_2 : (S, k)$ $\forall S \subseteq V(G^{\rightarrow}_{\triangleleft})$ *where* $1 \leqslant |S| \leqslant 4$*, and k is the number of vertices u such that* $S \subseteq N^+(u)$
3. $\mathcal{HM}_3 : ((S_1, S_2), \ell)$ $\forall S_1, S_2 \subseteq V(G^{\rightarrow}_{\triangleleft})$*, where* $1 \leqslant |S_1 \cup S_2| \leqslant 3$*, and ℓ is the number of edges $e = (u, v) \in E(G^{\rightarrow}_{\triangleleft})$ such that $S_1 \subseteq N^+(u)$ and $S_2 \subseteq N^+(v)$.*

Proof. We show how to enumerate and save all these structures in $\mathcal{HM}_1$, $\mathcal{HM}_2$, and $\mathcal{HM}_3$.

1. $\mathcal{HM}_1$: We can easily do this in $O(m)$ by enumerating the out-neighbors of each vertex
2. $\mathcal{HM}_2$: For each edge $e = (u, v)$, we can enumerate all subsets T of the set $\{w \in N^+(u) \mid v \lhd w\}$, where $|T| \leqslant 3$, in $O(\kappa^3)$ time, and increment the value for the key $T \cup \{v\}$ in the hash map by one.
3. $\mathcal{HM}_3$: For each edge $e = (u, v)$ $(v \in N^+(u))$, we enumerate all possible subset $S_1 \subseteq N^+(u) \setminus \{v\}$ where $|S_1| \leqslant 3$. And, for each S_1 we enumerate all possible $S_2 \setminus S_1$ in subsets of $N^+(v)$, such that $1 \leqslant |S_1 \cup S_2| \leqslant 3$. This takes $O(\kappa^3)$ as the out-degree of each vertex is at most κ, and we choose up to three vertices. All possible $S_1 \cap S_2$ can be determined by checking the connection between v and each vertex in S_1 using the hashed edges of $G^{\rightarrow}$ in $\mathcal{HM}_1$. ◀

The following lemma shows that we can count the number of ways to complete a match of $T_{\max}$, which is also an incomplete match of $H^{\rightarrow}$, to a match of $H^{\rightarrow}$ efficiently.

▶ **Lemma 16.** *Let H be a 5-vertex connected graph, $H^{\rightarrow}$ be a DAG of H, and $T_{\max}$ be one of the largest DRTSs in $H^{\rightarrow}$. Assume $\mathcal{HM}_1$, $\mathcal{HM}_2$, and $\mathcal{HM}_3$ are given. For each match π of $T_{\max}$ in $G^{\rightarrow}_{\lhd}$ which is an incomplete match of $H^{\rightarrow}$, we can count the number of ways to complete π to a match of $H^{\rightarrow}$ in expected constant time.*

Proof. By Lemma 13, $T_{\max}$ has at least 3 vertices, and since π is an incomplete match (not a match) of $H^{\rightarrow}$, we can assume that $|V(T_{\max})| < 5$. Observe that, $T_{\max}$ is a maximal DRTS. Any vertex in $H^{\rightarrow}$ which is not in $T_{\max}$ can only be connected to vertices of $T_{\max}$ by outgoing edges, otherwise they could be added to $T_{\max}$ to create a larger DRTS of $H^{\rightarrow}$, which contradicts the maximality of $T_{\max}$. We consider two cases where $T_{\max}$ has three or four vertices.

Let $|V(T_{\max})| = 4$, and i be the only vertex in $H^{\rightarrow}$ that is not in $T_{\max}$. To complete π to a match of $H^{\rightarrow}$, we need to choose a vertex in $G^{\rightarrow}_{\lhd}$, that is connected by outgoing edges to vertices mapped to the out-neighborhood of i in $H^{\rightarrow}$. Let $S_i = \{\pi^{-1}(t) \mid t \in N^+_{H^{\rightarrow}}(i)\}$. $\mathcal{HM}_2(S_i)$ is the number of vertices that could be mapped to i, but some of them may be already mapped to a vertex in $T_{\max}$, by π. Let r_i denote the number of vertices $v \in \{\pi^{-1}(t) \mid t \in V(T_{\max})\}$, where $S_i \subseteq N^+_{G^{\rightarrow}_{\lhd}}(v)$. We can obtain r_i in expected constant time, by enumerating vertices mapped to $V(T_{\max})$, and counting vertices that are connected to all vertices in S_i. For any vertex, we can check the connection to each vertex of S_i using $\mathcal{HM}_1$ in expected constant time. The number of ways to complete π to a match of $H^{\rightarrow}$ in this case is $\mathcal{HM}_2(S_i) - r_i$.

Now we consider the case where $|V(T_{\max})| = 3$. Let $V(H^{\rightarrow}) \setminus V(T_{\max}) = \{i, j\}$. To complete π to a match of $H^{\rightarrow}$, we only need to choose two vertices of $G^{\rightarrow}_{\lhd}$ to map to i and j. Let $S_i = \{\pi^{-1}(t) \mid t \in V(T_{\max}) \cap N^+_{H^{\rightarrow}}(i)\}$ and $S_j = \{\pi^{-1}(t) \mid t \in V(T_{\max}) \cap N^+_{H^{\rightarrow}}(j)\}$. We consider two cases, where i and j are connected or not. If they are connected, without loss of generality, assume $(i, j) \in E(H^{\rightarrow})$. If $(i, j) \in E(H^{\rightarrow})$, then we can use $\mathcal{HM}_3$ in Lemma 15, to find the number of edges (u, v) where u and v could be mapped to i and j, respectively. Let $r_{(i,j)}$ be the number of edges $e = (w, x) \in E(G^{\rightarrow}_{\lhd})$, where w and x are mapped to vertices in $T_{\max}$ by π, such that, $S_i \subseteq N^+_{G^{\rightarrow}_{\lhd}}(w)$, and $S_j \subseteq N^+_{G^{\rightarrow}_{\lhd}}(x)$. We can obtain $r_{(i,j)}$ in expected constant time using $\mathcal{HM}_1$. Then the number of edges (u, v) that could be mapped to (i, j) is $\mathcal{HM}_3((S_i, S_j)) - r_{(i,j)}$. Next case is when $(i, j) \notin E(H^{\rightarrow})$. In this case, we use $\mathcal{HM}_2$ to find the number of pair of vertices of $G^{\rightarrow}_{\lhd}$ which could be mapped to i and j. Let r_i (r_j resp.) denote the number of vertices $v \in V(G^{\rightarrow}_{\lhd})$ where v is mapped to a vertex in $T_{\max}$ and $S_i \subseteq N^+_{G^{\rightarrow}_{\lhd}}(v)$ ($S_j \subseteq N^+_{G^{\rightarrow}_{\lhd}}(v)$ resp.). Also, we use $r_{i,j}$ to denote the number of vertices $v \in V(G^{\rightarrow}_{\lhd})$ that are counted in both r_i and r_j, meaning

$S_i \cup S_j \subseteq N^+_{G^{\rightarrow}_{\lhd}}(v)$. We can obtain r_i, r_j, and $r_{i,j}$ easily in expected constant time using $\mathcal{HM}_1$. The number of pairs of vertices which could be mapped to i and j is equal to $(\mathcal{HM}_2(S_i) - r_i) \cdot (\mathcal{HM}_2(S_j) - r_j) - (\mathcal{HM}_2(S_i \cup S_j) - r_{i,j})$. ◀

Now, we have all the tools to efficiently count distinct matches of a DAG of $H^{\rightarrow}$ in $G^{\rightarrow}_{\lhd}$. The following lemma shows that we can do this in $O(m\kappa^3)$ expected time.

▶ **Lemma 17.** *There is an algorithm which counts distinct matches for each possible DAG (up to isomorphism) $H^{\rightarrow}$ of a 5-vertex connected subgraphs H, in $O(m\kappa^3)$ expected time.*

Proof. Fix a DAG $H^{\rightarrow}$ of H. If H is a 4-star and $H^{\rightarrow}$ has ℓ incoming neighbors, then the number of distinct matches of $H^{\rightarrow}$ is $\sum_{u \in V(G^{\rightarrow}_{\lhd})} \binom{d^-(u)}{\ell}\binom{d^+(u)}{4-\ell}$. Assume that H is not a 4-star. Find a DRTS of $H^{\rightarrow}$ with the most number of vertices among all its DRTSs, and call it $T_{\max}$. This can be done in constant time for $H^{\rightarrow}$. By Lemma 13, $T_{\max}$ has at least three vertices. We will now enumerate all matches of $T_{\max}$ in $G^{\rightarrow}_{\lhd}$. By Lemma 11, this step requires $O(m\kappa^3)$ expected time. For each match π of $T_{\max}$ in $G^{\rightarrow}_{\lhd}$, we can verify whether π is a match (if ($|V(T_{\max})| = 5$) or incomplete match of $H^{\rightarrow}$ in expected constant time, by Lemma 14. If $|V(T_{\max})| = 5$, while enumerating all matches of $T_{\max}$, we only count them if they are a match of $H^{\rightarrow}$. So in this case we can count $\mathrm{M}(G^{\rightarrow}_{\lhd}, H^{\rightarrow})$ in $O(m\kappa^3)$ expected time.

Otherwise, $T_{\max}$ has 3 or 4 vertices. In this case, for each match π of $T_{\max}$, we first verify that it is also an incomplete match of $H^{\rightarrow}$. Then, we count the number of ways to complete π to a match of $H^{\rightarrow}$, which we can do in expected constant time, by Lemma 16. To obtain $\mathrm{M}(G^{\rightarrow}_{\lhd}, H^{\rightarrow})$, we simply sum the ways to complete each incomplete match we have found, to a match of $H^{\rightarrow}$.

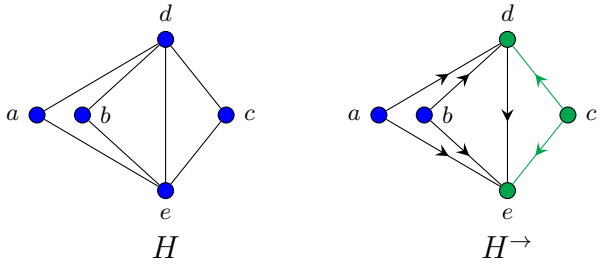
This approach gives us the number of all (not necessarily distinct) matches of $H^{\rightarrow}$ in $G^{\rightarrow}_{\lhd}$. Let $H^{\rightarrow}_{\pi}$ be a subgraph of $G^{\rightarrow}_{\lhd}$ that π maps to $H^{\rightarrow}$. Each automorphism of $H^{\rightarrow}$, gives a new match π' which is not distinct from π, as it is still mapping H_π (the same copy of H) to $H^{\rightarrow}$ (example in Fig. 1b). As each match of $H^{\rightarrow}$, also maps vertices to $T_{\max}$, resulting in a match of $T_{\max}$ and an (incomplete) match of $H^{\rightarrow}$, we will find all distinct matches of $H^{\rightarrow}$ and count each one exactly $|Aut(H^{\rightarrow})|$ times. We want the number of distinct matches, which we can obtain by dividing the count of all matches by $|Aut(H^{\rightarrow})|$.

Thus, it requires $O(m\kappa^3)$ expected time to create $\mathcal{HM}_1$, $\mathcal{HM}_2$, and $\mathcal{HM}_3$ by Lemma 15, $O(m\kappa^3)$ time for enumerating matches of $T_{\max}$, expected constant time to validate these matches, and expected constant time for counting ways to complete each such match, that is verified to be an incomplete match of $H^{\rightarrow}$, to a match of $H^{\rightarrow}$. So overall, we can find $\mathrm{DM}(G^{\rightarrow}_{\lhd}, H^{\rightarrow})$ in $O(m\kappa^3)$ expected time.

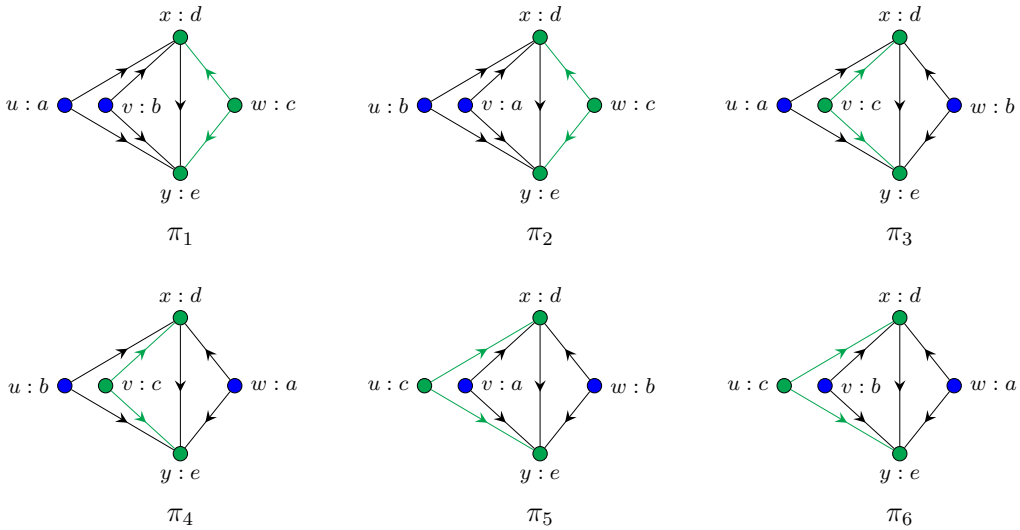
This completes the proof of this lemma. ◀

Lastly, we can prove Theorem 9 as follows.

Proof of Theorem 9. Given a 5-vertex connected subgraph H, we can count all distinct matches of each possible DAG $H^{\rightarrow}$ of H, in $G^{\rightarrow}_{\lhd}$ in $O(m\kappa^3)$ expected time, by Lemma 17. To count all distinct matches of H in G, we just need to sum the number of distinct matches of all possible DAGs (up to isomorphism) of H. The number of such DAGs is constant for H. There are 21 different connected 5-vertex subgraphs (illustrated in [57]), and we perform this process on all of them. This completes the proof of the theorem. ◀



(a) H is 5-vertex connected subgraph and $H^{\rightarrow}$ is one possible acyclic orientation of it. $T_{\max}$ (largest DRTS of $H^{\rightarrow}$) is shown in green and contains three vertices.



(b) All six figures show exactly the same subgraph in $G^{\rightarrow}_{\triangleleft}$. $\pi_1, \ldots, \pi_6$ are six equivalent matches of $H^{\rightarrow}$ in $G^{\rightarrow}_{\triangleleft}$, one for each automorphism of $H^{\rightarrow}$. Notice (u, v, w) being mapped to all permutations of (a, b, c).

Figure 1 Application of Alg. 2 on a DAG $H^{\rightarrow}$ of an example 5-vertex connected subgraph H.

4.2 Limitations of Our Framework for a Six Vertex Subgraph

Consider $\mathcal{C}_6$, shown as H in Fig. 2. Then $H^{\rightarrow}$, shown in the right side of Fig. 2, is a possible DAG of H. In $H^{\rightarrow}$, s_1, s_2, and s_3 are the source vertices, and t_1, t_2, and t_3 are the sink vertices. Any DRTS of $H^{\rightarrow}$ has at most three vertices, and there are three such DRTS, T_1, T_2, and T_3 rooted at s_1, s_2 and s_3, respectively. T_1 is shown by red in Fig. 2. For each of T_1, T_2, and T_3, the remaining vertices include a vertex, with two incoming edges, which we call an in-in wedge. For example, t_2 is such a vertex for T_1. Even graphs with bounded degeneracy can have $\Omega(n^2)$ in-in wedges. We cannot hash the count of such structures in expected time bounded by m and κ. So, Alg. 2 fails to count occurrences of $\mathcal{C}_6$ in the desired time. In the next section, we discuss why such limitations are natural to any framework for the SUB-CNT$_k$ problem at and beyond $k = 6$.

5 A Chasm at Six

At the end of the previous section, we showed the limitations of our framework in counting certain 6-vertex subgraphs. In this section, we show that perhaps such limitations are fundamental to any subgraph counting algorithms. In particular, the landscape of SUB-CNT$_k$ problem in the bounded degeneracy graphs changes dramatically as we move beyond $k = 5$.

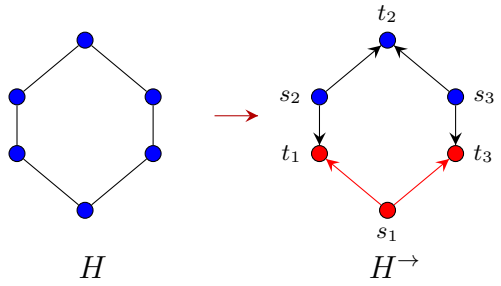


Figure 2 Let $H^{\rightarrow}$ be a DAG of H ($\mathcal{C}_6$). Considering any largest DRTS of $H^{\rightarrow}$, the remaining vertices include a vertex with two incoming edges (in-in wedge). Even graphs with bounded degeneracy can have $\Omega(n^2)$ in-in wedges. So hashing in Alg. 2 will not be bounded by m and κ for H.

We prove that for every integer $k \geqslant 6$, there exists a k-vertex subgraph H such that, the running time of any algorithm for the SUB-CNT$_H$ problem does not depend on the degeneracy of the input graph, assuming the Triangle Detection Conjecture. In contrast, for $k \leqslant 5$, $O(m\kappa^{k-2})$ algorithms exists for SUB-CNT$_k$ (see Section 4). The following theorem captures the main result of this section.

▶ **Theorem 18.** *Assume the* Triangle Detection Conjecture *(Conj. 2). There exists an absolute constant $\gamma > 0$ such that the following holds. For any $k \geqslant 6$ and any function $f : \mathbb{N} \to \mathbb{N}$, there exists a k-vertex subgraph H such that there is no (expected) $o(m^{1+\gamma} f(\kappa))$ algorithm for* SUB-CNT$_H$.

Outline of the Proof. For each $k \geqslant 6$ and $k \neq 8$, the subgraph of interest will be the k-cycle graph, $\mathcal{C}_k$. For $k = 8$, the subgraph of interest will be the $\mathcal{C}_7$ with a tail (see Figure 3). We first give a proof outline. Fix some $k \geqslant 6$ and let H_k denote the target subgraph of size k. Recall the TRI-CNT problem — count the number of triangles in a graph with m edges. Conjecture 2 asserts that for any algorithm $\mathcal{A}$ for the TRI-CNT problem, $T(\mathcal{A}) = \omega(m)$ where $T(\mathcal{A})$ denotes the worst case time complexity of the algorithm $\mathcal{A}$. Our strategy is to reduce from the TRI-CNT problem to the SUB-CNT$_{H_k}$ problem. To this end, we construct a new graph G_k from the input instance G of the TRI-CNT problem such that G_k has $O(m)$ edges, and has degeneracy at most 2. More importantly, the number of triangles in G is a simple linear function of the number of H_k in G_k. Hence, we can derive the number of triangles in G by counting the number of H_k in G_k. As $\kappa(G_k) \leqslant 2$, any $O(mf(\kappa))$ algorithm for the SUB-CNT$_{H_k}$ problem translates to a $O(m)$ algorithm for the TRI-CNT problem, contradicting the Triangle Detection Conjecture. We remark that, for $k = 8$, our proof strategy will be slightly different — instead of reducing from the TRI-CNT problem, we shall reduce from the triangle detection problem itself. However, the gadget construction will follow the same basic principle.

The construction of G_k from G is rather simple. The details of the construction depends on whether k is a multiple of 3 or not. We take two examples to describe the construction.

First, we take $k = 6$, and the target subgraph $H_6 = \mathcal{C}_6$. For each edge e in $E(G)$, we replace e with a length two path $\{e_1, e_2\}$ in $E(G_6)$. To accomplish this, we add a new vertex v_e for each edge: $V(G_6) = V(G) \cup \{v_e\}_{e \in E(G)}$. This is shown in Figure 4a. Each triangle in G creates a $\mathcal{C}_6$ in G_6. We formally prove in Lemma 20 that the number of triangles in G is same as the number of $\mathcal{C}_6$ in G_6. In Lemma 19, we bound the degeneracy of G_6 by 2. This construction can be generalized for any $k = 3\ell$ where $\ell \geqslant 2$, by replacing each edge in $E(G)$ with ℓ-length path.

Next consider the case $k = 7$. For each edge $e \in E(G)$, we first create two parallel copies of e, and then replace the first one with a length two path $\{e_{1,1}, e_{1,2}\}$, and the second one with a length three path $\{e_{2,1}, e_{2,2}, e_{2,3}\}$. So in $E(G_7)$, we have 5 edges for each edge in $E(G)$. We create 3 new vertices per edge to accomplish this, and denote them as v_e, u_{e_1}, u_{e_2}. See Figure 4b for a pictorial demonstration. In Lemma 20, we argue that the number of $\mathcal{C}_7$ is exactly 3 times the number of triangles in G. In Lemma 19, we bound the degeneracy of G_7 by 2. This construction generalizes to any $k = 3\ell + i$ where $\ell \geqslant 2$ and $i \in \{1, 2\}$ (except for the case when $k = 8$, that is $\ell = 2$ and $i = 2$) by splitting each edge into ℓ and $\ell + 1$ many parts respectively.

Finally, we consider the case of $k = 8$. Note that the target subgraph H_8 is the 7-cycle with a tail in this case (see Figure 3). It is natural to wonder why do we not simply take $H_8 = \mathcal{C}_8$? After all, for all other values of k, taking $H_k = \mathcal{C}_k$ suffices. At a first glance, it seems like if we consider the same graph G_7 as described above (and in Figure 4b) the number of $\mathcal{C}_8$ would be a simple linear function of the number of triangles in G — for each triangle in G, there will be exactly three $\mathcal{C}_8$ in G_7. However, each $\mathcal{C}_4$ in G would also lead to a $\mathcal{C}_8$ in G_8. Observe that for $k > 8$, we do not run into this problem. A more formal treatment of this issue appear in Section 5.

So instead, we take H_8 to be the subgraph $\mathcal{C}_7$ with a tail to prove our conditional lower bound for SUB-CNT$_8$. The construction of the graph G_8 remains exactly the same as that of G_7. We show in Lemma 21 that, there exists a $\mathcal{C}_7$ with a tail in G_8 if and only if there exist a triangle in G.

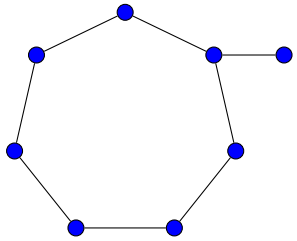


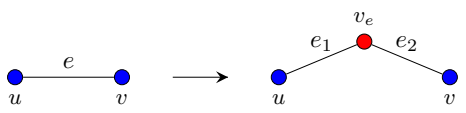
Figure 3 Target subgraph for proving conditional lower bounds for SUB-CNT$_8$: the $\mathcal{C}_7$ with a tail

We now present the proof of Theorem 18 in full details.

Proof of Theorem 18. Fix some $k \geqslant 6$. Let the subgraph H_k denote the target subgraph of size k. For $k \neq 8$, H_k is $\mathcal{C}_k$, and for $k = 8$, H_k is $\mathcal{C}_7$ with a tail (see Figure 3). We reduce from the TRI-CNT problem to the SUB-CNT$_{H_k}$. Let $G = (V, E)$ be the input instance for the TRI-CNT problem with $|V| = n$ and $|E| = m$. We construct an input instance $G_k = (V_k, E_k)$ for the SUB-CNT$_{H_k}$ problem from G. The construction of G_k differs based on whether k is divisible by 3 or not. We next consider these two cases separately.

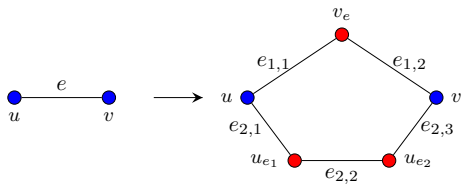
Details of the Reduction. First assume $k = 3\ell$ for some integer $\ell \geqslant 2$. We first define the vertex set V_k. For each vertex in V, we add a vertex in V_k. For each edge $e \in E$, we add a set of $\ell - 1$ many vertices, denoted as $V_e = \{v_{e,1}, v_{e,2}, \ldots, v_{e,\ell-1}\}$. We collect all these second type of vertices into the set V_E. Formally, we have

$$V_k = V \cup V_E\,,$$
$$\text{where } V_E = \bigcup_{e \in E} V_e\,,$$
$$\text{for } V_e = \{v_{e,1}, v_{e,2}, \ldots, v_{e,\ell-1}\}\,.$$



$e \in E(G)$ $\quad e_1, e_2 \in E(G_6)$

(a) Construction of the edge set $E(G_6)$ from the edge set $E(G)$. The red colored nodes are only present in $V(G_6)$, and not in $V(G)$.



$e \in E(G)$ $\quad e_{i,j} \in E(G_7)$

(b) Construction of the edge set $E(G_7)$ from the edge set $E(G)$. The red colored nodes are only present in $V(G_7)$, and not in $V(G)$.

Figure 4 Reduction from the TRI-CNT problem to the SUB-CNT$_{\mathcal{C}_k}$ problem for $k = 6$ (left) and $k = 7$ (right).

We now describe the edge set E_k. We treat each edge $e = \{u, v\} \in E$ as an ordered pair (u, v) where the ordering can be arbitrary of the vertices (for example, assume lexicographical ordering). Now for each edge $e = (u, v)$ construct an ℓ-length path between u and v in V_k by connecting the vertices in $\{u\} \cup V_e \cup \{v\}$ sequentially. More precisely, we define E_k as follows.

$$E_k = \bigcup_{e \in E} E_e \,,$$
$$\text{where } E_e = \{\{u, v_{e,1}\}, \{v_{e,1}, v_{e,2}\}, \ldots, \{v_{e,\ell-2}, v_{e,\ell-1}\}, \{v_{e,\ell-1}, v\}\} \text{ for } e = (u, v)\,.$$

This completes the construction of the graph $G_k = (V_k, E_k)$. We give an example in Figure 4a for $k = 6$.

Now assume $k = 3\ell + i$ for some some integer $\ell \geqslant 2$ and $i \in \{1, 2\}$. In the previous case, we added a set of $\ell - 1$ many vertices for each edge in E. But now, for each edge $e \in E$, we add two sets of vertices, one with $\ell - 1$ many vertices and the other with ℓ many vertices. We denote the first set as $V_e = \{v_{e,1}, v_{e,2}, \ldots, v_{e,\ell-1}\}$, and the second set as $U_e = \{u_{e,1}, u_{e,2}, \ldots, u_{e,\ell}\}$. We also add the set of vertices in V to V_k. Formally, we have

$$V_k = V \cup V_E \,,$$
$$\text{where } V_E = \bigcup_{e \in E} V_e \cup U_e \,,$$
$$\text{for } V_e = \{v_{e,1}, v_{e,2}, \ldots, v_{e,\ell-1}\} \,,$$
$$\text{and } U_e = \{u_{e,1}, u_{e,2}, \ldots, u_{e,\ell}\}\,.$$

To construct the edge set E_k, as before we treat each edge in $e = \{u, v\} \in E$ as an ordered pair (u, v) according to some arbitrary ordering of the vertices. Now, for each edge $e = (u, v)$, construct an $2\ell + 1$-length cycle between u and v in V_k by creating a ℓ-length path via the vertices in V_e and another $\ell + 1$-length path via the vertices in U_e. We denote the corresponding edge sets as $E_{V,e}$ and $E_{U,e}$ respectively. Formally, we define E_k as follows.

$$E_k = \bigcup_{e \in E} (E_{V,e} \cup E_{U,e}) \,,$$
$$\text{where } E_{V,e} = \{\{u, v_{e,1}\}, \{v_{e,1}, v_{e,2}\}, \ldots, \{v_{e,\ell-2}, v_{e,\ell-1}\}, \{v_{e,\ell-1}, v\}\} \,,$$
$$\text{and } E_{U,e} = \{\{u, u_{e,1}\}, \{u_{e,1}, u_{e,2}\}, \ldots, \{u_{e,\ell-1}, u_{e,\ell}\}, \{u_{e,\ell}, v\}\} \text{ for } e = (u, v)\,.$$

This completes the construction of the graph $G_k = (V_k, E_k)$. Note that the construction is independent of the value of i. Hence, we produce the same graph G_k for $k = 3\ell + 1$ and $k = 3\ell + 2$. We give an example in Figure 4b for $k = 7$.

Note that although our target subgraph for the case $k = 8$ is a 7-cycle with a tail instead of 8-cycle, our construction is still the same.

Correctness of the Reduction. In Lemma 19, we prove that G_k has degeneracy at most 2. In Lemma 20, we show that, for $k \neq 8$, the number of $\mathcal{C}_k$ in the graph G_k is a linear function of the number of triangles in G. In Lemma 21, we show that G_8 is H_8 free if and only if G is triangle free.

▶ **Lemma 19.** $\kappa(G_k) \leqslant 2$.

Proof. To prove the lemma it is sufficient to exhibit a vertex ordering $\prec$ such that in the corresponding directed graph $G^{\rightarrow}_{\prec}$, $d^+(v) \leqslant 2$ for all $v \in V_k$ (application of Lemma 6). We use an ordering $\prec$ where $V_E \prec V$ and the ordering within each set is arbitrary. Observe that each vertex $v \in V_E$ has degree exactly 2 and no two vertices in V are connected to each other. Hence, $d^+(v) \leqslant 2$ for all $v \in V_k$. ◀

▶ **Lemma 20.** *Let $\ell \geqslant 2$ be some integer. For $k = 3\ell$, $\mathrm{DM}(G_k, \mathcal{C}_k) = \mathrm{DM}(G, \mathcal{C}_3)$. For $k = 3\ell + i$ with $i \in \{1, 2\}$ and $k \neq 8$, $\mathrm{DM}(G_k, \mathcal{C}_k) = 3 \cdot \mathrm{DM}(G, \mathcal{C}_3)$.*

Proof. Let $\mathcal{T}$ be the set of triangles in G and $\mathcal{C}$ be the set of $\mathcal{C}_k$ in G_k. Note that a triangle in $\mathcal{T}$ and a k-cycle in $\mathcal{C}$ can be uniquely identified by a set of three and k edges, respectively.

We first take up case of $k = 3\ell$ for some $\ell \geqslant 2$. Let g be the mapping between the sets $\mathcal{T}$ and $\mathcal{C}$, $g : \mathcal{T} \to \mathcal{C}$, defined as follows: $g(\{e_1, e_2, e_3\}) = E_{e_1} \cup E_{e_2} \cup E_{e_3}$. To prove the lemma, it is sufficient to exhibit that g is a bijection. To this end, note that if $g(\tau_1) = g(\tau_2)$, then $\tau_1 = \tau_2$. This follows immediately from the definition of g, since $E_{e_1} \cap E_{e_2} = \emptyset$ for all $e_1 \neq e_2$. We now show that every k-cycle in $\mathcal{C}$ has an inverse mapping in g. Let ξ be a k-cycle in $\mathcal{C}$. Fix some edge $e \in E$. By construction, either all the edges from the set E_e are present in ξ, or none of them are. Hence, ξ must be of the form $E_{e_1} \cup E_{e_2} \cup E_{e_3}$ for some three distinct edges e_1, e_2, and e_3. Clearly, $\{e_1, e_2, e_3\}$ forms a triangle in G.

Now assume $k = 3\ell + i$ for some $\ell \geqslant 2$ and $i \in \{1, 2\}$, and $k \neq 8$. It is not difficult to see that each triangle in $\mathcal{T}$ leads to exactly three k-cycles in $\mathcal{C}$. The non-trivial direction is to show that for each k-cycles in $\mathcal{C}$ there is an unique triangle in $\mathcal{T}$. Let ξ be a k-cycle in $\mathcal{C}$. Fix some edge $e \in E$. By construction, exactly one of the following must be true: (i) all the ℓ edges from the set $E_{V,e}$ are present in ξ, (ii) all the $\ell + 1$ edges from the set $E_{U,e}$ are present in ξ, (iii) none of the edges from the set $E_{V,e} \cup E_{U,e}$ are present in ξ. First assume $i = 1$. Since ξ has $3\ell + 1$ many edges, and $\ell \geqslant 2$, it must consist of one $E_{U,e}$ set of size $\ell + 1$, and two $E_{V,e}$ sets of size ℓ. When $i = 2$ and $\ell > 2$, ξ must consist of two $E_{U,e}$ set of size $\ell + 1$, and one $E_{V,e}$ sets of size ℓ. Clearly, the three edges corresponding to these sets form a unique triangle in G. (When $k = 8$, that is $\ell = 2$ and $i = 2$, taking four distinct sets $E_{V,e}$ creates a copy of $\mathcal{C}_8$, and hence the argument does not work.) ◀

▶ **Lemma 21.** *The input graph G is triangle free if and only if G_8 does not have any $\mathcal{C}_7$ with a tail.*

Proof. Observe that, if there exists a triangle τ in G, then in G_8, there would be at least one $\mathcal{C}_7$ with a tail (in fact, the exact number would depend on the degree of the involved vertices). In the proof of Lemma 20, we argued that each 7-cycle in G_7 (which is isomorphic to G_8) corresponds to a triangle in G. Also, by our construction, if G_8 has a $\mathcal{C}_7$, then that 7-cycle necessarily has a tail. Therefore, existence of $\mathcal{C}_7$ with a tail in G_8 implies existence of a triangle in G. This completes the proof of the lemma. ◀

Lemmas 19 to 21 together prove the theorem: if there exists an algorithm $\mathcal{A}$ for the SUB-CNT$_{\mathcal{C}_k}$ problem with $T(\mathcal{A}) = O(mf(\kappa))$, then $\mathcal{A}$ is an algorithm for the TRI-CNT problem (or the triangle detection problem in the case of $k = 8$) with $T(\mathcal{A}) = O(m)$, where $T(\mathcal{A})$ denotes the worst case time complexity of the algorithm $\mathcal{A}$. ◀

6 Future Directions

Although our algorithmic framework fails to produce a linear time algorithm for SUB-CNT$_{\mathcal{C}_6}$ in bounded degeneracy graphs, there are certain other 6-vertex subgraphs where it indeed succeeds. An easy example is SUB-CNT$_{\mathcal{K}_6}$. In fact, our framework gives a linear time algorithm for counting any constant size clique in bounded degeneracy graphs – for each acyclic orientation of a clique, the source vertex construct a DRTS covering all the remaining vertices. There exists other non-clique 6-vertex subgraphs as well, where Alg. 2 succeeds. Consider the subgraph H shown in Fig. 5. It is easy to see that, any acyclic orientation of H such as $H^{\rightarrow}$ has at least one source vertex s that is a root of a DRTS with four vertices. Thus, we can solve SUB-CNT$_H$ in $O(m\kappa^3)$ expected time.

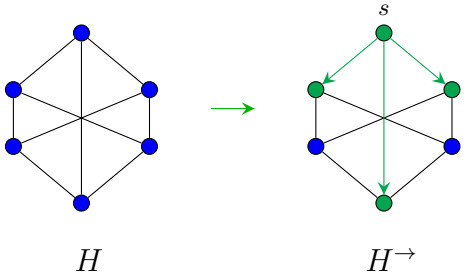


Figure 5 Alg. 2 succeeds to count the number of distinct matches of H in linear time for bounded (constant) degeneracy graphs. Each acyclic orientation of H has a source vertex s, which is connected to exactly three vertices, as in $H^{\rightarrow}$. So, the largest DRTS has at least four vertices (shown in green). Number of matches of the remaining vertices (shown in blue) could be counted using $\mathcal{HM}_2$.

Despite the chasm at six, there exist subgraphs H with 6-vertices (or more) such that SUB-CNT$_H$ admits a linear time algorithm in bounded degeneracy graph. We end this exposition with the following natural problem:

Characterize all subgraphs H such that SUB-CNT$_H$ *has a linear time algorithm in bounded degeneracy graphs.*

References

1 Amir Abboud and Virginia Vassilevska Williams. Popular conjectures imply strong lower bounds for dynamic problems. In *Proc. 55th Annual IEEE Symposium on Foundations of Computer Science*, 2014.

2 Nesreen K. Ahmed, Jennifer Neville, Ryan A. Rossi, and Nick Duffield. Efficient Graphlet Counting for Large Networks. In *International Conference on Data Mining*, 2015.

3 Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. Graph sketches: sparsification, spanners, and subgraphs. In *Proc. 31st ACM Symposium on Principles of Database Systems*, pages 5–14. ACM, 2012.

4 Noga Alon, Raphael Yuster, and Uri Zwick. Color-coding. *J. ACM*, 42(4):844–856, 1995.

5 Noga Alon, Raphael Yuster, and Uri Zwick. Finding and counting given length cycles. *Algorithmica*, 17(3):209–223, 1997.

6 Sepehr Assadi, Michael Kapralov, and Sanjeev Khanna. A Simple Sublinear-Time Algorithm for Counting Arbitrary Subgraphs via Edge Sampling. In *Proc. 10th Conference on Innovations in Theoretical Computer Science*, 2018.

7 Ziv Bar-Yossef, Ravi Kumar, and D. Sivakumar. Reductions in Streaming Algorithms, with an Application to Counting Triangles in Graphs. In *Proc. 13th Annual ACM-SIAM Symposium on Discrete Algorithms*, 2002.

8 A. Benson, D. F. Gleich, and J. Leskovec. Higher-order organization of complex networks. *Science*, 353(6295):163–166, 2016.

9 Suman K Bera and Amit Chakrabarti. Towards tighter space bounds for counting triangles and other substructures in graph streams. In *Proc. 34th International Symposium on Theoretical Aspects of Computer Science*, 2017.

10 Nadja Betzler, Rene Van Bevern, Michael R Fellows, Christian Komusiewicz, and Rolf Niedermeier. Parameterized algorithmics for finding connected motifs in biological networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 8(5):1296–1308, 2011.

11 M. Bhuiyan, M. Rahman, M. Rahman, and M. Al Hasan. GUISE: Uniform Sampling of Graphlets for Large Graph Analysis. In *International Conference on Data Mining*, pages 91–100, 2012.

12 Etienne Birmele et al. Detecting local network motifs. *Electronic Journal of Statistics*, 6:908–933, 2012.

13 Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. Counting paths and packings in halves. In *Proc. 17th Annual European Symposium on Algorithms*, pages 578–586, 2009.

14 Andreas Björklund, Petteri Kaski, and Łukasz Kowalik. Counting thin subgraphs via packings faster than meet-in-the-middle time. *ACM Transactions on Algorithms (TALG)*, 13(4):48, 2017.

15 R. Burt. Structural Holes and Good Ideas. *American Journal of Sociology*, 110(2):349–399, 2004.

16 Norishige Chiba and Takao Nishizeki. Arboricity and subgraph listing algorithms. *SIAM Journal on computing*, 14(1):210–223, 1985.

17 Jonathan Cohen. Graph twiddling in a mapreduce world. *Computing in Science & Engineering*, 11(4):29, 2009.

18 J. Coleman. Social Capital in the Creation of Human Capital. *American Journal of Sociology*, 94:S95–S120, 1988. URL: `http://www.jstor.org/stable/2780243`.

19 Radu Curticapean, Holger Dell, and Dániel Marx. Homomorphisms are a good basis for counting small subgraphs. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 210–223, 2017.

20 Radu Curticapean and Dániel Marx. Complexity of counting subgraphs: Only the boundedness of the vertex-cover number counts. In *Proc. 55th Annual IEEE Symposium on Foundations of Computer Science*, pages 130–139, 2014.

21 Reinhard Diestel. *Graph Theory, Fourth Edition*. Springer, 2010.

22 Talya Eden, Amit Levi, Dana Ron, and C Seshadhri. Approximately counting triangles in sublinear time. *SIAM Journal on Computing*, 46(5):1603–1646, 2017.

23 Talya Eden, Dana Ron, and C Seshadhri. On approximating the number of k-cliques in sublinear time. In *Proc. 50th Annual ACM Symposium on the Theory of Computing*, pages 722–734, 2018.

24 Friedrich Eisenbrand and Fabrizio Grandoni. On the complexity of fixed parameter clique and dominating set. *Theoretical Computer Science*, 326(1-3):57–67, 2004.

25 Ethan R Elenberg, Karthikeyan Shanmugam, Michael Borokhovich, and Alexandros G Dimakis. Beyond triangles: A distributed framework for estimating 3-profiles of large graphs. In *Proc. 12th Annual SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 229–238. ACM, 2015.

26 Ethan R Elenberg, Karthikeyan Shanmugam, Michael Borokhovich, and Alexandros G Dimakis. Distributed estimation of graph 4-profiles. In *Proc. 25th Proceedings, International World Wide Web Conference (WWW)*, pages 483–493. International World Wide Web Conferences Steering Committee, 2016.

27 David Eppstein. Arboricity and bipartite subgraph listing algorithms. *Information processing letters*, 51(4):207–211, 1994.

28 Mira Gonen and Yuval Shavitt. Approximating the number of network motifs. *Internet Mathematics*, 6(3):349–372, 2009.

29 Tomaž Hočevar and Janez Demšar. A combinatorial approach to graphlet counting. *Bioinformatics*, 30(4):559–565, 2014.

30 Tomaž Hočevar and Janez Demšar. Combinatorial algorithm for counting small induced graphs and orbits. *PloS ONE*, 12(2):e0171428, 2017.

31 Tomaž Hočevar, Janez Demšar, et al. Computation of graphlet orbits for nodes and edges in sparse graphs. *Journ. Stat. Soft*, 71, 2016.

32 P. Holland and S. Leinhardt. A method for detecting structure in sociometric data. *American Journal of Sociology*, 76:492–513, 1970.

33 F. Hormozdiari, P. Berenbrink, N. Prulj, and S. Cenk Sahinalp. Not All Scale-Free Networks Are Born Equal: The Role of the Seed Graph in PPI Network Evolution. *PLoS Computational Biology*, 118, 2007.

34 Alon Itai and Michael Rodeh. Finding a minimum circuit in a graph. *SIAM Journal on Computing*, 7(4):413–423, 1978.

35 Shweta Jain and C Seshadhri. A Fast and Provable Method for Estimating Clique Counts Using Turán's Theorem. In *Proc. 26th Proceedings, International World Wide Web Conference (WWW)*, pages 441–449. International World Wide Web Conferences Steering Committee, 2017.

36 Madhav Jha, C Seshadhri, and Ali Pinar. A space efficient streaming algorithm for triangle counting using the birthday paradox. In *Proc. 19th Annual SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 589–597, 2013.

37 Madhav Jha, C Seshadhri, and Ali Pinar. Path sampling: A fast and provable method for estimating 4-vertex subgraph counts. In *Proc. 24th Proceedings, International World Wide Web Conference (WWW)*, pages 495–505. International World Wide Web Conferences Steering Committee, 2015.

38 Hossein Jowhari and Mohammad Ghodsi. New streaming algorithms for counting triangles in graphs. In *Computing and Combinatorics*, pages 710–716, 2005.

39 Daniel M Kane, Kurt Mehlhorn, Thomas Sauerwald, and He Sun. Counting arbitrary subgraphs in data streams. In *Proc. 39th International Colloquium on Automata, Languages and Programming*, pages 598–609, 2012.

40 Arijit Khan, Nan Li, Xifeng Yan, Ziyu Guan, Supriyo Chakraborty, and Shu Tao. Neighborhood based fast graph search in large networks. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, 2011.

41 Ton Kloks, Dieter Kratsch, and Haiko Müller. Finding and counting small induced subgraphs efficiently. *Information Processing Letters*, 74(3-4):115–121, 2000.

42 Tamara G Kolda, Ali Pinar, Todd Plantenga, C Seshadhri, and Christine Task. Counting triangles in massive graphs with MapReduce. *SIAM Journal on Scientific Computing*, 36(5):S48–S77, 2014.

43 Tsvi Kopelowitz, Seth Pettie, and Ely Porat. Higher lower bounds from the 3SUM conjecture. In *Proc. 27th Annual ACM-SIAM Symposium on Discrete Algorithms*, 2016.

44 Ioannis Koutis and Ryan Williams. Limits and applications of group algebras for parameterized problems. In *International Colloquium on Automata, Languages and Programming*, pages 653–664, 2009.

45 Mirosław Kowaluk, Andrzej Lingas, and Eva-Marta Lundell. Counting and detecting small subgraphs via equations. *SIAM Journal on Discrete Mathematics*, 27(2):892–909, 2013.

46 Madhusudan Manjunath, Kurt Mehlhorn, Konstantinos Panagiotou, and He Sun. Approximate Counting of Cycles in Streams. In *Proc. 19th Annual European Symposium on Algorithms*, pages 677–688, 2011.

47 Dror Marcus and Yuval Shavitt. Efficient counting of network motifs. In *IEEE 30th International Conference on Distributed Computing Systems Workshops*, pages 92–98. IEEE, 2010.

48 David W Matula and Leland L Beck. Smallest-last ordering and clustering and graph coloring algorithms. *Journal of the ACM (JACM)*, 30(3):417–427, 1983.

49 Andrew McGregor, Sofya Vorotnikova, and Hoa T. Vu. Better Algorithms for Counting Triangles in Data Streams. In *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pages 401–411, 2016.

50 Tijana Milenković and Nataša Pržulj. Uncovering biological network function via graphlet degree signatures. *Cancer informatics*, 6:CIN–S680, 2008.

51 Burkhard Monien. How to find long paths efficiently. In *North-Holland Mathematics Studies*, volume 109, pages 239–254. Elsevier, 1985.

52 C. St. J. A. Nash-Williams. Decomposition of finite graphs into forests. *Journal of the London Mathematical Society*, 39(1):12, 1964.

53 Jaroslav Nešetřil and Svatopluk Poljak. On the complexity of the subgraph problem. *Commentationes Mathematicae Universitatis Carolinae*, 26(2):415–419, 1985.

54 Mark Ortmann and Ulrik Brandes. Efficient orbit-aware triad and quad census in directed and undirected graphs. *Applied network science*, 2(1), 2017.

55 Mihai Patrascu. Towards polynomial lower bounds for dynamic problems. In *Proc. 42nd Annual ACM Symposium on the Theory of Computing*, 2010.

56 Aduri Pavan, Kanat Tangwongsan, Srikanta Tirthapura, and Kun-Lung Wu. Counting and sampling triangles from a graph stream. *Proceedings of the VLDB Endowment*, 6(14):1870–1881, 2013.

57 Ali Pinar, C Seshadhri, and Vaidyanathan Vishal. Escape: Efficiently counting all 5-vertex subgraphs. In *Proceedings, International World Wide Web Conference (WWW)*, pages 1431–1440. International World Wide Web Conferences Steering Committee, 2017.

58 Alejandro Portes. Social Capital: Its Origins and Applications in Modern Sociology. *Annual Review of Sociology*, 24(1):1–24, 1998. doi:10.1146/annurev.soc.24.1.1.

59 Natasa Przulj. Biological network comparison using graphlet degree distribution. *Bioinformatics*, 23(2):177–183, 2007.

60 Natasa Przulj, Derek G. Corneil, and Igor Jurisica. Modeling interactome: scale-free or geometric? *Bioinformatics*, 20(18):3508–3515, 2004.

61 M. Rahman, M. A. Bhuiyan, and M. Al Hasan. GRAFT: An Efficient Graphlet Counting Method for Large Graph Analysis. *IEEE Transactions on Knowledge and Data Engineering*, PP(99), 2014.

62 Ahmet Erdem Sariyuce, C. Seshadhri, Ali Pinar, and Umit V. Catalyurek. Finding the Hierarchy of Dense Subgraphs Using Nucleus Decompositions. In *Proceedings, International World Wide Web Conference (WWW)*, pages 927–937, 2015.

63 T. Schank and D. Wagner. Finding, Counting and Listing All Triangles in Large Graphs, an Experimental Study. In *Experimental and Efficient Algorithms*, pages 606–609. Springer, 2005.

64 C. Seshadhri and Srikanta Tirthapura. Scalable Subgraph Counting: The Methods Behind The Madness: WWW 2019 Tutorial. In *Proceedings, International World Wide Web Conference (WWW)*, 2019.

65 Alina Stoica and Christophe Prieur. Structure of neighborhoods in a large social network. In *International Conference on Computational Science and Engineering*, volume 4, pages 26–33. IEEE, 2009.

66 Siddharth Suri and Sergei Vassilvitskii. Counting triangles and the curse of the last reducer. In *Proceedings of the 20th international conference on World wide web*, pages 607–614, 2011.

67 Charalampos E. Tsourakakis. The K-clique Densest Subgraph Problem. In *Proceedings, International World Wide Web Conference (WWW)*, pages 1122–1132, 2015.

68 Johan Ugander, Lars Backstrom, and Jon M. Kleinberg. Subgraph frequencies: mapping the empirical and extremal geography of large graph collections. In *Proceedings, International World Wide Web Conference (WWW)*, pages 1307–1318, 2013.

69 Virginia Vassilevska. Efficient algorithms for clique problems. *Information Processing Letters*, 109(4):254–257, 2009.

70 Virginia Vassilevska and Ryan Williams. Finding, minimizing, and counting weighted subgraphs. In *Proc. 41st Annual ACM Symposium on the Theory of Computing*, pages 455–464, 2009.

71 Pinghui Wang, Junzhou Zhao, Xiangliang Zhang, Zhenguo Li, Jiefeng Cheng, John CS Lui, Don Towsley, Jing Tao, and Xiaohong Guan. MOSS-5: A fast method of approximating counts of 5-node graphlets in large graphs. *IEEE Transactions on Knowledge and Data Engineering*, 30(1):73–86, 2017.

72 S. Wernicke and F. Rasche. FANMOD: a tool for fast network motif detection. *Bioinformatics*, 22(9):1152–1153, 2006.

73 Sebastian Wernicke. Efficient detection of network motifs. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 3(4):347–359, 2006.

74 Virginia Vassilevska Williams, Joshua R Wang, Ryan Williams, and Huacheng Yu. Finding four-node subgraphs in triangle time. In *Proc. 26th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1671–1680, 2014.

75 Zhao Zhao, Guanying Wang, Ali R Butt, Maleq Khan, VS Anil Kumar, and Madhav V Marathe. Sahad: Subgraph analysis in massive networks using hadoop. In *IEEE 26th International Parallel and Distributed Processing Symposium*, pages 390–401. IEEE, 2012.

Parameterization Above a Multiplicative Guarantee

Fedor V. Fomin
Department of Informatics, University of Bergen, Norway
fomin@ii.uib.no

Petr A. Golovach
Department of Informatics, University of Bergen, Norway
petr.golovach@uib.no

Daniel Lokshtanov
University of California, Santa Barbara, USA
daniello@ucsb.edu

Fahad Panolan
Indian Institute of Technology Hyderabad, India
fahad@iith.ac.in

Saket Saurabh
Department of Informatics, University of Bergen, Norway
The Institute of Mathematical Sciences, HBNI and IRL 2000 ReLaX, Chennai, India
saket@imsc.res.in

Meirav Zehavi
Ben-Gurion University of the Negev, Beersheba, Israel
meiravze@bgu.ac.il

Abstract

Parameterization above a guarantee is a successful paradigm in Parameterized Complexity. To the best of our knowledge, all fixed-parameter tractable problems in this paradigm share an *additive form* defined as follows. Given an instance (I, k) of some (parameterized) problem Π with a *guarantee* $g(I)$, decide whether I admits a solution of size at least (at most) $k + g(I)$. Here, $g(I)$ is usually a lower bound (resp. upper bound) on the maximum (resp. minimum) size of a solution. Since its introduction in 1999 for Max SAT and Max Cut (with $g(I)$ being half the number of clauses and half the number of edges, respectively, in the input), analysis of parameterization above a guarantee has become a very active and fruitful topic of research.

We highlight a *multiplicative* form of parameterization above a guarantee: Given an instance (I, k) of some (parameterized) problem Π with a guarantee $g(I)$, decide whether I admits a solution of size at least (resp. at most) $k \cdot g(I)$. In particular, we study the Long Cycle problem with a multiplicative parameterization above the girth $g(I)$ of the input graph, and provide a parameterized algorithm for this problem. Apart from being of independent interest, this exemplifies how parameterization above a multiplicative guarantee can arise naturally. We also show that, for any fixed constant $\epsilon > 0$, multiplicative parameterization above $g(I)^{1+\epsilon}$ of Long Cycle yields para-NP-hardness, thus our parameterization is tight in this sense. We complement our main result with the design (or refutation of the existence) of algorithms for other problems parameterized multiplicatively above girth.

2012 ACM Subject Classification Theory of computation → Fixed parameter tractability

Keywords and phrases Parameterized Complexity, Above-Guarantee Parameterization, Girth

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.39

11th Innovations in Theoretical Computer Science Conference (ITCS 2020).
Editor: Thomas Vidick; Article No. 39; pp. 39:1–39:13

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Funding *Fedor V. Fomin*: Research Council of Norway via the project MULTIVAL.
Petr A. Golovach: Research Council of Norway via the project MULTIVAL.
Daniel Lokshtanov: European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant no. 715744), and United States - Israel Binational Science Foundation grant no. 2018302.
Saket Saurabh: European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant no. 819416), and Swarnajayanti Fellowship grant DST/SJF/MSA-01/2017-18.
Meirav Zehavi: Israel Science Foundation grant no. 1176/18, and United States – Israel Binational Science Foundation grant no. 2018302.

1 Introduction

The goal of parameterized complexity is to find ways of solving NP-hard problems more efficiently than brute force: our aim is to restrict the combinatorial explosion to a parameter that is hopefully much smaller than the input size. Formally, a *parameterization* of a problem is the assignment of an integer k to each input instance, and we say that a parameterized problem is *fixed-parameter tractable (FPT)* if there is an algorithm that solves the problem in time $f(k) \cdot n^{O(1)}$, where n is the size of the input and f is an arbitrary computable function depending on the parameter k only. There is a long list of NP-hard problems that are FPT under various parameterizations: finding a vertex cover of size k, finding a cycle of length k, finding a maximum independent set in a graph of treewidth at most k, etc. For more background, the reader is referred to the monographs [12, 14, 21].

Choosing a suitable parameter plays an important role in the field of parameterized complexity. Traditionally, the *solution size* has been the most sought after parameter. However, in various circumstances this is not a good parameter. To illustrate this, consider the following problems: Max Sat and Max Cut. Observe that there always exists a truth assignment that satisfies half of the clauses, and there is always a max-cut containing at least half the edges. Thus, if we choose solution size as the parameter, then we get the following trivial algorithm: if $k \leq m/2$, then return yes; else $m \leq 2k$, so now any brute-force algorithm is an FPT algorithm. Thus if we want to design a nontrivial parameterized algorithm, then solution size is *not a suitable parameter*. In particular, a general message here is as follows.

> The natural parameterization of, say, a maximization/minimization problem by the solution size is not satisfactory if there is a lower bound for the solution size that is sufficiently large.

Thus, for such cases, it is more natural to parameterize the problem by *the difference between the solution size and the bound*. This perspective is known as "above guarantee" parameterization. This approach was introduced by Mahajan and Raman [33] for the Max Sat and Max Cut problem. This approach was successfully applied to many various problems (see, e.g., [1, 11, 24, 25, 26, 34, 10] for a few illustrative examples).

In some of the above examples there is an explicit lower bound on the solution size, given in terms of the input size. However, in many cases, we do not have explicit lower bounds. We substantiate this with the example of classic Vertex Cover problem. In this problem, we are given a graph G and a positive integer k, and the goal is to test whether there exists a set of vertices C of size at most k that is a vertex cover (i.e., every edge has at least one endpoint in C). Clearly, the size of a *maximum matching* of G or the value of the linear programming relaxation of an integer linear program for Vertex Cover is a lower bound on the size of a vertex cover of G. Observe that these lower bounds are graph dependent

and not the size dependent. This is what we mean by implicit lower bounds. Coming back to Vertex Cover, in polynomial time it is possible to reduce the size of the graph, without changing the answer, such that we are guaranteed that any vertex cover must contain half of the remaining vertices. Thus, again we can employ any brute-force algorithm and design an FPT algorithm for Vertex Cover. This has led to the study of the problem Vertex Cover Above LP (or Vertex Cover Above Matching), which has played a central role in the development of the field of parameterized complexity [12].

In this paper, we take the philosophy of above guarantee parameterization a significant conceptual step forwards. In particular,

> The goal of this paper is to study another classical problem – namely, Long Cycle – from the viewpoint of a new implicit parameterization, that is neither standard nor an additive above guarantee parameterization.

The Long Path problem is to decide, given a directed or undirected n-vertex graph G and an integer k, whether G contains a path on at least k vertices, that is, a self-avoiding walk with at least k vertices. Similarly, the Long Cycle problem is to decide whether G contains a cycle of length at least k, that is, a closed self-avoiding walk with at least k vertices. These problems are natural generalization of the classical Hamiltonian Path and Hamiltonian Cycle problems and have been actively studied. In particular, there is a plethora of results about parameterized complexity of Long Path and Long Cycle (see, e.g., [4, 5, 9, 8, 18, 22, 29, 30, 31, 38]) since the early work of Monien [35]. Let us just mention here that the fastest known randomized algorithm for Long Path is due to Björklund et al. [4] and runs in time $1.657^k \cdot n^{\mathcal{O}(1)}$, whereas the fastest known deterministic algorithm is due to Tsur [37] and runs in time $2.554^k \cdot n^{\mathcal{O}(1)}$. Respectively for Long Cycle, the randomized algorithm with the currently best running time of $4^k \cdot n^{\mathcal{O}(1)}$ was given by Zehavi in [40], and the best deterministic algorithm was given by Fomin et al. in [20] and runs in time $4.884^k \cdot n^{\mathcal{O}(1)}$.

For Long Path, the investigation of above guarantee parameterizations was initiated by Bezáková et al. in [2]. Let s and t be two vertices of a graph G. Clearly, the length of any (s,t)-path in G is lower bounded by the shortest distance, $d(s,t)$, between these vertices. Based on this straightforward observation, Bezáková et al. in [2] introduced the Longest Detour problem that asks, given a graph G, two vertices s,t, and a positive integer k, whether G has an (s,t)-path with at least $d(s,t)+k$ vertices. They proved that for undirected graphs, this problem can be solved in time $2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$. That is, it is FPT. For the variant of the problem where the question is whether G has an (s,t)-path with *exactly* $d(s,t)+k$ vertices, a randomized algorithm with running time $2.746^k \cdot n^{\mathcal{O}(1)}$ and a deterministic algorithm with running time $6.745^k \cdot n^{\mathcal{O}(1)}$ were obtained. In a recent work, Fomin et al. [17] studied the parameterization of Long Path and Long Cycle above the degeneracy d of the input graph G. Formally, a graph G has degeneracy at most d if every subgraph of G has a vertex of degree at most d. A classic result by Erdős and Gallai [15] from 1959 states that any graph of degeneracy $d > 1$ has a cycle (and hence also path) on at least d vertices. If the graph G is not guaranteed to be 2-connected, then deciding whether G contains a cycle of length $d+2$ is already NP-hard, and therefore parameterization above degeneracy does not make sense. However, when we add the requirement that G is 2-connected, then then deciding whether G contains a cycle of length $d+k$ parameterized by k can be done in time $2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$ and hence FPT. A similar situation holds for Long Path where connectivity replaces 2-connectivity.

1.1 Multiplicative Above Guarantee Parameterization

To the best of our knowledge, all successful above guarantee parameterizations are additive. Roughly speaking, this means that if we have a lower bound τ on the optimal solution size, then we ask whether we can find a solution of size (at least or at most) $\tau + p$ in time $f(p) \cdot n^{\mathcal{O}(1)}$. Our main message of this paper is the following.

> The goal of this paper is to introduce the notion of multiplicative above guarantee parameterization (which is neither standard nor an additive above guarantee parameterization). In multiplicative above guarantee parameterization, we ask whether we can find a solution of size (at least or at most) $\tau \cdot p$ in time $g(p) \cdot n^{\mathcal{O}(1)}$. We will illustrate our new definition by designing several FPT algorithms parameterized above a multiplicative guarantee.

We remark that a few problems in [36] were stated in a way fitting parameterization above a multiplicative guarantee. However, these were shown to be para-NP-hard [27, 28].

Recall that the *girth* of a graph G, denoted by $\gamma(G)$, is the length of the shortest cycle in G. First, consider the problem where we are given a graph G and an integer k and the objective is to test whether there is a cycle of length at least $\gamma(G)+k$ in G. If $\gamma(G) \leq 2k+6$, then clearly we can solve the problem in time $2^{\mathcal{O}(k)}n^{\mathcal{O}(1)}$ by using the algorithm in [39]. We now show that when $2k + 6 < \gamma(G)$, the problem is solvable in polynomial time. To this end, let (G, k) be a yes-instance, and let C be a hypothetical solution. That is, C is a cycle of length at least $\gamma(G) + k$. Let $g = \gamma(G)$. Let $u, v, w, x \in V(C)$ such that when we traverse C in some order from u, we encounter v at distance $\lfloor \frac{g}{2} \rfloor - 1$ from u, next we encounter w at additional distance $\lfloor \frac{g}{2} \rfloor - 1$ from v, and lastly we encounter x at additional distance $k' = (g + k) - (2\lfloor \frac{g}{2} \rfloor - 2)$ from w. Notice that $k' \in \{k + 2, k + 3\}$. Since the girth of G is g and $k + 3 < \frac{g}{2}$, we have that shortest paths between u and v, v and w, and w and x are unique and they are part of the cycle C. Therefore, we may compute the shortest paths between the pairs above, and later check whether u is reachable from x after deleting these shortest paths. Going over every possible choice of $u, v, w, x \in V(C)$, this leads to a polynomial time algorithm when $2k + 6 < \gamma(G)$. This implies that it can be decided in time $2^{\mathcal{O}(k)}n^{\mathcal{O}(1)}$ whether a graph has a cycle with at least $\gamma(G) + k$ vertices.

The informal argument above shows that Long Cycle parameterized additively above girth is not more "interesting" than just normal Long Cycle, where the input lower bound on solution size is the parameter. In a sense, it hints that the guarantee may be strengthened. In light of this, we introduce a new above guarantee version of Long Cycle (resp. Long Path), termed Long Cycle (resp. Long Path) parameterized multiplicatively above girth, as follows. The input consists of a graph G and an integer k, and the objective is to decide whether there is a cycle (resp. path) on at least $k \cdot \gamma(G)$ vertices.

1.2 Our Contribution

We first prove our main result, which states that Long Cycle (and Long Path) parameterized multiplicatively above girth is FPT. Specifically, we prove the following theorem.

▶ **Theorem 1.** Long Cycle *(and* Long Path*) parameterized multiplicatively above girth is solvable in time* $2^{\mathcal{O}(k^2)}n$.

As a complementary result to the above theorem, we assert that our parameterization is tight in the following sense.

▶ **Theorem 2.** *For any fixed constant $\epsilon > 0$,* Long Cycle *(and* Long Path*) parameterized multiplicatively above $g^{1+\epsilon}$ is para-NP-hard, where g is the girth of the input graph.*

Next, we further extend the scope of our problem domain by showing that also Vertex Cover, Connected Vertex Cover and Max Internal Spanning Tree parameterized multiplicatively above girth are FPT. Given a graph G and an integer k, the objectives of these problems are to decide whether G has a vertex cover of size at most $k \cdot \gamma(G)$, a connected vertex cover (that is, a vertex cover that induces a connected subgraph) of size at most $k \cdot \gamma(G)$, and a spanning tree with at least $k \cdot \gamma(G)$ internal vertices, respectively. Here, $\gamma(G)/2$ is a lower bound on the size of an optimal solution.

▶ **Theorem 3.** Vertex Cover*,* Connected Vertex Cover *and* Max Internal Spanning Tree *parameterized multiplicatively above girth are solvable in time $2^{\mathcal{O}(k \log k)}n$.*

Lastly, we observe that parameterization above girth (even additively) can often yield NP-hardness when k is a fixed constant. Specifically, we give Feedback Vertex Set and Cycle Packing as illustrative simple examples.

▶ **Theorem 4.** Feedback Vertex Set *and* Cycle Packing *parameterized additively above girth are para-NP-hard.*

2 Preliminaries

For terminology not explicitly defined here, we refer to the book of Diestel [13]. Throughout the paper, we consider finite undirected graphs. For a graph G, let $V(G)$ and $E(G)$ denote its vertex set and edge set, respectively. When G is clear from context, let $n = |V(G)|$ and $m = |E(G)|$. Given a subset $U \subseteq V(G)$, let $G[U]$ be the subgraph of G induced by U. The *girth* of G is the length of the shortest cycle in G,[1] and is denoted by $\gamma(G)$. A *vertex cover* of G is a set of vertices in G whose removal from G yields an edgeless graph. The *vertex cover number* of G is the smallest size of a vertex cover of G. A *feedback vertex set* of G is a set of vertices in G whose removal from G yields a forest. The *feedback vertex set number* of G is the smallest size of a feedback vertex set of G. For any $t \in \mathbb{N}$, let C_t denote the cycle on t vertices and K_t denote the complete graph on t vertices.

A *subdivision* of an edge $e = \{u, v\} \in E(G)$ is its replacement by a new degree-2 vertex whose neighbors are u and v. A *subdivision* of G is any graph that can be obtained by subdividing some of the edges of G (where a single edge can be subdivided multiple times). Let $\mathsf{Paths}(G)$ be the set of all (simple) paths in G. Given two (simple) paths P_1 and P_2 that share one or two endpoints and are internally vertex-disjoint, let $P_1 + P_2$ denote the (simple) path or cycle (if both endpoints are shared) obtained by concatenating P_1 and P_2. The *size* of a cycle or path is its number of vertices, and its *length* is its number of edges. A graph H is a *topological minor* of G if G contains a subgraph that is isomorphic to some subdivision of H. More explicitly, this notion is defined as follows.

▶ **Definition 5** (Topological Minor)**.** *A graph H is a* topological minor *of a graph G if there exist injective functions $\phi : V(H) \to V(G)$ and $\varphi : E(H) \to \mathsf{Paths}(G)$ such that for all $e = \{h, h'\} \in E(H)$, the endpoints of $\varphi(e)$ are $\phi(h)$ and $\phi(h')$, for all distinct $e, e' \in E(H)$, the paths $\varphi(e)$ and $\varphi(e')$ are internally vertex-disjoint, and there do not exist a vertex v in the image of ϕ and an edge $e \in E(H)$ such that v is an internal vertex on $\varphi(e)$.*

[1] If G does not contain any cycle, then its girth is defined as ∞.

The Catersian product of two graphs is defined as follows.

▶ **Definition 6** (Cartesian Product of Graphs). *The* Cartesian product $G \times H$ *of two graphs* G *and* H *is the graph whose vertex set is the Cartesian product* $V(G) \times V(H)$*, where any two vertices* (u, u') *and* (v, v') *are adjacent if and only if one of the following holds: (i)* $u = v$ *and* $\{u', v'\} \in E(H)$*; (ii)* $u' = v'$ *and* $\{u, v\} \in E(G)$*.*

Treewidth is a measure of how "treelike" is a graph, formally defined as follows.

▶ **Definition 7** (Treewidth). *A* tree decomposition *of a graph* G *is a pair* (T, β) *of a tree* T *and* $\beta : V(T) \to 2^{V(G)}$*, such that*

1. *for any edge* $\{x, y\} \in E(G)$ *there exists a node* $v \in V(T)$ *such that* $x, y \in \beta(v)$*, and*
2. *for any vertex* $x \in V(G)$*, the subgraph of* T *induced by the set* $T_x = \{v \in V(T) : x \in \beta(v)\}$ *is a non-empty tree.*

The width *of* (T, β) *is* $\max_{v \in V(T)}\{|\beta(v)|\} - 1$*. The* treewidth *of* G*, denoted by* $\mathtt{tw}(G)$*, is the minimum width over all tree decompositions of* G*.*

The treewidth of a graph can be efficiently approximated up to a constant factor as follows.

▶ **Proposition 8** ([7]). *There exists an algorithm that, given a graph* G *and an integer* t*, in time* $2^{\mathcal{O}(t)}n$ *either determines that the treewidth of* G *is larger than* t*, or outputs a tree decomposition of* G *of width at most* $5t + 4$*.*

The following relation between treewidth and feedback vertex set number is folklore.

▶ **Proposition 9** ([12]). *There exists an algorithm that, given a graph* G *with a feedback vertex set* U*, in time* $\mathcal{O}(|U|n)$ *outputs a tree decomposition of* G *of width* $|U|$*.*

3 FPT Algorithm for Long Cycle

In this section, we consider the parameterized complexity of Long Cycle parameterized multiplicatively above girth, and prove Theorems 1 and 2. For our positive result, we required the following definition and propositions. First, we give the definition of a graph called a t-prism, which has $2t$ vertices and $3t$ edges (see Fig. 1).

▶ **Definition 10** (Prism). *For any* $t \in \mathbb{N}$*, the* t-prism *is the Cartesian product* $C_t \times K_2$*.*

The following proposition asserts that a graph of sufficiently high treewidth necessarily contains a large prism as a topological minor.

▶ **Proposition 11** ([3]). *For any* $k \in \mathbb{N}$*, any graph* G *of treewidth at least* $60k^2$ *contains a* k*-prism as a topological minor.*[2]

In case the treewidth of the graph is low, we will be able to solve the problem by making use of the following proposition.

▶ **Proposition 12** ([6]). *There exists an algorithm that, given a graph* G *and a tree decomposition of* G *of width* t*, in time* $2^{\mathcal{O}(t)}n$ *outputs a cycle (if one exists) and a path in* G *of maximum sizes.*

[2] Although the result is mentioned in terms of minors, the proof given in [3] yields the result stated here.

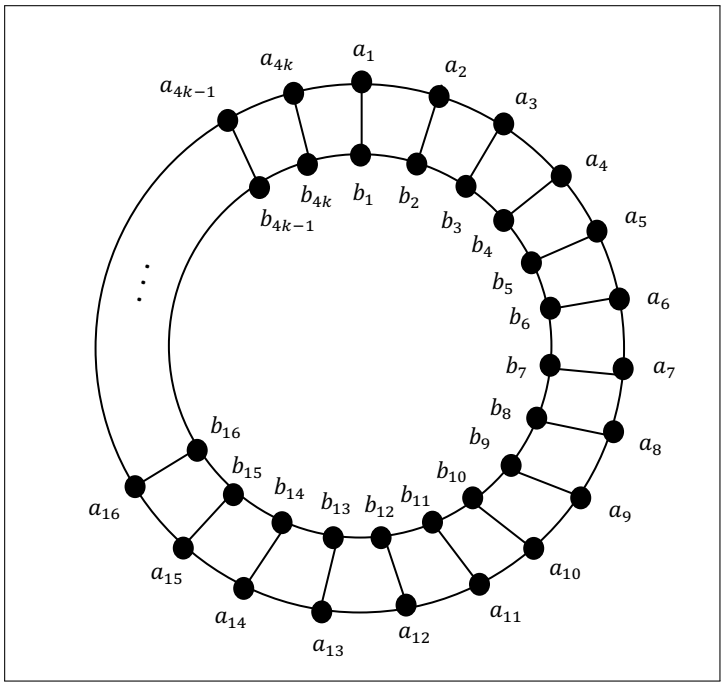


Figure 1 A $4k$-prism.

The main combinatorial lemma we need for our positive result is as follows. This lemma handles the case where the treewidth of the graph is large.

▶ **Lemma 13.** *For any $k \in \mathbb{N}$, any graph G that contains the $4k$-prism as a topological minor also has a cycle on at least $\gamma(G) \cdot k$ vertices.*

Proof. Let H be the $4k$-prism. Notice that the vertex set of H can be denoted by $V(H) = \{a_1, a_2, \ldots, a_{4k}, b_1, b_2, \ldots, b_{4k}\}$ so that $H[\{a_1, a_2, \ldots, a_{4k}\}]$ and $H[\{b_1, b_2, \ldots, b_{4k}\}]$ are cycles and $\{\{a_i, b_i\} : i \in \{1, 2, \ldots, 4k\}\} \subseteq E(H)$ (see Fig. 1). Let $\phi\colon V(H) \to V(G)$ and $\varphi\colon E(H) \to \mathsf{Paths}(G)$ be some two functions that witness that H is a topological minor of G. Let $S_1, \ldots, S_{2k}$ be a set of $2k$ vertex-disjoint cycles of length 4 in H defined as follows: For each $i \in \{0, 1, \ldots, 2k-1\}$, $S_i = H[\{a_{2i+1}, a_{2i+2}, b_{2i+2}, b_{2i+1}\}]$ (see Fig. 1). Additionally, we define two (not vertex-disjoint) cycles C and C' in H as follows: $C = H[\{a_1, a_2, b_2, b_3, a_3, a_4, \ldots, a_{4k-1}, a_{4k}, b_{4k}\}]$ and $C' = H[\{a_1, b_1, b_2, a_2, a_3, b_3, \ldots, b_{4k}, a_{4k}, a_1\}]$ (see Fig. 1).

Now, for each $i \in \{0, 1, \ldots, 2k-1\}$, let $A_i = \varphi(\{a_{2i+1}, a_{2i+2}\}) + \varphi(\{a_{2i+2}, b_{2i+2}\}) + \varphi(\{b_{2i+2}, b_{2i+1}\}) + \varphi(\{b_{2i+1}, a_{2i+1}\})$. This notation is well defined as required to concatenate paths – specifically, here we concatenated internally vertex-disjoint paths sharing exactly one endpoint except for the last concatenation where both endpoints are shared (by the paths $\varphi(\{a_{2i+1}, a_{2i+2}\}) + \varphi(\{a_{2i+2}, b_{2i+2}\}) + \varphi(\{b_{2i+2}, b_{2i+1}\})$ and $\varphi(\{b_{2i+1}, a_{2i+1}\})$). Roughly speaking, A_i is the cycle to which φ maps S_i. Notice that $\{A_i\}_{i=0}^{2k-1}$ is a collection of $2k$ vertex-disjoint cycles in G. Therefore, together they contain at least $2\gamma(G) \cdot k$ edges.

Additionally, let $B = \varphi(\{a_1, a_2\}) + \varphi(\{a_2, b_2\}) + \varphi(\{b_2, b_3\}) + \varphi(\{b_3, a_3\} + \cdots + \varphi(\{a_{4k}, b_{4k}\} + \varphi(\{b_{4k}, a_1\})$. Again, this notation is well defined as required to concatenate paths. Similarly, let $B' = \varphi(\{a_1, b_1\}) + \varphi(\{b_1, b_2\}) + \varphi(\{b_2, a_2\}) + \varphi(\{a_2, a_3\} + \cdots + \varphi(\{b_{4k}, a_{4k}\} + \varphi(\{a_{4k}, a_1\})$. Roughly speaking, B and B' are the cycles to which φ maps C and C', respectively. Notice that $E(H) = E(C) \cup E(C')$. Thus, we deduce that $\bigcup_{i=0}^{2k-1} E(A_i) \subseteq E(C) \cup E(C')$. From this,

we further deduce least one among the cycles B and B' must contain at least half the edges in $\bigcup_{i=0}^{2k-1} E(A_i)$. Because we already showed that $|\bigcup_{i=0}^{2k-1} E(A_i)| \geq 2\gamma(G) \cdot k$, this means that at least one among the cycles B and B' has length (and therefore also size) at least $\gamma(G) \cdot k$. This completes the proof. ◀

By Proposition 11, any graph G of treewidth at least $60 \cdot (4k)^2 = 960k^2$ contains a $4k$-prism as a topological minor. Thus, we derive the following corollary to Lemma 13.

▶ **Corollary 14.** *For any $k \in \mathbb{N}$, any graph G of treewidth at least $960k^2$ has a cycle on at least $\gamma(G) \cdot k$ vertices.*

We are now ready to prove our main theorem.

▶ **Theorem 1.** Long Cycle *(and* Long Path*) parameterized multiplicatively above girth is solvable in time $2^{\mathcal{O}(k^2)}n$.*

Proof. Given an instance (G, k) of Long Cycle (Long Path), the algorithm is executed as follows. First, it calls the algorithm in Proposition 8 with $t = 960k^2$ in time $2^{\mathcal{O}(t)}n = 2^{\mathcal{O}(k^2)}n$ to either determine that the treewidth of G is larger than t, or output a tree decomposition of G of width at most $5t + 4$. In the first case, it concludes that (G, k) is a Yes-instance of Long Cycle (resp. Long Path), which is correct by Corollary 14. In the second case, it calls the algorithm in Proposition 12 to obtain a cycle (resp. path) of maximum size in G in time $2^{\mathcal{O}(5t+4)}n = 2^{\mathcal{O}(k^2)}n$, and concludes that (G, k) is a Yes-instance of Long Cycle (resp. Long Path) if and only if the size of this cycle (resp. path) is at least $\gamma(G) \cdot k$. ◀

▶ **Theorem 2.** *For any fixed constant $\epsilon > 0$,* Long Cycle *(and* Long Path*) parameterized multiplicatively above $g^{1+\epsilon}$ is para-NP-hard, where g is the girth of the input graph.*

Proof. Consider some fixed constant $\epsilon > 0$. Our proof is based on a reduction from the Hamiltonian Cycle (resp. Hamiltonian Path) problem, which is known to be NP-hard [23], to Long Cycle (resp. Long Path) parameterized multiplicatively above $g^{1+\epsilon}$. In the Hamiltonian Cycle (resp. Hamiltonian Path) problem, we are given a graph G and the objective is to decide whether G contains a (simple) cycle (resp. path) on n vertices. In what follows, we only describe the proof for Long Cycle (since the proof for Long Path is similar).

We now describe the reduction. To this end, let G be an instance of Hamiltonian Cycle. Let b be the smallest positive integer such that $b(1 + \epsilon) \in \mathbb{N}$. (Note that b depends only on ϵ.) Let $a = b(1 + \epsilon) - 1$. Notice that $b \leq a$ (since $b \cdot \epsilon$ is a positive integer). Now, subdivide each edge in G $n^a - 1$ times, and let G' be the resulting graph. Let H be the disjoint union of G' and C, where C is a cycle of length n^b. Finally, set $k = 1$, and let (H, k) be the output instance of Long Cycle parameterized multiplicatively above $g^{1+\epsilon}$. Notice that here, g is the girth of H, i.e. $g = \gamma(H)$.

Notice that for any $\ell \in \mathbb{N}$, for any cycle of size ℓ in G, there is a corresponding cycle of size $\ell \cdot n^a$ in G', and vice versa. This implies that any cycle in G' has size at least $3n^a$. Thus, as $b \leq a$, there is a unique shortest cycle in H, which is C. Therefore, $\gamma(H) = n^b$. Then, any cycle (resp. path) C' on at least $(\gamma(H))^{1+\epsilon}k = n^{b(1+\epsilon)} = n^{1+a}$ vertices in H is fully contained in G'. From this, we conclude that for any cycle of size n in G, there is a corresponding cycle of size $(\gamma(H))^{1+\epsilon}k$ in H, and vice versa. This yields the correctness of the reduction. In particular, a parameterized algorithm for Long Cycle parameterized multiplicatively above $g^{1+\epsilon}$ would solve (H, k) is polynomial time (because $k = 1$), and thereby yield a polynomial-time for Hamiltonian Cycle. This completes the proof. ◀

4 FPT Algorithms for (Connected) Vertex Cover and Max Internal Spanning Tree

In this section, we consider the parameterized complexity of (Connected) Vertex Cover and Max Internal Spanning Tree parameterized multiplicatively above girth, and prove Theorem 3. The following proposition asserts that every graph contains either a small feedback vertex set or a large number of vertex-disjoint cycles (with a logarithmic gap), which can also be computed efficiently.

▶ **Proposition 15** ([16, 32]). *For some fixed constant $c \in \mathbb{N}$, there exists an algorithm that, given any $r \in \mathbb{N}$ and a graph G, in time $r^{\mathcal{O}(1)}n$ outputs either a feedback vertex set of G of size at most $cr \log r$ or r vertex-disjoint cycles in G.*

In what follows, we use c to refer to the constant in this proposition. In case the treewidth of the graph is low, we will be able to solve the problem by making use of the following proposition.

▶ **Proposition 16** ([12, 19]). *There exist algorithms for* Vertex Cover, Connected Vertex Cover *and* Max Internal Spanning Tree *that run in time $2^{\mathcal{O}(t)}n$, where t is the treewidth of the input graph.*[3]

The main combinatorial lemmas required for our algorithms are as follows. They handle the case where the feedback vertex set number (and hence also treewidth) of the graph is low. First, for (Connected) Vertex Cover, we need the following lemma.

▶ **Lemma 17.** *For any $r \in \mathbb{N}$, any graph G that contains r vertex-disjoint cycles has vertex cover number at least $\frac{\gamma(G)}{2} \cdot r$.*

Proof. Let $\mathcal{C}$ be a collection of r vertex-disjoint cycles of G. Consider any vertex cover U of G. Then, U must contain at least half the vertices of each cycle in $\mathcal{C}$ in order to contain an endpoint of every edge of that cycle. Since each cycle in $\mathcal{C}$ contains at least $\gamma(G)$ vertices, the lemma follows. ◀

Second, for Max Internal Spanning Tree, we need the following lemma.

▶ **Lemma 18.** *For any $r \in \mathbb{N}$, any* connected *graph G that contains r vertex-disjoint cycles has a spanning tree with at least $(\gamma(G) - 1) \cdot r - 1$ internal vertices.*[4]

Proof. Let $\mathcal{C}$ be a collection of r vertex-disjoint cycles of G. Let H be the graph obtained from G by replacing each cycle $C \in \mathcal{C}$ by a single new vertex v_C (that is made adjacent to all vertices previously adjacent to at least one vertex in C). Let T be some spanning tree of H (whose existence follows from the supposition that G, and hence also H, is connected).

Now, we traverse T in preorder, and when we encounter a new vertex of the form v_C, we perform the following operations. Let p be the parent of v_C in T (if it is not the root), and let u be some vertex in C that is adjacent to p (if v_C is the root, let u be any vertex in C). Then, replace v_C in T by a path P from u to one of its neighbors in C so that this path P contains exactly all the edges of C except one (between u and the chosen neighbor).

[3] Such an algorithm for Max Internal Spanning Tree is not given explicitly, but it is easily seen to follow from the approach of dynamic programming over tree decompositions using representative sets as in [19].

[4] Notice that the supposition that r is positive also implies that G is not a forest and hence $\gamma(G)$ is finite.

Here, make u be the child of p in T, and every child w of v_C is handled as follows: the subtree of w is "hanged" from a vertex of the path P such that w (or some vertex in the cycle represented by w, if w is a new vertex) is adjacent to it in G. Notice that at the end of this process, we obtain a spanning tree of G with the following property. For each cycle $C \in \mathcal{C}$, all vertices except possibly one are internal vertices, with the exception of possibly one cycle (if there was a cycle represented by the root of T) where all vertices except possibly two are internal vertices. Thus, this spanning tree has at least $(\gamma(G) - 1) \cdot r - 1$ internal vertices. This completes the proof. ◀

We are now ready to prove our main theorem.

▶ **Theorem 3.** Vertex Cover, Connected Vertex Cover *and* Max Internal Spanning Tree *parameterized multiplicatively above girth are solvable in time* $2^{\mathcal{O}(k \log k)}n$.

Proof. Given an instance (G, k) of (Connected) Vertex Cover (resp. Max Internal Spanning Tree), the algorithm is executed as follows. Without loss of generality, we suppose that G is connected in the case of Max Internal Spanning Tree, else we clearly have a No-instance of this problem. First, the algorithm calls the algorithm in Proposition 15 with $r = 2k + 1$ in time $r^{\mathcal{O}(1)}n = k^{\mathcal{O}(1)}n$ to obtain either a feedback vertex set of G of size at most $cr \log r$ or $r > 2k$ vertex-disjoint cycles in G. In the second case, by Lemma 17 G has vertex cover number strictly larger than $\frac{\gamma(G)}{2} \cdot 2k = \gamma(G) \cdot k$, and by Lemma 18, it also has a spanning tree with at least $(\gamma(G) - 1) \cdot 2k - 1 \geq \gamma(G) \cdot k$ internal vertices (where the last inequality follows from the facts that $\gamma(G) \geq 3$ and $k \in \mathbb{N}$). Thus, for Vertex Cover and Connected Vertex Cover the algorithm correctly determines that the answer is No, and for Max Internal Spanning Tree the algorithm correctly determines that the answer is Yes. In the first case, the algorithm calls the algorithm in Proposition 9 in time $\mathcal{O}(k \log k \cdot n)$ to obtain a tree decomposition of G of width at most $cr \log r = \mathcal{O}(k \log k)$. Then, it calls the algorithm in Proposition 12 in time $2^{\mathcal{O}(k \log k)}n$ to obtain a (connected) vertex cover of G of minimum size (resp. a spanning tree of G of maximum number of internal vertices), and concludes that (G, k) is a Yes-instance of (Connected) Vertex Cover (resp. Max Internal Spanning Tree) if and only if the output (connected) vertex cover has size at most $\gamma(G) \cdot k$ (resp. the output spanning tree has at least $\gamma(G) \cdot k$ internal vertices). ◀

5 Hardness for Feedback Vertex Set and Cycle Packing

Lastly, we prove the correctness of Theorem 4.

▶ **Theorem 4.** Feedback Vertex Set *and* Cycle Packing *parameterized additively above girth are para-NP-hard.*

Proof. We only prove the lemma for Feedback Vertex Set since the proof for Cycle Packing follows from symmetric arguments. For this purpose, we give a reduction from Feedback Vertex Set itself, which is an NP-hard problem [23]. Towards this, let (G, k) be an instance of Feedback Vertex Set. Then, obtain H from G by subdividing each edge of G k times, and adding a new cycle of size k. Clearly, G has a feedback vertex set of size at most k if and only if H has a feedback vertex set of size at most $k + 1$ (the extra 1 is required to hit the new cycle). Moreover, the girth of H is exactly k. Thus, an algorithm for Feedback Vertex Set parameterized additively above girth should solve $(H, 1)$ in polynomial time (since the parameter is 1), thereby determining whether the feedback vertex set number of H is at most k, and consequently solving (G, k). ◀

References

1. Noga Alon, Gregory Gutin, Eun Jung Kim, Stefan Szeider, and Anders Yeo. Solving MAX-r-SAT Above a Tight Lower Bound. In *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 511–517. SIAM, 2010.
2. Ivona Bezáková, Radu Curticapean, Holger Dell, and Fedor V. Fomin. Finding Detours is Fixed-Parameter Tractable. In *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017*, volume 80 of *LIPIcs*, pages 54:1–54:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017.
3. E. Birmelé, J. A. Bondy, and B. A. Reed. Brambles, Prisms and Grids. In Adrian Bondy, Jean Fonlupt, Jean-Luc Fouquet, Jean-Claude Fournier, and Jorge L. Ramírez Alfonsín, editors, *Graph Theory in Paris: Proceedings of a Conference in Memory of Claude Berge*, pages 37–44. Birkhäuser Basel, Basel, 2007.
4. Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. Narrow sieves for parameterized paths and packings. *CoRR*, abs/1007.1161, 2010. `arXiv:1007.1161`.
5. Hans L. Bodlaender. On Linear Time Minor Tests with Depth-First Search. *J. Algorithms*, 14(1):1–23, 1993. `doi:10.1006/jagm.1993.1001`.
6. Hans L. Bodlaender, Marek Cygan, Stefan Kratsch, and Jesper Nederlof. Deterministic single exponential time algorithms for connectivity problems parameterized by treewidth. *Inf. Comput.*, 243:86–111, 2015. `doi:10.1016/j.ic.2014.12.008`.
7. Hans L. Bodlaender, Pål Grønås Drange, Markus S. Dregi, Fedor V. Fomin, Daniel Lokshtanov, and Michal Pilipczuk. A $c^k n$ 5-Approximation Algorithm for Treewidth. *SIAM J. Comput.*, 45(2):317–378, 2016. `doi:10.1137/130947374`.
8. Jianer Chen, Joachim Kneis, Songjian Lu, Daniel Mölle, Stefan Richter, Peter Rossmanith, Sing-Hoi Sze, and Fenghui Zhang. Randomized divide-and-conquer: improved path, matching, and packing algorithms. *SIAM J. Comput.*, 38(6):2526–2547, 2009. `doi:10.1137/080716475`.
9. Jianer Chen, Songjian Lu, Sing-Hoi Sze, and Fenghui Zhang. Improved algorithms for path, matching, and packing problems. In *Proceedings of the18th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 298–307. SIAM, 2007.
10. Robert Crowston, Michael R. Fellows, Gregory Z. Gutin, Mark Jones, Eun Jung Kim, Fran Rosamond, Imre Z. Ruzsa, Stéphan Thomassé, and Anders Yeo. Satisfying more than half of a system of linear equations over GF(2): A multivariate approach. *J. Comput. Syst. Sci.*, 80(4):687–696, 2014.
11. Robert Crowston, Mark Jones, Gabriele Muciaccia, Geevarghese Philip, Ashutosh Rai, and Saket Saurabh. Polynomial Kernels for lambda-extendible Properties Parameterized Above the Poljak-Turzik Bound. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, volume 24 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 43–54, Dagstuhl, Germany, 2013. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.
12. Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. `doi:10.1007/978-3-319-21275-3`.
13. Reinhard Diestel. *Graph theory*, volume 173 of *Graduate Texts in Mathematics*. Springer-Verlag, Berlin, third edition, 2005.
14. Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013. `doi:10.1007/978-1-4471-5559-1`.
15. P. Erdős and T. Gallai. On maximal paths and circuits of graphs. *Acta Math. Acad. Sci. Hungar*, 10:337–356 (unbound insert), 1959.
16. P Erdős and L Pósa. On independent circuits contained in a graph. *Canad. J. Math.*, 17:347–352, 1965.

17 Fedor V. Fomin, Petr A. Golovach, Daniel Lokshtanov, Fahad Panolan, Saket Saurabh, and Meirav Zehavi. Going Far From Degeneracy. In Michael A. Bender, Ola Svensson, and Grzegorz Herman, editors, *27th Annual European Symposium on Algorithms (ESA 2019)*, volume 144 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 47:1–47:14, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.ESA.2019.47.

18 Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, and Saket Saurabh. Efficient Computation of Representative Families with Applications in Parameterized and Exact Algorithms. *J. ACM*, 63(4):29:1–29:60, 2016. doi:10.1145/2886094.

19 Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, and Saket Saurabh. Efficient Computation of Representative Families with Applications in Parameterized and Exact Algorithms. *J. ACM*, 63(4):29:1–29:60, 2016.

20 Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, Saket Saurabh, and Meirav Zehavi. Long directed (s, t)-path: FPT algorithm. *Inf. Process. Lett.*, 140:8–12, 2018.

21 F.V. Fomin, D. Lokshtanov, S. Saurabh, and M. Zehavi. *Kernelization: Theory of Parameterized Preprocessing*. Cambridge University Press, 2018.

22 Harold N. Gabow and Shuxin Nie. Finding a long directed cycle. *ACM Transactions on Algorithms*, 4(1), 2008. doi:10.1145/1328911.1328918.

23 M. R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

24 Gregory Gutin, Eun Jung Kim, Michael Lampis, and Valia Mitsou. Vertex Cover Problem Parameterized Above and Below Tight Bounds. *Theory of Computing Systems*, 48(2):402–410, 2011. doi:10.1007/s00224-010-9262-y.

25 Gregory Gutin, Leo van Iersel, Matthias Mnich, and Anders Yeo. Every ternary permutation constraint satisfaction problem parameterized above average has a kernel with a quadratic number of variables. *J. Computer and System Sciences*, 78(1):151–163, 2012. doi:10.1016/j.jcss.2011.01.004.

26 Gregory Z. Gutin and Viresh Patel. Parameterized Traveling Salesman Problem: Beating the Average. *SIAM J. Discrete Math.*, 30(1):220–238, 2016.

27 Gregory Z. Gutin, Arash Rafiey, Stefan Szeider, and Anders Yeo. The Linear Arrangement Problem Parameterized Above Guaranteed Value. *Theory Comput. Syst.*, 41(3):521–538, 2007.

28 Gregory Z. Gutin, Stefan Szeider, and Anders Yeo. Fixed-Parameter Complexity of Minimum Profile Problems. *Algorithmica*, 52(2):133–152, 2008.

29 Falk Hüffner, Sebastian Wernicke, and Thomas Zichner. Algorithm Engineering for Color-Coding with Applications to Signaling Pathway Detection. *Algorithmica*, 52(2):114–132, 2008. doi:10.1007/s00453-007-9008-7.

30 Joachim Kneis, Daniel Mölle, Stefan Richter, and Peter Rossmanith. Divide-and-Color. In *Proceedings of the 34th International Workshop Graph-Theoretic Concepts in Computer Science (WG)*, volume 4271 of *Lecture Notes in Computer Science*, pages 58–67. Springer, 2008.

31 Ioannis Koutis. Faster Algebraic Algorithms for Path and Packing Problems. In *Proceedings of the 35th International Colloquium on Automata, Languages and Programming (ICALP)*, volume 5125 of *Lecture Notes in Comput. Sci.*, pages 575–586. Springer, 2008.

32 Daniel Lokshtanov, Amer E. Mouawad, Saket Saurabh, and Meirav Zehavi. Packing Cycles Faster Than Erdos-Posa. In *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, pages 71:1–71:15, 2017.

33 Meena Mahajan and Venkatesh Raman. Parameterizing above Guaranteed Values: MaxSat and MaxCut. *J. Algorithms*, 31(2):335–354, 1999.

34 Meena Mahajan, Venkatesh Raman, and Somnath Sikdar. Parameterizing above or below guaranteed values. *J. Computer and System Sciences*, 75(2):137–153, 2009. doi:10.1016/j.jcss.2008.08.004.

35 B. Monien. How to find long paths efficiently. In *Analysis and design of algorithms for combinatorial problems (Udine, 1982)*, volume 109 of *North-Holland Math. Stud.*, pages 239–254. North-Holland, Amsterdam, 1985. doi:10.1016/S0304-0208(08)73110-4.

36 Maria J. Serna and Dimitrios M. Thilikos. Parameterized Complexity for Graph Layout Problems. *Bulletin of the EATCS*, 86:41–65, 2005.

37 Dekel Tsur. Faster deterministic parameterized algorithm for k-Path. *CoRR*, abs/1808.04185, 2018. arXiv:1808.04185.

38 Ryan Williams. Finding Paths of Length k in $O^*(2^k)$ Time. *Inf. Process. Lett.*, 109(6):315–318, 2009.

39 Meirav Zehavi. Mixing Color Coding-Related Techniques. In *ESA 2015*, volume 9294 of *Lecture Notes in Computer Science*, pages 1037–1049. Springer, 2015.

40 Meirav Zehavi. A randomized algorithm for long directed cycle. *Inf. Process. Lett.*, 116(6):419–422, 2016. doi:10.1016/j.ipl.2016.02.005.

Ad Hoc Multi-Input Functional Encryption

Shweta Agrawal
Dept. of Computer Science and Engineering, IIT Madras, Chennai, India
shweta@iitm.ac.in

Michael Clear
Dept. of Computer Science, Georgetown University, Washington, DC, USA
clrmic2@gmail.com

Ophir Frieder
Dept. of Computer Science, Georgetown University, Washington, DC, USA
ophir@ir.cs.georgetown.edu

Sanjam Garg
Dept. of EECS, University of California at Berkeley, CA, USA
sanjamg@berkeley.edu

Adam O'Neill
Dept. of Computer Science, University of Massachusetts, Amherst, MA, USA
amoneill@gmail.com

Justin Thaler
Dept. of Computer Science, Georgetown University, Washington, DC, USA
Justin.Thaler@georgetown.edu

Abstract

Consider *sources* that supply sensitive data to an *aggregator*. Standard encryption only hides the data from eavesdroppers, but using specialized encryption one can hope to hide the data (to the extent possible) from the aggregator itself. For flexibility and security, we envision schemes that allow sources to supply encrypted data, such that at any point a dynamically-chosen subset of sources can allow an agreed-upon joint function of their data to be computed by the aggregator. A primitive called multi-input functional encryption (MIFE), due to Goldwasser et al. (EUROCRYPT 2014), comes close, but has two main limitations:

- it requires trust in a third party, who is able to decrypt all the data, and
- it requires function arity to be fixed at setup time and to be equal to the number of parties.

To drop these limitations, we introduce a new notion of *ad hoc* MIFE. In our setting, each source generates its own public key and issues individual, function-specific secret keys to an aggregator. For successful decryption, an aggregator must obtain a separate key from each source whose ciphertext is being computed upon. The aggregator could obtain multiple such secret-keys from a user corresponding to functions of varying arity. For this primitive, we obtain the following results:

- We show that standard MIFE for general functions can be bootstrapped to ad hoc MIFE for *free*, i.e. without making any additional assumption.
- We provide a direct construction of ad hoc MIFE for the inner product functionality based on the Learning with Errors (LWE) assumption. This yields the first construction of this natural primitive based on a standard assumption.

At a technical level, our results are obtained by combining standard MIFE schemes and *two-round* secure multiparty computation (MPC) protocols in novel ways highlighting an interesting interplay between MIFE and two-round MPC.

2012 ACM Subject Classification Security and privacy → Cryptography; Security and privacy → Public key (asymmetric) techniques; Security and privacy → Mathematical foundations of cryptography

Keywords and phrases Multi-Input Functional Encryption

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.40

Related Version A full version of the paper is available at `https://eprint.iacr.org/2019/356`.

Funding *Sanjam Garg*: Research supported in part from DARPA/ARL SAFEWARE Award W911NF15C0210, AFOSR Award FA9550- 15-1-0274, AFOSR YIP Award, DARPA and SPAWAR under contract N66001-15-C-4065, a Hellman Award and research grants by the Okawa Foundation, Visa Inc., and Center for Long-Term Cybersecurity (CLTC, UC Berkeley). The views expressed are those of the author and do not reflect the official policy or position of the funding agencies.

Acknowledgements We thank Romain Gay for pointing out a subtlety in the security definition of (standard) MIFE for inner products, which necessitated a minor adjustment in our ad hoc MIFE construction.

1 Introduction

In modern society, there is an inherent need for external entities to aggregate and analyze sensitive data from a variety of sources. A few prominent examples are:

- To track diseases, disease control centers would like hospital patients' medical information.
- To determine medication efficacy for a given subpopulation, pharmaceutical companies would like patients' genomic information.
- To provide targeted advertising to consumers, corporations would like buyers' demographic information.

However, this release of sensitive data to external entities is unsettling, as these entities must now be trusted to preserve the confidentiality of the released data. We would like to avoid the need for this trust and believe that specialized encryption schemes will be an important tool for doing so. At a high level, we would like schemes that allow users to encrypt their data before transferring them to an external entity, such that only certain user-specified joint functions of the data are revealed to the entity holding it. We would like this security guarantee to be supported in a flexible way, allowing joint functions of the data to be revealed by any dynamically-chosen subset of users that permit it.

A primitive that comes close, due to Goldwasser et al. [45], is *multi-input functional encryption* (MIFE). To understand MIFE, we first recall the simpler notion of functional encryption (FE) [20]. Just as in traditional encryption, in functional encryption ciphertexts can be generated with an encryption key. However, each *decryption* key is associated with a function f, and decryption of an encryption of m using this key results in not m but $f(m)$. Intuitively, security requires that nothing more than $f(m)$ can be learned from the encryption of m and the decryption key for f. In MIFE, decryption keys allow computing *joint* functions of (possibly) different plaintexts underlying multiple ciphertexts. That is, decryption takes a key for a function f and ciphertexts $c_1, \ldots c_n$ encrypting $m_1, \ldots, m_n$, and outputs $f(m_1, \ldots, m_n)$.

However, MIFE has an important drawback: encryption and decryption keys are generated via a global setup procedure run by an external entity usually called the *key authority*. This begs the question of whether putting trust in the key authority is really better than putting trust in the external entities that aggregate and analyze the data in the first place. A similar point was made by Rogaway about identity-based encryption (IBE) [60]. Indeed, removing this in the case of IBE (and other settings) has been an active area of investigation, *e.g.* see [18, 49, 35]. We contend that for MIFE (and indeed FE) the concern is heightened, as the authority can not only decrypt all the data but is also the one in charge of which functions of the data other external entities can compute. Hence, MIFE does not allow users to enforce their own privacy policies.

Additionally, from a flexibility standpoint, MIFE is limited in that it fixes the number of senders and function arity at setup time. This does not support a dynamic setting in which users can join or leave. Progress on removing this limitation was made by Badrinarayanan et al. [10], who introduced a notion of MIFE for unbounded arity functions. However, their notion does not allow any subset of users to reveal a joint function of their data to an external entity without coordination from all other users, and moreover relies on strong "knowledge type" assumptions. Another closely related notion to ours is that of *decentralized multi-client FE*, recently introduced by Chotard et al. [27]. While this work shares some of the high level motivation of the present work, the two primitives are very different. Please see Section 1.3.3 for a comparison.

1.1 Our Notion: *Ad Hoc* MIFE

To address the above limitations, we introduce a new notion of *ad hoc* MIFE. In ad hoc MIFE, each source (aka. sender or user) will run a *local* setup procedure to generate some public parameters as well as a private encryption key. (One can also consider a public-key setting, but this puts limits on achievable security and we do not do so in this work.[1]) Each source publishes their public parameters and encrypts using their private key. These ciphertexts can be sent to an aggregator (aka. decryptor). Furthermore, using their private keys, sources can issue "partial decryption keys" to an aggregator. Each partial decryption key is associated with an ℓ-ary function f for some ℓ and is generated using the public keys of $\ell - 1$ other (dynamically chosen) sources. If these other $\ell - 1$ sources also issue "matching" partial decryption keys for f to this aggregator, it can decrypt any ℓ ciphertexts $c_1, \ldots, c_\ell$, each produced by the corresponding source, to $f(m_1, \ldots, m_\ell)$ where $m_1, \ldots, m_\ell$ are the plaintexts. One can also consider restricted versions of the above notion, that bound the number of users or fix ℓ (or both). In particular, taking the number of users equal to ℓ gives a version of MIFE that still drops the global setup procedure but lacks the dynamic aspect. Finally, we also consider the restricted notion of *bounded* ad hoc MIFE, where we place a bound on the number of "partial decryption keys" a user is allowed to issue. Intuitively, for security, we require that an aggregator learns only the functions of the data for which it has been given all of the matching partial decryption keys.

Note that while we assume each source can obtain the authentic public parameters of other sources with whom it wants to allow joint functions of the data to be computed, there is no other prior coordination between users (this is one of the main advantages of our notion over [27]). In particular, there is no external entity that generates public parameters or keys. In some of our constructions, we work in the common reference string (CRS) model, but note that this is still much weaker than having an authority who can decrypt all the data.

1.2 Our Results

Our results may be summarized as follows:

- *Feasibility result for general functions*: First, we show that standard MIFE for general functions can be bootstrapped to ad hoc MIFE for general functions for *free*. More specifically, we show that ad hoc MIFE for any functionality is implied by standard MIFE for that functionality combined in a novel way with general FE and a special type of *two-round* secure multiparty computation (MPC) protocol. The latter two are implied by standard MIFE for general functions.

[1] In more detail, in the public-key setting a decryptor could launch an attack where it replaces one user's input with various values to determine information about the input of another user.

Table 1 Our new constructions of ad hoc MIFE.

Functionality	Assumptions	Security	CRS?	Section
General	std. MIFE	Semi-honest	No	4
General	std. MIFE	Malicious	Yes	4
Inner Product	LWE	Semi-honest	Yes	5
Inner Product	LWE	Malicious	Yes	5
Inner Product (Bounded)	DDH, LWE, DCR	Semi-honest	No	5
Inner Product (Bounded)	DDH, LWE, DCR	Malicious	Yes	5

While very general, the result leaves open the goal of obtaining ad hoc MIFE under standard assumptions. In general, this is challenging as standard MIFE is already known to be equivalent to indistinguishability obfuscation [33, 8, 17], which is a central open problem in cryptography. In fact, some negative evidence about the hardness of obtaining such constructions has also been provided [37, 38]. Thus, with the goal of moving towards using standard assumptions, we consider the task of ad hoc MIFE for special but natural functionalities.

- *Constructions for Inner Products from Standard Assumptions*: We provide a construction of ad hoc MIFE for the *inner product* functionality from standard assumptions, namely LWE.[2] Introduced by Abdalla et al. [1] in the single-input setting, this functionality has applications in data mining and information retrieval. Our result is obtained via a general paradigm for constructing ad-hoc MIFE schemes from standard MIFE schemes satisfying certain natural properties; or, what we call "ad hoc friendly" standard MIFE schemes. We show that certain constructions of standard MIFE scheme for inner products from the literature [3, 2] based on standard assumptions (DDH, LWE, or DCR) already satisfy these properties. Additionally, we use a specific two-round MPC protocol [57] that can also be obtained via LWE. We note that by using a two-round MPC protocol from any two-round OT protocol [41, 42, 15] here, we also obtain results for the case of *bounded* ad hoc MIFE for inner products – namely, we get bounded ad hoc MIFE for inner products from DDH, LWE and DCR as well.[3] We remark that since our general construction (first result) already relies on general MIFE for circuits, there is no advantage to mitigating assumptions for the two-round MPC protocol in that setting.

  We emphasize that our transformation is general. Thus, our transformation can be used to upgrade the security of any "ad hoc friendly" standard MIFE for a given functionality to ad hoc MIFE for the same functionality. This result might also be useful in obtaining future constructions of ad hoc MIFE. Furthermore, the modularity of this approach allows for simplifications in our constructions.

We tabulate our results in Table 1 and provide explanation of which MPC protocol is needed for each of the results in Section 1.3.

[2] We stress that this functionality outputs inner products in the clear and is therefore a different type of functionality than that of Katz et al. in [54], which tests if an inner product is zero or not.
[3] Note that semi-honest constructions of two-round OT are known under each of these assumptions.

1.3 Technical Overview

In this section, we describe at a high level the challenges involved in constructing ad hoc MIFE and our techniques for overcoming them.

1.3.1 Ad Hoc MIFE for Arbitrary Functions.

Standard MIFE and ad hoc MIFE can be seen as secure multiparty computation (MPC) protocols with a particular allowable interaction pattern and certain additional reuse capabilities.[4] To begin, let us consider the interaction pattern followed by standard MIFE. In standard MIFE, there is a *trusted* global setup which receives as input the number of parties ℓ, and outputs a public key and a set of ℓ encryption keys. Additionally, global setup on input a function f generates the decryption key DK_f. Of these, the public parameters are broadcast to all users and encryption key EK_i is provided to encryptor i, for $i \in [\ell]$. The encryptors then compute their ciphertexts $\mathsf{CT}(m_i)$ and send these to the aggregator who may now compute the function output $f(m_1 \ldots, m_\ell)$ using the decryption/function key DK_f. Finally, the system supports arbitrary number of decryption keys and ciphertexts. As explained in Section 1.1, in ad hoc MIFE, we seek to eliminate the trusted global procedure as well as support dynamic choice of parties involved in any function computation.

1.3.1.1 Challenges Involved

An approach to eliminating the trusted setup from standard MIFE is to use MPC to replace the global setup. However, naively computing the setup procedure using MPC would introduce interaction between the parties, which the syntax of MIFE does not allow. Moreover, this (interactive) procedure would need to be rerun each time a function key is required to be generated. Using two round MPC, one may hope to overcome the barrier of interaction using the following natural idea: let parties perform a two-round MPC to perform the setup and key generation for a standard MIFE. In more detail, parties in the first round could publish as their public parameters the first round MPC messages with their secret randomness as input. Given the first round messages, parties could send the second round MPC message to the aggregator, who could use it to compute the function key. However, this approach does not suffice since:

1. MIFE requires that the public parameters be published only once whereas the above template requires publishing fresh public parameters for each function key.
2. Even more fundamentally, the above approach precludes users from being able to encrypt, as their encryption keys are not available given just the first round MPC message.

In particular, the above approach does not decouple ciphertexts and functions as in traditional MIFE, which leads to the limitation that an evaluator cannot evaluate the same function on multiple inputs chosen by the parties, nor evaluate other functions on the same set of inputs. Additionally, the problem is made challenging by the fact that in MIFE an aggregator might obtain arbitrary number of secret keys and an encryptor might generate arbitrary number of ciphertexts.

[4] Recall that MPC allows a set of parties to compute a joint function of their inputs without revealing anything else but in general allows these parties to freely interact (although restrictions may apply in special cases).

1.3.1.2 Overcoming the first barrier: Function re-runnable two-round MPC

In order to mitigate the first problem above, we require that the first round MPC message be sent only once, and reused for all subsequent second round messages thus providing re-usability/re-runnability for secret key generation. Towards achieving this re-usability, an idea is to use function rerunnable two-round MPC protocols, where the same first round message can be reused for multiple functions in the second round. As we will see, certain existing two-round MPC protocols satisfy this requirement (see later), but this still does not solve the problem. This is because in adhoc MIFE, we additionally need that for the MPC protocol, the function or even its arity are not known at the time the first round message is sent. We overcome this hurdle by using "function delayed" protocols, which permit the choice of function to be delayed to the second round of the protocol. Together, these special protocols may be used to overcome the first barrier outlined above.

1.3.1.3 Overcoming the second barrier: Delaying Encryption

In order to overcome the second barrier, we allow the encryptor to delay the encryption process until the encryption keys are known (in similar spirit as [13, 30]). In more detail, we will have each source independently run the setup algorithm of a single input FE scheme, denoted as FE and compute the first round message of an MPC protocol using the FE master key as input. This message is published as part of the public key and made available to all other sources. Additionally, each source provides an encryption of its input m_i using the algorithm $\mathsf{FE.Encrypt}$.

The sources may choose the function f to be computed and the group that will participate in the computation dynamically. At this point, each source independently executes the partial key generation algorithm as follows: it generates the second round message of an MPC protocol for a suitable f dependent functionality $\mathsf{GenKeys}_f$ and sends this to the aggregator. Intuitively, the functionality $\mathsf{GenKeys}_f$ has the circuit f hard-coded in it, and enables the aggregator to compute the output.

However, recall that the inputs to the $\mathsf{GenKeys}$ functionality are not the messages on which the computation must be performed, but rather the FE master keys generated independently by each player. To proceed, the functionality instead uses the FE master keys to compute FE *function* keys for a re-encryption procedure, which *translates* FE ciphertexts to MIFE ciphertexts for a freshly generated (standard) MIFE scheme. During this time, the arity of the function f is known, so a suitable standard MIFE scheme may be instantiated. It further outputs an MIFE function key for the function f.

We are almost done: $\mathsf{GenKeys}_f$ runs the setup procedure for a suitable fixed arity standard MIFE scheme, computes FE function keys for each party for the re-encryption functionality, computes the MIFE function key for f and outputs these. The aggregator uses the FE keys together with the FE ciphertexts provided by each encryptor to translate $\mathsf{FE.Enc}(m_i)$ to $c_i = \mathsf{MIFE.Enc}(m_i)$ and then runs the MIFE decryption procedure to obtain $f(m_1, \ldots, m_\ell)$.

Put together, we resolved all difficulties by carefully nesting a a multi-input FE scheme MIFE, within the single input FE scheme FE, which in turn is nested within a re-runnable two-round MPC protocol MPC.

One final problem remains: the adversary could get some partial information from an "incomplete" set of partial decryption keys for some function. This is because standard MPC makes no guarantee when an honest party does not send their final message. We solve this problem by masking the output of MPC by pseudorandom values generated for each party. The partial decryption keys contains the respective user's pseudorandom masks so that only a complete set of user keys can be used to unmask the output.

While security appears to follow intuitively from the security of MPC, FE and MIFE, the proof must contend with several technical hurdles as we are forced to deal with indistinguishability style security of FE, MIFE (simulation security for these primitives is known to be impossible [20, 6]). We argue security via a careful sequence of hybrids, please see Section 4 for details.

1.3.1.4 Instantiating MPC

We now discuss possible instantiations of MPC to fit the above template. Depending on the properties of the underlying two-round MPC protocol, we obtain different properties of the resulting ad hoc MIFE scheme. In both the semi-honest (passive decryptor) and the malicious (active decryptor) settings, the most general function-rerunnable, two-round MPC protocols without CRS can be constructed [32, 48] from indistinguishability obfuscation [33], which itself can be constructed from multi-input functional encryption [8]. Furthermore, as already noted in [40], we remark that in the semi-honest setting the construction of [32] can actually be instantiated in the plain model. This is based on the observation that the CRS in the protocol of [32] was only needed for the computation in the second round. Thus semi-honest parties could obtain a CRS by just performing a one-round coin flipping in the first round. This yields our first result: we get ad hoc MIFE for general functions from standard MIFE for general functions. In the semi-honest setting, the ad hoc MIFE construction is in the plain model. On the other hand, in the malicious setting, these protocols work in the common reference string (CRS) model.

Alternatively, function-rerunnable two-round MPC in the common reference string (CRS) model can be constructed [28, 57, 24, 59] from learning-with-errors (LWE). This yields ad hoc MIFE from LWE and standard MIFE in the CRS model (either semi-honest or malicious). While bounded two-round MPC in the CRS model can be constructed from bilinear maps [41] and even two-round oblivious transfer [15, 42, 39] or information theoretically [9, 36], these constructions are *not* function-rerunnable so do not suffice for our general construction. We note that these constructions would suffice for obtaining bounded ad hoc MIFE, where a user issues only a bounded number of partial decryption keys and maintains *state* across key issues. However, since in our general result we anyway require the minimum assumption of FE/MIFE, instantiating MPC from weaker assumptions does not yield any benefits, and we do not discuss this further. Our results are highlighted in Table 1.

1.3.2 Ad Hoc MIFE for Inner Products.

While our construction of ad hoc MIFE above applies to arbitrary functionalities, it requires use of standard MIFE for general functions. Unfortunately, as noted above, standard MIFE implies indistinguishability obfuscation. Hence, there is limited hope of basing it on standard assumptions. Additionally, our general transformation uses an FE scheme for a potentially complicated re-encryption functionality and also requires general-purpose, function-rerunnable two-round MPC for computing a complex functionality. These aspects limit the practical applicability of our general result.

Next, we describe a paradigm for constructing ad-hoc MIFE schemes from standard MIFE schemes that are "ad hoc friendly" and a (hopefully simple) two-round MPC protocol. This paradigm significantly simplifies our general construction and provides a way for basing it on standard assumptions. We then show that the standard MIFE scheme for inner products [3, 2], which may be based on DDH, LWE or DCR, is ad hoc friendly and the corresponding two-round MPC protocol is only required to compute inner-products, thus obtaining an efficient ad hoc MIFE scheme for inner products.

More formally, in the inner product functionality a decryption key corresponds to a concatenated vector $\mathbf{y} = (\mathbf{y_1} \parallel \cdots \parallel \mathbf{y_n})$ where $\mathbf{y_i} \in \mathbb{Z}_q^m$, and a ciphertext encrypts a vector $\mathbf{x_i} \in \mathbb{Z}_q^m$. The desired result of decryption is $\sum_{i=1}^{n} \langle \mathbf{x_i}, \mathbf{y_i} \rangle$. Importantly, the decryptor should not learn the partial sums $\langle \mathbf{x_i}, \mathbf{y_i} \rangle$. The inner product functionality has applications in data mining and information retrieval [1, 3]. We use constructions of standard MIFE for inner product by [3, 2].

Below, we start by summarizing our notion of "ad hoc freindliness," which (as we see later) is indeed satisfied by the above mentioned standard MIFE for inner product by [3, 2]. Our notion of ad hoc friendliness may be summarized as follows:

1. *Decentralized Setup.* The $\mathsf{MIFE.Setup}$ algorithm of the MIFE is decentralized in the sense that:
   a. The encryption keys EK_i for $i \in [n]$ corresponding to party i may be generated *independently* of the encryption keys of the remaining parties $[n] \setminus i$.
   b. The master secret key MSK can be decomposed into n components $\{\mathsf{MSK}_i\}_{i \in [n]}$. The partial MSK_i corresponding to party i may be generated locally by party i, without any interaction or shared state with the remaining parties.
2. *Local Encryption.* The encryption algorithm only takes its encryption key and message as input and does not depend on the number of parties or their public parameters.
3. *Piecewise Master Secret Key.* The master secret in standard MIFE $\mathsf{MSK} = \{\mathsf{MSK}_1, \ldots, \mathsf{MSK}_n\}$, if restricted to some subset $S \subseteq [n]$ with $|S| = \ell$, has the same distribution as a master secret generated for functions of arity ℓ.

We show that a standard MIFE with the above properties can be upgraded to ad hoc MIFE described above in a more direct manner than our generic transformation from standard MIFE to ad hoc MIFE. To see this, recall that one of the key challenges in ad hoc MIFE is that the encryptor must encrypt her messages without knowing the encryption key for the underlying standard MIFE. This is because the members or size of the group that will participate in the computation are chosen dynamically later.

To handle this, we used single input FE to encrypt messages and the MPC protocol for functionality $\mathsf{GenKeys}$ to sample an MIFE scheme and then translate the FE ciphertexts to MIFE ciphertexts. In the current setting however, due to properties (1) and (2) above, the encryption key of each party can be generated locally and each party can directly perform MIFE encryption locally. Since the re-encryption functionality involves computing a PRF and computing an MIFE encryption, the savings accrued by skipping this step are significant.

We will still require MPC to compute the MIFE function key, but no longer need the MPC functionality to sample the master secret key, so for simple functionalities this protocol may be much leaner than our general MPC protocol. For instance, in the case of inner products, we show that the required MPC protocol only needs to support inner product computations.

While the structural requirements described above may seem very strong, as mentioned earlier, we show that these requirements are enjoyed by the MIFE for inner products recently constructed by Abdalla et al. [2] and can be used to instantiate our compiler providing a very simple and efficient ad hoc MIFE for inner products. Please see Section 5 for details.

1.3.2.1 Instantiating MPC

As discussed for the case of our generic construction, we use two-round MPC protocols to obtain ad hoc MIFE for inner products. Specifically, since function-rerunnable two-round MPC in the common reference string (CRS) model can be constructed [28, 57, 24, 59] from learning-with-errors (LWE), we immediately get ad hoc MIFE from LWE. This result can

be upgraded to the malicious setting at the additional cost of NIZKs. On the other hand, construction of two-round MPC from two-round oblivious transfer [15, 42, 39], yields *bounded* ad hoc MIFE for inner products under DDH, LWE or DCR, albeit with the requirement that the sources maintain state across key issues. One nice feature of these schemes is that they work without the need for a CRS in the semi-honest setting and upgrade to the malicious setting can be made just using CRS. Please see Section 5 for details.

Our results are highlighted in Table 1.

1.3.3 Related Work

In this section, we discuss the prior work in this area. Here, we focus on two related primitives, decentralized multi-client FE and and non-interactive MPC. Further related work on FE, MIFE, MPC and multi-authority FE can be found in Appendix A.

1.3.3.1 Decentralized Multi-Client Functional Encryption

Very recently, Chotard et al [27] proposed the notion of decentralized multi-client functional encryption (D-MCFE). While the motivation for the two works is similar in removing the common key authority, our notion of adhoc MIFE is significantly more general in that:

1. MCFE itself is more restricted than MIFE, since only CTs with the *same labels* can be combined. In MIFE there is no such restriction. MIFE for circuits captures MCFE for circuits (by checking for equal labels within the MIFE functionality) but not vice versa.
2. Crucially, the setup algorithm in D-MCFE is a protocol that is run between multiple senders, requiring interaction, whereas our setup algorithm is run independently by each source and is thus *non-interactive*. Note that developing a non-interactive solution is one of the main motivations of this work.
3. The work of Chotard et al [27] only provides a construction for inner products. We provide a general construction as well as one for inner products. Since our model is stronger, our inner product construction is significantly more involved than theirs.
4. Decentralized MCFE lacks the dynamic aspect, which is one of the main contributions of our work. We permit the function arity and participating parties to be chosen dynamically – a feature no other construction supports (to the best of our knowledge).

Non-Interactive MPC

Another related notion is that of *non-interactive MPC* (NI-MPC), where a group of asynchronous parties may evaluate a function over their inputs by sending a single message to an evaluator who computes the output [51]. While they appear superficially similar, we note that the model of ad hoc MIFE is fundamentally different from NI-MPC since, unlike NI-MPC, it separates inputs and functions, i.e. provides ciphertexts and function keys which allows *reusing* an input/ciphertext with many different functions, and a function with many different inputs. On the other hand, NI-MPC does not support function reusability at all, and only a very restricted version of input re-usability, namely where only ciphertexts in the same "session" may be combined. The function arity in NI-MPC is also fixed, unlike ad hoc MIFE.

2 Preliminaries

Due to space constraints, the preliminaries can be found in Appendix B.

3 *Ad hoc* Multi-Input Functional Encryption

We are now ready to define our new notion of *ad hoc* multi-input functional encryption (MIFE). For simplicity, we define ad hoc MIFE in the private-key setting only. We leave the study of ad hoc MIFE in the public-key setting for future work.

3.1 Syntax and Correctness

An *ad hoc multi-input functional encryption scheme* aMIFE for a message space $\{\mathcal{M}_\kappa\}_{\kappa\in\mathbb{N}}$ and a functionality $\{\mathcal{F}_\kappa\}_{\kappa\in\mathbb{N}}$, where for each $\kappa \in \mathbb{N}$, each $f \in \mathcal{F}_\kappa$ is a (description of a) function on $(\mathcal{M}_\kappa)^\ell$ for some ℓ (which may depend on f), is given by a set of algorithms with the following syntax:

- $\mathsf{aMIFE.Setup}(1^\kappa)$: A PPT algorithm taking the security parameter κ, and outputting the master secret key MSK and the public parameters PP.
- $\mathsf{aMIFE.KeyGen}(i, \mathsf{MSK}_i, (\mathsf{PP}_1, \cdots, \mathsf{PP}_\ell), f)$: A PT algorithm taking an index $i \in [\ell]$, a master secret key MSK_i corresponding to PP_i, a set of public parameters $\mathsf{PP}_1, \cdots, \mathsf{PP}_\ell$, a function $f \in \mathcal{F}_\kappa$ of arity ℓ, and outputting a corresponding partial decryption key $\mathsf{PDK}_{i,f}$.
- $\mathsf{aMIFE.Enc}(\mathsf{MSK}, \mathbf{x})$: A PPT algorithm taking a master secret key MSK and a message $\mathbf{x} \in \mathcal{M}_\kappa$, and outputting a ciphertext c.
- $\mathsf{aMIFE.Dec}((\mathsf{PDK}_{1,f}, \ldots, \mathsf{PDK}_{\ell,f}), (c_1, \ldots, c_\ell))$: A PT algorithm taking partial decryption keys $(\mathsf{PDK}_{1,f}, \ldots, \mathsf{PDK}_{\ell,f})$ and ciphertexts $(c_1, \ldots, c_\ell)$, and outputting a string y.

▶ **Definition 1** (Correctness). *We say that* aMIFE *is* correct *if for all* $\kappa \in \mathbb{N}$ *and* $\ell = \mathit{poly}(\kappa)$, *all* $\mathbf{x}_1 \ldots \mathbf{x}_\ell \in \mathcal{M}_\kappa$ *and all* $f \in \mathcal{F}_\kappa$ *of arity* ℓ

$$\Pr\left[y = f(\mathbf{x}_1, \cdots, \mathbf{x}_\ell) \middle| \begin{array}{l} (\mathsf{PP}_i, \mathsf{MSK}_i) \leftarrow\!\!\$\ \mathsf{aMIFE.Setup}(1^\kappa) \quad (\forall i \in [\ell]) \\ c_i \leftarrow\!\!\$\ \mathsf{aMIFE.Enc}(\mathsf{MSK}_i, \mathbf{x}_i) \\ \mathsf{PDK}_{i,f} \leftarrow\!\!\$\ \mathsf{aMIFE.KeyGen}(i, \mathsf{MSK}_i, (\mathsf{PP}_i)_{i\in[\ell]}, f) \\ y \leftarrow \mathsf{aMIFE.Dec}((\mathsf{PDK}_{i,f})_{i\in[\ell]}, (c_i)_{i\in[\ell]}) \end{array} \right] = 1 \ .$$

▶ Remark 2. We highlight two ways that ad hoc MIFE differs from standard MIFE [45]. First, the $\mathsf{aMIFE.Setup}$ algorithm is run per user and does not output all of the $\mathsf{MSK}_1, \ldots \mathsf{MSK}_n$ at once. Second, the total number of users n and the function arity ℓ are not fixed and input to the $\mathsf{aMIFE.Setup}$ algorithm. We also note that for simplicity in our formulation of ad hoc MIFE the public parameters of the parties input to the key generation algorithm are ordered.

▶ Remark 3. We also consider two relaxations of ad hoc MIFE:

- We can allow an additional algorithm CRSGen taking 1^κ and outputting a common reference string CRS that is input to the remaining algorithms. We refer to this as ad hoc MIFE *in the CRS model*. The CRS model is weaker than having a key generation authority who can decrypt all the data.
- We can allow the total number of users 1^n to be input to the setup algorithm. We refer to this as *bounded* ad hoc MIFE. We can additionally require $n = \ell$, which we refer to as *static* (vs. dynamic) ad hoc MIFE. In particular, static ad hoc MIFE recovers a variant of MIFE that is similar to the standard one but still eliminates the trusted key generation authority.

3.2 Indistinguishability-Based Security

We first present an indistinguishability-based security notion. We note that the fact that the public parameters of the parties input to the key generation algorithm are ordered allows us to work with a somewhat simpler definition than the corresponding one in [45] for the standard MIFE case.

For an ad hoc MIFE scheme aMIFE as above and adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2)$, consider the experiment in Figure 1.

For an ad hoc MIFE scheme aMIFE as above and adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2)$, consider the experiment in Figure 1.

Experiment $\mathbf{IND}^{\mathsf{aMIFE}}_{\mathcal{A}}(1^\kappa)$
$(1^n, I, (\mathsf{PP}_i)_{i\in I}, st) \leftarrow\$ \mathcal{A}_0(1^\kappa)$
$b \leftarrow\$ \{0,1\}$
For all $i \notin I$ do:
$(\mathsf{MSK}_i, \mathsf{PP}_i) \leftarrow\$ \mathsf{aMIFE.Setup}(1^\kappa)$
$st \leftarrow\$ \mathcal{A}_1^{\mathcal{O}_{\mathsf{enc}}(\cdot,\cdot,\cdot),\mathcal{O}_{\mathsf{kg}}(\cdot,\cdot,\cdot)}(st)$
$b' \leftarrow\$ \mathcal{A}_2^{\mathcal{O}_{\mathsf{enc}}(\cdot,\cdot,\cdot),\mathcal{O}_{\mathsf{kg}}(\cdot,\cdot,\cdot)}((\mathsf{PP}_i)_{i\notin I}, st)$
Return $(b = b')$

Oracle $\mathcal{O}_{\mathsf{enc}}(i, \mathbf{x}_0, \mathbf{x}_1)$
If $\mathbf{x}_0, \mathbf{x}_1 \in \mathcal{M}_\kappa$ and $|\mathbf{x}_0| = |\mathbf{x}_1|$
$c_b \leftarrow\$ \mathsf{aMIFE.Enc}(\mathsf{MSK}_i, \mathbf{x}_b)$
Return c_b
Else Return $\perp$

Oracle $\mathcal{O}_{\mathsf{kg}}(i, (i_1, \ldots, i_\ell), f)$
If $i \in \{i_1, \cdots, i_\ell\}$
and $f \in \mathcal{F}_\kappa$ of arity ℓ
$\mathsf{PDK}_{i,f} \leftarrow$
$\mathsf{aMIFE.KeyGen}(i, \mathsf{MSK}_i, (\mathsf{PP}_{i_j}), f)$
Return $\mathsf{PDK}_{i,f}$
Else return $\perp$

Figure 1 Experiment for IND-security of ad hoc MIFE.

We say that $f \in \mathcal{F}_\kappa$ is *queried* if for every user associated with its input wires, either the user is corrupt, or has submitted it's partial decryption key to the adversary. Formally, for every $k \in \ell$ where ℓ is the arity of f either $i_k \in I$, *i.e.* i_k is corrupted or there is a key-generation query $(i_k, (\mathsf{PP}_{i_1}, \cdots, \mathsf{PP}_{i_\ell}), f)$.

We call $\mathcal{A}$ *legitimate* if for all $\kappa \in \mathbb{N}$, in all transcripts $\mathbf{IND}^{\mathsf{aMIFE}}_{\mathcal{A}}(1^\kappa)$ it holds that for every queried $f \in \mathcal{F}_\kappa$, there does not exist two sequences $(y_{i_1,0}, \cdots, y_{i_\ell,0})$ and $(y_{i_1,1}, \cdots, y_{i_\ell,1})$ such that

$$f(y_{i_1,0}, \cdots, y_{i_\ell,0}) \neq f(y_{i_1,1}, \cdots, y_{i_\ell,1})$$

and for every $j \in \{i_1, \cdots, i_\ell\}$

- $j \in I$, *i.e.* j is corrupted (so there is no restriction on $y_{j,0}, y_{j,1}$ above), or
- There is an encryption query $(j, \mathbf{x}_0, \mathbf{x}_1)$ such that $y_{j,0} = \mathbf{x}_0$ and $y_{j,1} = \mathbf{x}_1$.

We assume adversaries are legitimate unless otherwise stated. We call $\mathcal{A}$ *passive* if $I = \emptyset$. We call $\mathcal{A}$ *selective* if $\mathcal{A}_2$ makes no queries. We say that aMIFE is xxx-IND-secure if for any adversary $\mathcal{A}$ of type xxx

$$|\Pr\left[\mathbf{IND}^{\mathsf{aMIFE}}_{\mathcal{A}}(\cdot) \text{ outputs } 1\right] - 1/2| = \mathsf{negl}(\cdot)\ .$$

We provide the definition of simulation based security in Appendix C.

4 Ad Hoc MIFE from MIFE + Two-Round MPC

We show how to construct of ad hoc MIFE for any polynomial sized circuit from standard MIFE for the same functionality and a two-round MPC protocol.

Building Blocks

Our scheme will be using the following building blocks:

- A MIFE scheme

  $$\mathsf{MIFE} = (\mathsf{MIFE.Setup}, \mathsf{MIFE.KeyGen}, \mathsf{MIFE.Enc}, \mathsf{MIFE.Dec})$$

  for some message-space $\{\mathcal{M}_\kappa\}_{\kappa\in\mathbb{N}}$ and functionality $\{\mathcal{F}_\kappa\}_{\kappa\in\mathbb{N}}$. For simplicity, we assume $\mathsf{MIFE.KeyGen}$ is deterministic; note that this is without loss of generality since it can be made so by using a PRF.
- A two-round two-round MPC protocol

  $$\mathsf{MPC} = (\mathsf{MPC.RunRoundOne}, \mathsf{MPC.RoundRoundTwo}, \mathsf{MPC.ComputeResult})$$

  for programs of the form $\mathsf{GenKeys}_f$ in Figure 2 for $f \in \mathcal{F}_\kappa$. We assume MPC is function-rerunnable, unbounded and without setup (we discuss the other cases below).
- A PRF F with keyspace $\{\mathcal{K}_\kappa\}_{\kappa\in\mathbb{N}}$, and a punctured PRF puncF with keyspace $\{\mathcal{K}^{\mathsf{punc}}_\kappa\}_{\kappa\in\mathbb{N}}$, both with domain $\{0,1\}^*$. (We leave the ranges implicit for readability, taking the output to be sufficiently long.)
- A private-key single input functional encryption scheme

  $$\mathsf{FE} = (\mathsf{FE.Setup}, \mathsf{FE.KeyGen}, \mathsf{FE.Enc}, \mathsf{FE.Dec})$$

4.1 Construction

Below we provide our construction for adhoc MIFE for general circuits. Note that setup, encryption and key generation are done independently and in parallel by all the parties in the system.

$\mathsf{aMIFE.Setup}(1^\kappa)$: Upon input the security parameter, do the following:

1. Sample the seed of PRF $K \leftarrow\!\!\$\ \mathcal{K}_\kappa$ and the seed of a puncturable PRF $K^{\mathsf{punc}} \leftarrow\!\!\$\ \mathcal{K}^{\mathsf{punc}}_\kappa$. Puncturing will only be required in the proof.
2. Invoke the single input FE scheme, $(\mathsf{PP_{FE}}, \mathsf{MSK_{FE}}) \leftarrow\!\!\$\ \mathsf{FE.Setup}(1^\kappa)$.
3. Invoke the first round of the MPC protocol

   $$(\rho^{(1)}, \mathfrak{s}) \leftarrow\!\!\$\ \mathsf{MPC.RunRoundOne}(1^\kappa, (K, \mathsf{MSK_{FE}}))$$

   Note that the function is specified later.
4. Return $(\mathsf{PP} = \rho^{(1)}, \mathsf{MSK} = (K, K^{\mathsf{punc}}, \mathsf{MSK_{FE}}, \mathfrak{s}))$.

$\mathsf{aMIFE.KeyGen}((\mathsf{PP}_i)_{i\in[\ell]}, f, \mathsf{MSK})$: Upon input the public parameters of the ℓ parties that are chosen to participate in the computation, and the master secret key, do the following:

1. Parse the public parameters of each party as the first message in an MPC protocol, i.e. $\rho_i^{(1)} \leftarrow \mathsf{PP}_i \quad \forall i \in [\ell]$.
2. Parse the master secret key as $(K, K^{\mathsf{punc}}, \mathsf{MSK}_{\mathsf{FE}}, \mathfrak{s}) \leftarrow \mathsf{MSK}$
3. Run round two of the MPC protocol using round 1 messages as input, for the functionality $\mathsf{GenKeys}$ described in Figure 2:

$$\rho^{(2)} \leftarrow\!\!\$\ \mathsf{MPC.RunRoundTwo}(\mathfrak{s}, \mathsf{GenKeys}_{(\mathsf{PP}_i)_{i\in[\ell]},f}, (\rho_i^{(1)})_{i\in[\ell]})$$

4. Compute the mask $s \leftarrow \mathsf{PRF.Eval}(K, 0 \,\|\, (\mathsf{PP}_i)_{i\in[\ell]} \,\|\, f)$
5. Return $(\rho^{(2)}, s)$.

aMIFE.Enc($\mathsf{MSK}, \mathbf{x}$): Upon input the master secret key and the message $\mathbf{x}$, do the following:

1. Parse the master secret key as $(K, K^{\mathsf{punc}}, \mathsf{MSK}_{\mathsf{FE}}, \mathfrak{s}) \leftarrow \mathsf{MSK}$.
2. Sample the tag $T \leftarrow\!\!\$\ \{0,1\}^\kappa$.
3. Initialize the data structure Trap defined in Figure 10 by setting $\mathsf{mode\text{-}real} = 1$ and all other fields as $\perp$. This indicates that we are in the real system. The remaining fields are only relevant in the proof.
4. Compute the ciphertext $c \leftarrow\!\!\$\ \mathsf{FE.Enc}(\mathsf{MSK}_{\mathsf{FE}}, (\mathbf{x}, T, K^{\mathsf{punc}}, \mathsf{Trap}))$.
5. Return c.

aMIFE.Dec($(\mathsf{PDK}_{i,f})_{i\in[\ell]}, (c_i)_{i\in[\ell]}$): Upon input the partial decryption keys from all relevant parties, as well as ciphertexts from all relevant parties, do the following:

1. Parse $(\rho_i^{(2)}, s_i) \leftarrow \mathsf{PDK}_{i,f} \quad \forall i \in [\ell]$
2. Compute the output of the MPC protocol as $Z \leftarrow \mathsf{MPC.ComputeResult}((\rho_i^{(2)})_{i\in[\ell]})$
3. Unmask the output using partial shares provided by all parties. In more detail, compute $S \leftarrow \bigoplus_{i\in[\ell]} s_i$; $Z \leftarrow Z \oplus S$.
4. Parse the output of the MPC computation as $(\mathsf{SK}_f, \mathsf{SK}_{\mathsf{FE}_1}, \ldots, \mathsf{SK}_{\mathsf{FE}_\ell}) \leftarrow Z$.
5. Perform decryption of the single input FE scheme to obtain MIFE ciphertexts $\psi_i \leftarrow \mathsf{FE.Dec}(\mathsf{SK}_{\mathsf{FE}_i}, c_i) \quad \forall i \in [\ell]$.
6. Perform decryption of the MIFE scheme to obtain the output $y \leftarrow \mathsf{MIFE.Dec}(\mathsf{SK}_f, \psi_1, \ldots, \psi_\ell)$.
7. Return y.

▶ Remark 4. If we use a bounded 2-round MPC protocol then we will obtain a bounded ad hoc MIFE scheme where the setup algorithm also takes 1^n which is passed to $\mathsf{MPC.RunRoundOne}$. If MPC has a setup algorithm (outputting a CRS) then so does the resulting ad hoc MIFE scheme.

Correctness

Correctness follows from the correctness of MPC, MIFE and FE. In more detail, we have:

Step 1: MPC. By correctness of MPC, we have that the decryptor recovers the output $S \oplus (\mathsf{SK}_f, \mathsf{SK}_{\mathsf{FE}_1}, \ldots, \mathsf{SK}_{\mathsf{FE}_\ell})$. Next, if each party $i \in [\ell]$ provides a partial decryption key for f, then this contains partial mask s_i as part of the output of $\mathsf{aMIFE.KeyGen}$. Using these, the decryptor can compute $S \leftarrow \bigoplus_{i\in[\ell]} s_i$ and recover $(\mathsf{SK}_f, \mathsf{SK}_{\mathsf{FE}_1}, \ldots, \mathsf{SK}_{\mathsf{FE}_\ell})$.

Function $\mathsf{GenKeys}_{(\mathsf{PP}_i)_{i\in[\ell]},f}((K_1,\mathsf{MSK}_{\mathsf{FE}_1}),\ldots,(K_\ell,\mathsf{MSK}_{\mathsf{FE}_\ell}))$

1. For $i \in [\ell]$, compute randomness to be used for algorithms below:

$$r_i \leftarrow \mathsf{PRF.Eval}(K_i, 1 \parallel (\mathsf{PP}_i)_{i\in[\ell]} \parallel f), \qquad r_i' \leftarrow \mathsf{PRF.Eval}(K_i, 2 \parallel (\mathsf{PP}_i)_{i\in[\ell]} \parallel f)$$

2. Run the MIFE setup algorithm for the desired arity as:

$$((\mathsf{EK}_1,\ldots,\mathsf{EK}_\ell),\mathsf{MSK}) \leftarrow \mathsf{MIFE.Setup}(1^\kappa; r_1 \oplus \cdots \oplus r_\ell)$$

3. For $i \in [\ell]$, generate the single input FE function key:

$$\mathsf{SK}_{\mathsf{FE}_i} \leftarrow \mathsf{FE.KeyGen}(\mathsf{MSK}_{\mathsf{FE}_i}, \mathsf{ReEnc}_{\mathsf{EK}_i,\perp}; r_i')$$

4. Compute the MIFE secret key as $\mathsf{SK}_f \leftarrow \mathsf{MIFE.KeyGen}(\mathsf{MSK}, f)$
5. Sample the mask s_i for each partial key and compute the mask S as:

$$\forall i \in [\ell], \quad s_i \leftarrow \mathsf{PRF.Eval}(K_i, 0 \parallel (\mathsf{PP}_i)_{i\in[\ell]} \parallel f), \qquad S \leftarrow \bigoplus_{i\in[\ell]} s_i$$

6. Return the masked output $S \oplus (\mathsf{SK}_f, \mathsf{SK}_{\mathsf{FE}_1}, \ldots, \mathsf{SK}_{\mathsf{FE}_\ell})$

Figure 2 Functionality computed by the MPC protocol to generate single and multi input FE keys.

Function $\mathsf{ReEnc}_{\mathsf{EK},\mathsf{ctr}}(\mathbf{x}, T, K, \mathsf{Trap})$

1. Let $r \leftarrow \mathsf{PRF.Eval}(K, 0 \parallel \mathsf{EK} \parallel T)$.
2. If $\mathsf{mode} = R$ then compute $\mathsf{MIFE.Enc}(\mathsf{EK}, \mathbf{x}; r)$ and return it.
3. If $\mathsf{mode} = T$ and $\mathsf{ctr} < \mathsf{Trap.index}$ then compute $\mathsf{MIFE.Enc}(\mathsf{EK}, \mathsf{Trap.val}_1; r)$ and return it.
4. If $\mathsf{mode} = T$ and $\mathsf{ctr} > \mathsf{Trap.index}$ then compute $\mathsf{MIFE.Enc}(\mathsf{EK}, \mathsf{Trap.val}_0; r)$ and return it.
5. If $\mathsf{mode} = T$ and $\mathsf{ctr} = \mathsf{Trap.index}$ then output $\mathsf{Trap.CT}$.

Figure 3 Functionality for translating the ciphertext from FE to MIFE using dynamically generated encryption keys.

Step 2: FE. Next, by correctness of FE, we have that if

$$\psi_i = \mathsf{FE.Dec}(\mathsf{SK}_{\mathsf{FE}_i}, c_i) \quad \forall i \in [\ell]$$

Then, ψ_i are the MIFE ciphertexts computed as $\mathsf{MIFE.Enc}(\mathsf{EK}_i, \mathbf{x}_i; r_i)$.

Step 3: MIFE. Finally, by correctness of MIFE, we have that

$$f(\mathbf{x}_1,\ldots,\mathbf{x}_\ell) = \mathsf{MIFE.Dec}(\mathsf{SK}_f, \psi_1, \ldots, \psi_\ell)$$

4.2 Security Proof

In this section, we argue that the scheme described above is secure. In more detail:

▶ **Theorem 5.** *If* MIFE *is a IND-secure MIFE scheme and* MPC *is a SIM-secure 2-round MPC protocol, then our construction is* sel*-IND-secure.*

Due to space constraints, the proof can be found in Appendix D.

5 Ad Hoc Friendly MIFE and its Application to Inner Products

In this section, we describe a paradigm for constructing ad-hoc MIFE schemes from MIFE schemes that are "ad hoc friendly" and a (hopefully simple) two-round MPC protocol. This paradigm significantly simplifies our general construction. We then show that the standard MIFE scheme for inner products [3, 2], which may be based on DDH, LWE or DCR, is ad hoc friendly and the corresponding two-round MPC protocol is only required to compute inner-products, thus obtaining an efficient ad hoc MIFE scheme for inner products.

Ad Hoc Friendliness

In more detail, we define a notion of "ad hoc friendly" standard MIFE which satisfies the following properties:

- **Decentralized Setup.** The $\mathsf{MIFE.Setup}$ algorithm of the MIFE is decentralized in the sense that:
  1. The encryption keys EK_i for $i \in [n]$ corresponding to party i may be generated *independently* of the encryption keys of the remaining parties $[n] \setminus i$. In more detail, the algorithm $(\mathsf{EK}_1, \ldots, \mathsf{EK}_n) \leftarrow \mathsf{MIFE.Setup}(1^\kappa)$ may be decomposed into n invocations $\mathsf{EK}_i \leftarrow \mathsf{MIFE.SetupLocal}(1^\kappa)$ for $i \in [n]$, which can be run locally by each party.
  2. The master secret key MSK can be decomposed into n components $\{\mathsf{MSK}_i\}_{i\in[n]}$. The partial MSK_i corresponding to party i may be generated locally by party i, without any interaction or shared state with the remaining parties.

- **Local Encryption.** The $\mathsf{MIFE.Enc}$ algorithm of the MIFE is "local" in that it does not take as input the total number of parties or the public parameters of other parties. In more detail, $\mathsf{MIFE.Enc}$ algorithm only takes as input its encryption key EK_i and its input $\mathbf{x}_i$, and nothing else.

- **Piecewise Master Secret Key.** In standard MIFE schemes, the function is assumed to have fixed arity n. However, in ad hoc MIFE, we allow the function to have arity $\ell < n$. To support this, we require that the master secret in standard MIFE $\mathsf{MSK} = \{\mathsf{MSK}_1, \ldots, \mathsf{MSK}_n\}$, if restricted to some subset $S \subseteq [n]$ with $|S| = \ell$, has the same distribution as a master secret generated for functions of arity ℓ.
  Formally, let $\mathsf{MSK} = \{\mathsf{MSK}_1, \ldots, \mathsf{MSK}_n\} \leftarrow \mathsf{FE.Setup}(1^\kappa, 1^n)$ and $\mathsf{MSK}' = \{\mathsf{MSK}'_1, \ldots, \mathsf{MSK}'_\ell\} \leftarrow \mathsf{FE.Setup}(1^\kappa, 1^\ell)$. Then, we require that MSK restricted to subset S, namely $(\mathsf{MSK}_{S[1]}, \ldots, \mathsf{MSK}_{S[\ell]})$ has the same distribution as MSK'.

Since the intuition was discussed in Section 1, we proceed to our construction of ad hoc MIFE for inner products.

5.1 Ad Hoc MIFE for Inner Products

Inner-product functionality

We recall the *multi-input inner-product functionality* over $\mathbb{Z}_p$ for a prime p, adapted from Abdalla et al. [2, Section 2.3]. For $m, n \in \mathbb{N}$, this is the functionality

$$\mathcal{IP}^m_{p,n} = \{\mathsf{ip}_{\mathbf{y}_1,\ldots,\mathbf{y}_n} \colon (\mathbb{Z}_p^m)^n \to \mathbb{Z}_p\}$$

defined by

$$\mathsf{ip}_{\mathbf{y}_1,\ldots,\mathbf{y}_n}(\mathbf{x}_1,\ldots,\mathbf{x}_n) \;=\; \sum_{i=1}^{n}\langle \mathbf{x}_i, \mathbf{y}_i\rangle \bmod p\ .$$

We omit parameters p, m, n when they are arbitrary or clear from context.

5.2 Building Blocks

In the context of ad hoc MIFE for inner products, we want to evaluate a functionality given by a sequence of vectors $\mathbf{y} = (\mathbf{y_1}, \ldots, \mathbf{y_n})$ where $\mathbf{y_i} \in \mathbb{Z}_q^m$. Evaluating the function on inputs $\{\mathbf{x_i}\}_{i\in[n]}$ where $\mathbf{x_i} \in \mathbb{Z}_q^m$ reveals $\sum_{i\in[n]}\langle \mathbf{x_i}, \mathbf{y_i}\rangle$ and nothing more. In particular the evaluator should not be able to learn the partial sums $\langle \mathbf{x_i}, \mathbf{y_i}\rangle$.

Our scheme will be using the following building blocks:

- A 2-round *function-rerunnable MPC for a functionality* GenKey-ip*, which must be support inner product computation.*
- *A standard MIFE with for the inner product functionality, denoted by* $\mathsf{MIFE}_{\mathsf{ip}}$*, satisfying the aforementioned ad hoc friendly properties.*

The MIFE scheme(s) of Abdalla et al. [2]:

Abdalla et al. [2] provide two multi-input encryption schemes for inner products, one for computing inner products over some finite ring $\mathbb{Z}_L$, and the second for computing bounded-norm inner products over the integers. Both schemes rely on:

1. An information theoretic scheme for inner products where only one ciphertext query is supported.
2. A single input functional encryption scheme FE for inner products which is applied on top of the above one time scheme.

Unrolling the above two components, the final MIFE scheme(s) of [2] have algorithms of the form described below.

Below, we unroll the above two components to establish that the schemes of [2] satisfy ad hoc friendliness.

1. **Decentralized Setup.** The encryption keys EK_i corresponding to party i may be generated *independently* of the encryption keys of the remaining parties $[n] \setminus i$. In more detail, the setup algorithm is defined as:

   $\mathsf{MIFE.Setup}(1^\kappa, n)$: Do the following:
   - For $i \in [n]$, sample $\mathbf{u}_i \leftarrow \mathbb{Z}_L^m$.
   - For $i \in [n]$, sample $(\mathsf{FE.PK}_i, \mathsf{FE.MSK}_i) \leftarrow \mathsf{FE.Setup}(1^\kappa, 1^m)$.
   - Output $\mathsf{PP}_i = \mathsf{FE.PK}_i$ and $\mathsf{EK}_i = (\mathsf{FE.MSK}_i,\ \mathbf{u}_i)$ for $i \in [n]$.

   Then, we may define:

   a. $\mathsf{MIFE.SetupLocal}(1^\kappa, n)$: Do the following:
      - Sample $\mathbf{u} \leftarrow \mathbb{Z}_L^m$.
      - Sample $(\mathsf{FE.PK}, \mathsf{FE.MSK}) \leftarrow \mathsf{FE.Setup}(1^\kappa, 1^m)$.
      - Output $\mathsf{PP} = \mathsf{FE.PP}$ and $\mathsf{EK} = (\mathsf{FE.MSK},\ \mathbf{u})$.

To compute the set of n encryption keys, the algorithm $\mathsf{MIFE.SetupLocal}(1^\kappa, n)$ is invoked n times. Additionally, in [2], the master secret key can be decomposed into n components by setting:

$$\mathsf{MSK}_i = \mathsf{EK}_i = (\mathsf{FE.MSK}_i, \mathbf{u}_i) \; \forall \;\; i \in [n]$$

2. **Local Encryption.** The encryption algorithm only takes its encryption key and message as input and does not depend on the number of parties or their public parameters. In more detail, the encryption algorithm is defined as:

   $\mathsf{MIFE.Enc}(\mathsf{EK}_i, \mathbf{x}_i)$: Do the following:
   - Parse $\mathsf{EK}_i = (\mathsf{FE.MSK}_i, \; \mathbf{u}_i)$.
   - Compute $\mathbf{y}_i = \mathbf{x}_i + \mathbf{u}_i \mod L$.
   - Compute $\mathbf{c}_i = \mathsf{FE.Enc}(\mathsf{FE.MSK}_i, \mathbf{y}_i)$.
   - Output $(\mathbf{y}_i, \mathbf{c}_i)$.

   Thus, the ciphertext encoding party i's input may be computed independently by party i.

3. **Piecewise Master Secret Key.** For the inner product functionality, if $\mathsf{MSK} = (\mathsf{MSK}_1, \ldots, \mathsf{MSK}_n)$ is the master secret key for function vector $\mathbf{y} = (\mathbf{y}_1 \| ... \| \mathbf{y}_n)$ then for any $S \subseteq [n]$, we have the corresponding master key $\mathsf{MSK}' = (\mathsf{MSK}_{S[1]}, \ldots, \mathsf{MSK}_{S[\ell]})$ is a well formed master secret key for the vector $\mathbf{y}' = (\mathbf{y}_{S[1]} \| \ldots \| \mathbf{y}_{S[\ell]})$. In more detail, the key generation algorithm is defined as:

   $\mathsf{MIFE.KeyGen}(\mathsf{MSK}, \mathbf{y})$: Do the following:
   - Output $\mathsf{DK}_\mathbf{y} \leftarrow \big(\{\mathsf{FE.KeyGen}(\mathsf{MSK}_i, \mathbf{y}_i)\}_{i \in [n]}, \; \sum_{i \in [n]} \langle \mathbf{u}_i, \; \mathbf{y}_i \rangle \big)$.

   It is easy to see that the function key for $\mathbf{y}'$ can be obtained from the above by simply setting $\mathbf{y}_i = 0$ for $i \notin S$.

5.3 Our Construction

We are now ready to present the construction. Note that for ease of presentation, we describe the scheme for all n users but we remark that it works for any subset of $\ell \leq n$ users.

aMIFE.Setup$(1^\kappa, 1^m)$: Upon input the security parameter and the dimension of the input vector for each party, do the following:

1. Run the partial MIFE setup algorithm to obtain the public parameters and encryption key: $(\mathsf{MIFE.PP}, \mathsf{MIFE.EK}) \leftarrow\!\!\$ \; \mathsf{MIFE.SetupLocal}(1^\kappa, 1^m)$
2. Invoke the first round of the MPC protocol with the encryption key as input:

   $$(\rho^{(1)}, \mathfrak{s}) \leftarrow\!\!\$ \; \mathsf{MPC.RunRoundOne}(1^\kappa, \mathsf{EK})$$

3. Return $\mathsf{PP} := (\mathsf{MIFE.PP}, \rho^{(1)}), \mathsf{MSK} := (\mathsf{MIFE.EK}, \mathfrak{s})$

aMIFE.Enc$(\mathsf{EK}, \mathrm{x})$: Upon input the encryption key and the input, compute $\mathsf{MIFE.enc}(\mathsf{EK}, \mathbf{x})$ and output it.

aMIFE.KeyGen$((\mathsf{PP}_i)_{i \in [\ell]}, \mathrm{y}, \mathsf{MSK}_i)$: Upon input the public parameters of the ℓ parties, the function vector $\mathbf{y}$ and the master secret key MSK_i, do the following:

1. Parse $(\mathsf{MIFE.EK}, \mathfrak{s}) \leftarrow \mathsf{MSK}_i$ and $(\mathsf{MIFE.PP}_j, \rho_j^{(1)}) \leftarrow \mathsf{PP}_j \quad \forall j \in [\ell]$
2. Parse $(\mathbf{y_1}, \ldots, \mathbf{y}_\ell) \leftarrow \mathbf{y}$ where $\mathbf{y_j} \in \mathbb{Z}_q^m$ for $j \in [\ell]$.

3. Invoke round 2 of the MPC protocol $\mathsf{GenKey\text{-}ip}_{\mathbf{y}}$ as defined in Figure 4

$$\rho^{(2)} \leftarrow\!\!\$\ \mathsf{MPC.RunRoundTwo}(\mathfrak{s}, \rho_1^{(1)}, \ldots, \rho_\ell^{(1)},)$$

4. Return $\mathsf{PDK} := \rho^{(2)}$.

aMIFE.Dec$((\mathsf{PDK}_{i,f})_{i\in[\ell]}, (\mathbf{c}_i)_{i\in[\ell]})$**:** Upon input the partial decryption keys from all relevant parties, as well as ciphertexts from all relevant parties, do the following:

1. Compute the output of the MPC protocol as

$$\mathsf{MIFE.DK}_{\mathbf{y}} \leftarrow \mathsf{MPC.ComputeResult}((\rho_i^{(2)})_{i\in[\ell]})$$

2. Compute $\mathsf{MIFE.Dec}(\mathsf{MIFE.DK}_{\mathbf{y}}, \mathbf{y}, \mathbf{c}_1, \ldots, \mathbf{c}_\ell)$ and output it.

Function $\mathsf{GenKey\text{-}ip}_{\mathbf{y}}(\mathsf{EK}_1, \ldots, \mathsf{EK}_\ell)$

1. Let $\mathsf{MIFE.MSK} = (\mathsf{EK}_1, \ldots, \mathsf{EK}_\ell)$.
2. Compute $\mathsf{MIFE.KeyGen}(\mathsf{MIFE.MSK}, \mathbf{y})$ and output it.

Figure 4 Functionality for computing the MIFE function key.

Note that for the inner product functionality, the MIFE key generation algorithm is very simple and in some cases only involves computing inner products, please see [2] for details.

Correctness

Correctness follows from correctness of the MPC protocol and of the standard MIFE scheme. We have by correctness of the MPC protocol, that the output $\mathsf{MIFE.DK}_{\mathbf{y}} = \mathsf{MIFE.KeyGen}(\mathsf{MSK}, \mathbf{y})$ is produced correctly. Since the encryptors encrypted $\mathbf{c}_i = \mathsf{MIFE.Enc}(\mathsf{EK}_i, \mathbf{x}_i)$, it follows from the correctness of MIFE that $\mathsf{MIFE.Dec}(\mathsf{MIFE.DK}_{\mathbf{y}}, \mathbf{y}, \mathbf{c}_1, \ldots, \mathbf{c}_\ell)$ outputs $\sum_{i\in[\ell]} \langle \mathbf{y}_i,\ \mathbf{x}_i \rangle$ as desired.

Security

Given the proof of security in Section 4, the proof of security of the present construction is straightforward, since the present construction is a (much) simplified instance of the general construction. Intuitively, the security of MPC ensures that the output $\mathsf{MIFE.DK}_{\mathbf{y}}$, which is computed using inputs $(\mathsf{MSK}_i, \mathbf{y}_i)_{i\in[\ell]}$ of ℓ disjoint parties, is indistinguishable from the output of a "global" MIFE key generation algorithm which takes the entire $(\mathsf{MSK}, \mathbf{y})$ as input. The encryption algorithm is exactly the same as that of the standard MIFE scheme, with the result that the decryptor sees exactly the same view as in the standard MIFE scheme. Please see Appendix E for details.

An issue with the MIFE scheme of Abdalla et al. [2] which we use above is that an adversary may exploit partial ciphertexts to learn unauthorized information [43]. Specifically, say there are two parties, and the first one provides ciphertexts, for vectors $\mathbf{x}_0$ and $\mathbf{x}_1$ (say). The second encryptor does not give any ciphertexts. Suppose the key generation algorithm in the standard MIFE scheme gives a key for $(\mathbf{y} \parallel \mathbf{0})$. Now, it could be that $\langle \mathbf{x}_0, \mathbf{y} \rangle \neq \langle \mathbf{x}_1, \mathbf{y} \rangle$ but this does not violate admissibility, because admissibility only checks for decryption with respect to complete ciphertexts whereas decryption with respect to partial ciphertexts is not defined. Ideally, the key for $(\mathbf{y} \parallel \mathbf{0})$ should not work to decrypt the partial ciphertexts of encryptor 1, but in the construction of Abdalla et al., it does so (if the corresponding sub-vector in the key is zero).

Note that in a MIFE scheme, the above situation only occurs if one party (party 2 say) *never* gives any ciphertext[5]. In our setting however, a party issues a partial decryption key only if it wishes its data to participate in some computation. Hence, we resolve the issue by requiring that a party only issue a partial decryption key if it has also issued at least one ciphertext.

Instantiating MPC

Since function-rerunnable two-round MPC in the common reference string (CRS) model can be constructed [28, 57, 24, 59] from learning-with-errors (LWE), we get ad hoc MIFE for inner products from LWE. This result can be upgraded to the malicious setting as the additional cost of NIZKs.

While function rerunnable two-round MPC in the CRS model can be constructed from bilinear maps [41] and even two-round oblivious transfer [15, 42, 39] or information theoretically [9, 36], these constructions are *not* function-rerunnable so do not suffice for multi-key ad hoc MIFE. However, if we restrict ourselves to the setting of bounded ad hoc MIFE, where a user issues only a bounded number of partial decryption keys and additionally maintains state across key issues, we may use the above MPC protocols (just via repetition). This yields such a bounded ad hoc MIFE under DDH, LWE or DCR for the both the semi-honest and malicious cases. One nice feature of the semi-honest construction is that it does not use a common random string and is in the plain model.

References

1. Michel Abdalla, Florian Bourse, Angelo De Caro, and David Pointcheval. Simple Functional Encryption Schemes for Inner Products. In Jonathan Katz, editor, *PKC 2015*, volume 9020 of *LNCS*, pages 733–751. Springer, Heidelberg, March/April 2015. doi:10.1007/978-3-662-46447-2_33.
2. Michel Abdalla, Dario Catalano, Dario Fiore, Romain Gay, and Bogdan Ursu. Multi-Input Functional Encryption for Inner Products: Function-Hiding Realizations and Constructions Without Pairings. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 597–627. Springer, Heidelberg, August 2018. doi:10.1007/978-3-319-96884-1_20.
3. Michel Abdalla, Romain Gay, Mariana Raykova, and Hoeteck Wee. Multi-input Inner-Product Functional Encryption from Pairings. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part I*, volume 10210 of *LNCS*, pages 601–626. Springer, Heidelberg, April/May 2017. doi:10.1007/978-3-319-56620-7_21.
4. Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient Lattice (H)IBE in the Standard Model. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 553–572. Springer, Heidelberg, May/June 2010. doi:10.1007/978-3-642-13190-5_28.
5. Shweta Agrawal, David Mandell Freeman, and Vinod Vaikuntanathan. Functional Encryption for Inner Product Predicates from Learning with Errors. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 21–40. Springer, Heidelberg, December 2011. doi:10.1007/978-3-642-25385-0_2.
6. Shweta Agrawal, Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional Encryption: New Perspectives and Lower Bounds. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 500–518. Springer, Heidelberg, August 2013. doi:10.1007/978-3-642-40084-1_28.

[5] If party 2 gives even a single ciphertext (say for vector $\mathbf{z}_0$) then admissibility will force $\langle \mathbf{x}_0 \parallel \mathbf{z}_0, \mathbf{y} \parallel \mathbf{0} \rangle = \langle \mathbf{x}_1 \parallel \mathbf{z}_0, \mathbf{y} \parallel \mathbf{0} \rangle$ which implies that $\langle \mathbf{x}_0, \mathbf{y} \rangle = \langle \mathbf{x}_1, \mathbf{y} \rangle$.

7 Shweta Agrawal, Benoît Libert, and Damien Stehlé. Fully Secure Functional Encryption for Inner Products, from Standard Assumptions. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part III*, volume 9816 of *LNCS*, pages 333–362. Springer, Heidelberg, August 2016. doi:10.1007/978-3-662-53015-3_12.

8 Prabhanjan Ananth and Abhishek Jain. Indistinguishability Obfuscation from Compact Functional Encryption. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 308–326. Springer, Heidelberg, August 2015. doi:10.1007/978-3-662-47989-6_15.

9 Benny Applebaum, Zvika Brakerski, and Rotem Tsabary. Perfect Secure Computation in Two Rounds. In *TCC 2018, Part I*, LNCS, pages 152–174. Springer, Heidelberg, March 2018. doi:10.1007/978-3-030-03807-6_6.

10 Saikrishna Badrinarayanan, Divya Gupta, Abhishek Jain, and Amit Sahai. Multi-input Functional Encryption for Unbounded Arity Functions. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT 2015, Part I*, volume 9452 of *LNCS*, pages 27–51. Springer, Heidelberg, November/December 2015. doi:10.1007/978-3-662-48797-6_2.

11 Carmen Elisabetta Zaira Baltico, Dario Catalano, Dario Fiore, and Romain Gay. Practical Functional Encryption for Quadratic Functions with Applications to Predicate Encryption. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 67–98. Springer, Heidelberg, August 2017. doi:10.1007/978-3-319-63688-7_3.

12 Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. *J. ACM*, 59(2):6, 2012.

13 Boaz Barak, Iftach Haitner, Dennis Hofheinz, and Yuval Ishai. Bounded Key-Dependent Message Security. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 423–444. Springer, Heidelberg, May/June 2010. doi:10.1007/978-3-642-13190-5_22.

14 Amos Beimel, Ariel Gabizon, Yuval Ishai, Eyal Kushilevitz, Sigurd Meldgaard, and Anat Paskin-Cherniavsky. Non-Interactive Secure Multiparty Computation. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part II*, volume 8617 of *LNCS*, pages 387–404. Springer, Heidelberg, August 2014. doi:10.1007/978-3-662-44381-1_22.

15 Fabrice Benhamouda and Huijia Lin. k-Round Multiparty Computation from k-Round Oblivious Transfer via Garbled Interactive Circuits. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 500–532. Springer, Heidelberg, April/May 2018. doi:10.1007/978-3-319-78375-8_17.

16 John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-Policy Attribute-Based Encryption. In *2007 IEEE Symposium on Security and Privacy*, pages 321–334. IEEE Computer Society Press, May 2007. doi:10.1109/SP.2007.11.

17 Nir Bitansky and Vinod Vaikuntanathan. Indistinguishability Obfuscation from Functional Encryption. In Venkatesan Guruswami, editor, *56th FOCS*, pages 171–190. IEEE Computer Society Press, October 2015. doi:10.1109/FOCS.2015.20.

18 Dan Boneh and Matthew K. Franklin. Identity-Based Encryption from the Weil Pairing. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 213–229. Springer, Heidelberg, August 2001. doi:10.1007/3-540-44647-8_13.

19 Dan Boneh and Matthew K. Franklin. Identity-Based Encryption from the Weil Pairing. In *CRYPTO*, pages 213–229, 2001. doi:10.1007/3-540-44647-8_13.

20 Dan Boneh, Amit Sahai, and Brent Waters. Functional Encryption: Definitions and Challenges. In Yuval Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 253–273. Springer, Heidelberg, March 2011. doi:10.1007/978-3-642-19571-6_16.

21 Dan Boneh and Brent Waters. Conjunctive, Subset, and Range Queries on Encrypted Data. In Salil P. Vadhan, editor, *TCC 2007*, volume 4392 of *LNCS*, pages 535–554. Springer, Heidelberg, February 2007. doi:10.1007/978-3-540-70936-7_29.

22 Xavier Boyen and Brent Waters. Anonymous Hierarchical Identity-Based Encryption (Without Random Oracles). In Cynthia Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 290–307. Springer, Heidelberg, August 2006. doi:10.1007/11818175_17.

23 Zvika Brakerski, Ilan Komargodski, and Gil Segev. Multi-input Functional Encryption in the Private-Key Setting: Stronger Security from Weaker Assumptions. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 852–880. Springer, Heidelberg, May 2016. doi:10.1007/978-3-662-49896-5_30.

24 Zvika Brakerski and Renen Perlman. Lattice-Based Fully Dynamic Multi-key FHE with Short Ciphertexts. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 190–213. Springer, Heidelberg, August 2016. doi:10.1007/978-3-662-53018-4_8.

25 David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai Trees, or How to Delegate a Lattice Basis. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 523–552. Springer, Heidelberg, May/June 2010. doi:10.1007/978-3-642-13190-5_27.

26 Nishanth Chandran, Vipul Goyal, Aayush Jain, and Amit Sahai. Functional Encryption: Decentralised and Delegatable. *IACR Cryptology ePrint Archive*, 2015:1017, 2015. URL: http://dblp.uni-trier.de/db/journals/iacr/iacr2015.html#ChandranGJS15.

27 Jérémy Chotard, Edouard Dufour Sans, Romain Gay, Duong Hieu Phan, and David Pointcheval. Decentralized Multi-Client Functional Encryption for Inner Product. In Thomas Peyrin and Steven Galbraith, editors, *ASIACRYPT 2018, Part II*, LNCS, pages 703–732. Springer, Heidelberg, December 2018. doi:10.1007/978-3-030-03329-3_24.

28 Michael Clear and Ciaran McGoldrick. Multi-identity and Multi-key Leveled FHE from Learning with Errors. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 630–656. Springer, Heidelberg, August 2015. doi:10.1007/978-3-662-48000-7_31.

29 Clifford Cocks. An Identity Based Encryption Scheme Based on Quadratic Residues. In *IMA Int. Conf.*, volume 2260 of *Lecture Notes in Computer Science*, pages 360–363. Springer, 2001.

30 Nico Döttling and Sanjam Garg. Identity-Based Encryption from the Diffie-Hellman Assumption. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 537–569. Springer, Heidelberg, August 2017. doi:10.1007/978-3-319-63688-7_18.

31 Uriel Feige, Joe Kilian, and Moni Naor. A minimal model for secure computation (extended abstract). In *26th ACM STOC*, pages 554–563. ACM Press, May 1994. doi:10.1145/195058.195408.

32 Sanjam Garg, Craig Gentry, Shai Halevi, and Mariana Raykova. Two-Round Secure MPC from Indistinguishability Obfuscation. In Yehuda Lindell, editor, *TCC 2014*, volume 8349 of *LNCS*, pages 74–94. Springer, Heidelberg, February 2014. doi:10.1007/978-3-642-54242-8_4.

33 Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate Indistinguishability Obfuscation and Functional Encryption for all Circuits. In *54th FOCS*, pages 40–49. IEEE Computer Society Press, October 2013. doi:10.1109/FOCS.2013.13.

34 Sanjam Garg, Craig Gentry, Shai Halevi, Amit Sahai, and Brent Waters. Attribute-Based Encryption for Circuits from Multilinear Maps. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 479–499. Springer, Heidelberg, August 2013. doi:10.1007/978-3-642-40084-1_27.

35 Sanjam Garg, Mohammad Hajiabadi, Mohammad Mahmoody, and Ahmadreza Rahimi. Registration-Based Encryption: Removing Private-Key Generator from IBE. In *TCC 2018, Part I*, LNCS, pages 689–718. Springer, Heidelberg, March 2018. doi:10.1007/978-3-030-03807-6_25.

36 Sanjam Garg, Yuval Ishai, and Akshayaram Srinivasan. Two-Round MPC: Information-Theoretic and Black-Box. In *TCC 2018, Part I*, LNCS, pages 123–151. Springer, Heidelberg, March 2018. doi:10.1007/978-3-030-03807-6_5.

37 Sanjam Garg, Mohammad Mahmoody, and Ameer Mohammed. Lower Bounds on Obfuscation from All-or-Nothing Encryption Primitives. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 661–695. Springer, Heidelberg, August 2017. doi:10.1007/978-3-319-63688-7_22.

38 Sanjam Garg, Mohammad Mahmoody, and Ameer Mohammed. When Does Functional Encryption Imply Obfuscation? In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 82–115. Springer, Heidelberg, November 2017. doi:10.1007/978-3-319-70500-2_4.

39 Sanjam Garg, Peihan Miao, and Akshayaram Srinivasan. Two-Round Multiparty Secure Computation Minimizing Public Key Operations. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part III*, volume 10993 of *LNCS*, pages 273–301. Springer, Heidelberg, August 2018. doi:10.1007/978-3-319-96878-0_10.

40 Sanjam Garg, Pratyay Mukherjee, Omkant Pandey, and Antigoni Polychroniadou. The Exact Round Complexity of Secure Computation. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 448–476. Springer, Heidelberg, May 2016. doi:10.1007/978-3-662-49896-5_16.

41 Sanjam Garg and Akshayaram Srinivasan. Garbled Protocols and Two-Round MPC from Bilinear Maps. In *58th FOCS*, pages 588–599. IEEE Computer Society Press, 2017. doi:10.1109/FOCS.2017.60.

42 Sanjam Garg and Akshayaram Srinivasan. Two-Round Multiparty Secure Computation from Minimal Assumptions. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 468–499. Springer, Heidelberg, April/May 2018. doi:10.1007/978-3-319-78375-8_16.

43 Romain Gay. Personal Communication, 2019.

44 Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 197–206. ACM Press, May 2008. doi:10.1145/1374376.1374407.

45 Shafi Goldwasser, S. Dov Gordon, Vipul Goyal, Abhishek Jain, Jonathan Katz, Feng-Hao Liu, Amit Sahai, Elaine Shi, and Hong-Sheng Zhou. Multi-input Functional Encryption. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 578–602. Springer, Heidelberg, May 2014. doi:10.1007/978-3-642-55220-5_32.

46 Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. Reusable garbled circuits and succinct functional encryption. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 555–564. ACM Press, June 2013. doi:10.1145/2488608.2488678.

47 Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Predicate Encryption for Circuits from LWE. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 503–523. Springer, Heidelberg, August 2015. doi:10.1007/978-3-662-48000-7_25.

48 S. Dov Gordon, Feng-Hao Liu, and Elaine Shi. Constant-Round MPC with Fairness and Guarantee of Output Delivery. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 63–82. Springer, Heidelberg, August 2015. doi:10.1007/978-3-662-48000-7_4.

49 Vipul Goyal. Reducing Trust in the PKG in Identity Based Cryptosystems. In Alfred Menezes, editor, *CRYPTO 2007*, volume 4622 of *LNCS*, pages 430–447. Springer, Heidelberg, August 2007. doi:10.1007/978-3-540-74143-5_24.

50 Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-Based Encryption for Fine-Grained Access Control of Encrypted Data. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 06*, pages 89–98. ACM Press, October/November 2006. Available as Cryptology ePrint Archive Report 2006/309. doi:10.1145/1180405.1180418.

51 Shai Halevi, Yuval Ishai, Abhishek Jain, Ilan Komargodski, Amit Sahai, and Eylon Yogev. Non-Interactive Multiparty Computation Without Correlated Randomness. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part III*, volume 10626 of *LNCS*, pages 181–211. Springer, Heidelberg, December 2017. doi:10.1007/978-3-319-70700-6_7.

52 Shai Halevi, Yuval Ishai, Abhishek Jain, Ilan Komargodski, Amit Sahai, and Eylon Yogev. Non-Interactive Multiparty Computation without Correlated Randomness. Cryptology ePrint Archive, Report 2017/871, 2017. URL: http://eprint.iacr.org/2017/871.

53 Shai Halevi, Yuval Ishai, Abhishek Jain, Eyal Kushilevitz, and Tal Rabin. Secure Multiparty Computation with General Interaction Patterns. In Madhu Sudan, editor, *ITCS 2016*, pages 157–168. ACM, January 2016. doi:10.1145/2840728.2840760.

54 Jonathan Katz, Amit Sahai, and Brent Waters. Predicate Encryption Supporting Disjunctions, Polynomial Equations, and Inner Products. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 146–162. Springer, Heidelberg, April 2008. doi:10.1007/978-3-540-78967-3_9.

55 Allison B. Lewko, Tatsuaki Okamoto, Amit Sahai, Katsuyuki Takashima, and Brent Waters. Fully Secure Functional Encryption: Attribute-Based Encryption and (Hierarchical) Inner Product Encryption. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 62–91. Springer, Heidelberg, May/June 2010. doi:10.1007/978-3-642-13190-5_4.

56 Huijia Lin. Indistinguishability Obfuscation from SXDH on 5-Linear Maps and Locality-5 PRGs. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 599–629. Springer, Heidelberg, August 2017. doi:10.1007/978-3-319-63688-7_20.

57 Pratyay Mukherjee and Daniel Wichs. Two Round Multiparty Computation via Multi-key FHE. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 735–763. Springer, Heidelberg, May 2016. doi:10.1007/978-3-662-49896-5_26.

58 Adam O'Neill. Definitional Issues in Functional Encryption. Cryptology ePrint Archive, Report 2010/556, 2010. URL: http://eprint.iacr.org/2010/556.

59 Chris Peikert and Sina Shiehian. Multi-key FHE from LWE, Revisited. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 217–238. Springer, Heidelberg, October/November 2016. doi:10.1007/978-3-662-53644-5_9.

60 Phillip Rogaway. The Moral Character of Cryptographic Work, December 2015. URL: http://web.cs.ucdavis.edu/~rogaway/papers/moral.html.

61 Amit Sahai and Brent R. Waters. Fuzzy Identity-Based Encryption. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 457–473. Springer, Heidelberg, May 2005. doi:10.1007/11426639_27.

62 Brent Waters. Functional Encryption for Regular Languages. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 218–235. Springer, Heidelberg, August 2012. doi:10.1007/978-3-642-32009-5_14.

A Additional Related Work

Functional Encryption

FE started with the notion of "attribute-based encryption" [61, 50] and evolved over time to a more general primitive that encompasses several primitives such as (hierarchical) identity based encryption [19, 29, 22, 44, 25, 4], attribute based encryption [61, 50, 16, 34], predicate encryption [21, 47, 54, 55, 5, 62, 47] and reusable garbled circuits [46]. Formal definitions of the general primitive were first given in [20, 58]. While there has been substantial progress in constructing FE from standard assumptions [1, 7, 56], the general notion of FE for arbitrary polynomial sized circuits was constructed in the breakthrough work of [33] from indistinguishability obfuscation (iO) [12, 33, 56]. Functional encryption for restricted functionalities such as inner products [1, 7], and quadratic functions [56, 11] from more standard assumptions has also been developed.

Multi-Input Functional Encryption

Extending the more basic concept of functional encryption (FE) [61, 20, 58], the notion of multi-input function encryption (MIFE) was first introduced by Goldwasser et al. [45] and there have since been a number of follow-up works. Ananth and Jain [8] show that private key MIFE for general polynomial-arity functions *implies* iO. On the other hand, Brakerski, Komargodski and Segev [23] construct private-key MIFE for constant-arity functions, based on a private-key single-input FE scheme. They achieve adaptive security but also do not consider sender corruption. Badrinarayanan et al. [10] construct MIFE schemes for "unbounded arity" functions. More recent work [3, 2] constructs *inner-product* MIFE.

Multi-Party Computation

Traditional MPC is *interactive*. Ad hoc MIFE can be seen as a special form of *non-interactive* MPC [31, 14, 53]. In particular, ad hoc MIFE separates inputs and functions, which affords greater flexibility – one can use the same encrypted inputs with different functions, or different encrypted inputs with the same function. Moreover, previous non-interactive MPC protocols require a global setup procedure. In a recent work, [52] constructs non-interactive MPC from indistinguishability obfuscation and DDH, assuming a PKI setup and a CRS, without this requirement. In contrast, our schemes are based on standard MIFE for a given functionality and do not require a CRS in general, as we do not necessarily consider sender corruption.

Multi-Authority Functional Encryption

Our work should also be compared to that of Chandran et al. [26], who proposed a notion of "multi-authority" FE (MAFE). In MAFE, key authorities independently generate their own keys. Roughly speaking, to derive a decryption key for a function f, a user must obtain a partial decryption key for f from each authority. In our context, we could think of the authorities as sources. However, a fundamental difference between multi-authority FE and ad hoc MIFE is that in the former, to encrypt, one needs to know the master public keys of all authorities (users). This is a severe limitation, as a user may not be aware of which other parties are to be involved in a computation at the time of encryption. Furthermore, in multi-authority FE, a given ciphertext can only be used in a computation associated with one fixed group, unlike ad hoc MIFE, where a ciphertext can be used in an unbounded number of dynamically-chosen groups. Finally, in MAFE, decryption only operates on a *single* ciphertext, unlike our notion which is intrinsically multi-user.

B Preliminaries

In this section we define the notation and preliminaries used in our work.

B.1 Notation and Conventions

PPT stands for "probabilistic polynomial time" and PT stands for "polynomial time." Algorithms are PPT unless otherwise noted. Throughout, κ denotes the security parameter and 1^κ its unary encoding. For a probabilistic algorithm $\mathcal{A}$, we denote by $\mathcal{A}(x;r)$ the output of $\mathcal{A}$ on input x with random tape r. We denote $y \leftarrow\!\!\$\, A(x)$ as the process of sampling r at random and letting $y \leftarrow A(x;r)$. For a finite set S, we denote $x \leftarrow\!\!\$\, S$ as the process of sampling x uniformly from S. For a distribution $\mathcal{D}$ we denote $x \leftarrow\!\!\$\, \mathcal{D}$ as the process of sampling x according to $\mathcal{D}$. For $k \in \mathbb{N}$ we let $[k]$ denote the set $\{1, \cdots, k\}$. If s is string

then $|s|$ denotes its length and $s[i]$ denotes its i-th bit. If $\mathbf{x}$ is a vector then $|\mathbf{x}|$ denotes its number of components and $\mathbf{x}[i]$ denotes its i-th component. We will use $\mathsf{negl}(\cdot)$ to denote an unspecified negligible function and $\mathsf{poly}(\cdot)$ to denote an unspecified polynomial. We say that (families of) distributions $\{\mathcal{D}_{0,\kappa}\}_{\kappa\in\mathbb{N}}, \{\mathcal{D}_{1,\kappa}\}_{\kappa\in\mathbb{N}}$ are computationally indistinguishable if for all PPT adversaries A, $\Pr[A(\mathcal{D}_{0,\kappa})=1] - \Pr[A(\mathcal{D}_{1,\kappa})=1] = \mathsf{negl}(\kappa)$. We write this $\mathcal{D}_{0,\kappa} \underset{C}{\approx} \mathcal{D}_{1,\kappa}$.

B.2 Two-Round MPC

A *2-round MPC protocol* MPC for message-space $\{\mathcal{M}_\kappa\}_{\kappa\mathbb{N}}$ and functionality $\{\mathcal{F}_\kappa\}_{\kappa\in\mathbb{N}}$ where for each $\kappa\in\mathbb{N}$ each $f\in\mathcal{F}_\kappa$ is a function on $(\mathcal{M}_\kappa)^n$ for some n, consists of three algorithms with the following syntax:

- $\mathsf{RunRoundOne}(1^\kappa, 1^n, f, i, x)$: A PPT algorithm taking the security parameter κ, number of users n, a (description of a) function $f\in\mathcal{F}_\kappa$ of arity n, an index $i\in[n]$, an input $x\in\mathcal{M}_\kappa$, and outputting a first protocol message $\rho^{(1)}$ and secret $\mathfrak{s}$.
- $\mathsf{RunRoundTwo}(\mathfrak{s}, (\rho_1^{(1)},\ldots,\rho_n^{(1)}))$: A PPT algorithm taking a secret $\mathfrak{s}$ and the first protocol message for all n parties $\rho_1^{(1)},\ldots,\rho_n^{(1)}$, and outputting a second protocol message $\rho^{(2)}$.
- $\mathsf{ComputeResult}$: A PT algorithm taking as input the n second-round protocol messages $\rho_1^{(2)},\ldots,\rho_n^{(2)}$ for each party and outputting a value y.

Correctness

We say that MPC is *correct* if for all $\kappa, n, \in\mathbb{N}$, $\mathbf{x}_1\ldots\mathbf{x}_n\in\mathcal{M}_\kappa$ and $f\in\mathcal{F}_\kappa$

$$\Pr\left[y = f(\mathbf{x}) \;\middle|\; \begin{array}{ll} (\rho_i^{(1)}, \mathfrak{s}_i) \leftarrow\!\!\$\ \mathsf{RunRoundOne}(1^\kappa, 1^n, f, i, \mathbf{x}_i) & \forall i\in[n] \\ \rho_i^{(2)} \leftarrow\!\!\$\ \mathsf{RunRoundTwo}(\mathfrak{s}_i, (\rho_1^{(1)},\ldots,\rho_n^{(1)})) & \forall i\in[n] \\ y \leftarrow \mathsf{ComputeResult}(\rho_1^{(2)},\ldots,\rho_n^{(2)}) & \end{array} \right] = 1\ .$$

▶ Remark 6. The above definition of two-round MPC is without setup (*i.e.*, a CRS). We also consider the case that there is an additional algorithm CRSGen taking 1^κ and outputting a common reference string CRS that is input to the remaining algorithms. We call this two-round MPC *in the CRS model*.

We say that MPC is *unbounded* if the output of $\mathsf{RunRoundOne}$ does not depend on n. In this case, we input n to $\mathsf{RunRoundTwo}$ instead of $\mathsf{RunRoundOne}$. We call MPC *input-delayed* (resp. *function-delayed*) if the output of $\mathsf{RunRoundOne}$ does not depend on x (resp. f) but just on $1^{|x|}$ (resp. $1^{|f|}$). In this case, we input x (resp. f) to $\mathsf{RunRoundTwo}$ instead of $\mathsf{RunRoundOne}$. We call MPC *input-rerunnable* (resp. *function-rerunnable*) if it is *input-delayed* (resp. *function-delayed*) and if $\mathsf{RunRoundTwo}$ can be executed multiple times with different input choices (resp. function choices) while still preserving the security properties of the MPC protocol (see below).

Security

Let MPC be a 2-round MPC protocol as above. Let Coins be the coin-space for the protocol. For an adversary $\mathcal{A}=(\mathcal{A}_0,\mathcal{A}_1)$ and simulator $\mathcal{S}$, consider the experiments in Figure 5. We say that $\mathcal{A}$ is *passive* (aka. semi-honest) of $I=\emptyset$. We say that MPC is $\widetilde{\mathsf{SIM}}$-secure if for any PPT adversary $\mathcal{A}$ there is a stateful PPT simulator $\mathcal{S}=(\widetilde{\mathsf{CRSGen}}, \widetilde{\mathsf{Extract}}, \widetilde{\mathsf{Sim}})$ such that $\mathbf{REAL}_{\mathcal{A}}^{\mathsf{MPC}}(\cdot)$ and $\mathbf{IDEAL}_{\mathcal{A},\mathcal{S}}^{\mathsf{MPC}}(\cdot)$ are computationally indistinguishable. Note that

Experiment $\mathbf{REAL}^{\mathsf{MPC}}_{\mathcal{A}}(1^\kappa)$

(**Optional**) $\mathsf{crs} \leftarrow\!\!\$\ \mathsf{CRSGen}(1^\kappa)$
$(1^n, I, f, (x_i)_{i\notin I}) \leftarrow\!\!\$\ \mathcal{A}_0(1^\kappa)$
// $f \in \mathcal{F}_\kappa$ of arity n, $x_i \in \mathcal{M}_\kappa$
$((\rho_i^{(1)})_{i\in I}, \mathsf{st}) \leftarrow\!\!\$\ \mathcal{A}_1(n, I, f)$
For $i \notin I$ do:
$r_i \leftarrow\!\!\$\ \mathsf{Coins}(1^\kappa)$
$(\rho_i^{(1)}, \mathfrak{s}_i)$
$\leftarrow\!\!\$\ \mathsf{RunRoundOne}(1^\kappa, 1^n, f, i, x_i; r_i)$
For $i \notin I$ do:
$\rho_i^{(2)}$
$\leftarrow\!\!\$\ \mathsf{RunRoundTwo}(\mathfrak{s}_i, (\rho_1^{(1)}, \ldots, \rho_n^{(1)}); r_i)$
$\alpha \leftarrow\!\!\$\ \mathcal{A}_2(\mathsf{st}, (\rho_i^{(1)}, \rho_i^{(2)})_{i\notin I})$
Return α

Experiment $\mathbf{IDEAL}^{\mathsf{MPC}}_{\mathcal{A},\mathcal{S}}(1^\kappa)$

(**Optional**) $\mathsf{crs} \leftarrow\!\!\$\ \widetilde{\mathsf{CRSGen}}(1^\kappa)$
$(1^n, I, f, (x_i)_{i\notin I}) \leftarrow\!\!\$\ \mathcal{A}_0(1^\kappa)$
// $f \in \mathcal{F}_\kappa$ of arity n, $x_i \in \mathcal{M}_\kappa$
$((\rho_i^{(1)})_{i\in I}, \mathsf{st}) \leftarrow\!\!\$\ \mathcal{A}_1(n, I, f)$
$x_i \leftarrow \widetilde{\mathsf{Extract}}(\rho_i^{(1)}) \quad \forall i \in I$
$(\rho_i^{(1)}, \rho_i^{(2)})_{i\notin I}$
$\leftarrow\!\!\$\ \widetilde{\mathsf{Sim}}((x_i)_{i\in I}, f(x_1, \ldots, x_n))$
$\alpha \leftarrow\!\!\$\ \mathcal{A}_2(\mathsf{st}, (\rho_i^{(1)}, \rho_i^{(2)})_{i\notin I})$
Return α

Figure 5 Experiments for SIM-security of two-round MPC.

simulation of the first-round protocol messages for the honest parties are independent of the inputs, so for convenience and ease of presentation we will partition the algorithm $\widetilde{\mathsf{Sim}}$ into two algorithms: $\widetilde{\mathsf{Sim}_1}$ and $\widetilde{\mathsf{Sim}_2}$, defined as follows:

- $\widetilde{\mathsf{Sim}_1}() \mapsto (\rho_i^{(1)})_{i\notin I}$: Outputs the first-round protocol messages for the honest parties.
- $\widetilde{\mathsf{Sim}_2}((x_i)_{i\in I}, y) \mapsto (\rho_i^{(2)})_{i\notin I}$: On input the inputs of the corrupted parties along with the target output value of the protocol, $\widetilde{\mathsf{Sim}_2}$ outputs the second-round protocol messages for the honest parties.

Input/Function-Rerunnability

For simplicity, the definition in Figure 5 does not capture input/function-rerunnability. It is straightforward to see how the definition can be extended. For example in the case of function-rerunnability (the situation is analogous for input-rerunnability where inputs and functions are swapped), the changes to the definition are (1) $\mathcal{A}_0$ outputs a set of functions $\{f_i\}$ instead of a single function, (2) in the real experiment $\mathsf{RunRoundTwo}$ is executed for each function, (3) in the ideal experiement $\widetilde{\mathsf{Sim}_2}$ is called for each function, and (4) the complete set of second-round protocol messages for all functions is given to $\mathcal{A}_2$.

Input Extractability

Our results in this work rely on the simulator's ability to extract the inputs of the corrupted parties, hence the need for the $\widetilde{\mathsf{Extract}}$ algorithm. In the semi-honest setting, extraction is not necessary. In the malicious case, both known constructions of two-round MPC [33, 57] for general functions satisfy the above extractability property, albeit in the CRS model.

B.3 Punctured Pseudorandom Functions

A PRF F is specified by two algorithms:

- $\mathsf{PRF.Setup}(1^\kappa)$: The setup algorithm takes as input the security parameter and outputs a description of the key space $\mathcal{K}_\kappa$, domain $\mathcal{X}$, range $\mathcal{Y}$ as well as the PRF key K.
- $\mathsf{PRF.Eval}(K, x)$: The eval algorithm takes a key $K \in \mathcal{K}_\kappa$ and domain point $x \in \mathcal{X}_\kappa$ and outputs a range point $y \in \mathcal{Y}_\kappa$.

We require that for all adversaries $\mathcal{A}$

$$\Pr\left[\mathcal{A}^{\mathsf{PRF.Eval}(K,\cdot)} \text{ outputs } 1\right] - \Pr\left[\mathcal{A}^{\$(\cdot)} \text{ outputs } 1\right]$$

is negligible in κ, where $K \leftarrow \mathsf{PRF.Setup}(1^\kappa)$ and $\$(\cdot)$ denotes a random function from $\mathcal{X}_\kappa$ to $\mathcal{Y}_\kappa$.

A punctureable PRF additionally includes an algorithm $\mathsf{PRF.Punc}$ which takes as input a PRF key K and a point $x^* \in \mathcal{X}$ and outputs a punctured key K_{x^*}. For correctness, we require that $\mathsf{PRF.Eval}(K_{x^*}, x) = \mathsf{PRF.Eval}(K, x)$ for all $x \neq x^*$ and $\perp$ when $x = x^*$.

Security of punctured PRF

The security game between the challenger and the adversary A consists of the following four phases.

Setup Phase: The challenger samples a PRF key K and a random bit b.

Evaluation Query Phase: The adversary $\mathcal{A}$ queries for polynomially many evaluations. For each evaluation query x, the challenger sends $\mathsf{F}(K, x)$ to $\mathcal{A}$.

Challenge Phase: $\mathcal{A}$ chooses a challenge x^* and the challenger computes $K_{x^*} \leftarrow \mathsf{PRF.Punc}(K, x^*)$. If $b = 0$, the challenger outputs K_{x^*} and $\mathsf{F}(K, x^*)$. Else, the challenger outputs K_{x^*} and $y \leftarrow\!\!\$\ \mathcal{Y}$ chosen uniformly at random.

Guess: The adversary $\mathcal{A}$ outputs a guess b' of b.

The adversary $\mathcal{A}$ wins if $b' = b$ and the adversary did not query for evaluation on x^*. The advantage of $\mathcal{A}$ is defined to be

$$\mathsf{Adv}^{\mathsf{F}}_{\mathcal{A}}(1^\kappa) = |\Pr[\mathcal{A} \text{ wins}] - 1/2|$$

The PRF F is a secure puncturable PRF if for all probabilistic polynomial time adversaries $\mathcal{A}$, we have that $\mathsf{Adv}^{\mathsf{F}}_{\mathcal{A}}(1^\kappa)$ is negligible in κ.

B.4 Multi-Input Functional Encryption

An *n-input FE scheme* [45] MIFE for a message space $\{\mathcal{M}_\kappa\}_{\kappa\in\mathbb{N}}$ and a functionality $\{\mathcal{F}_\kappa\}_{\kappa\in\mathbb{N}}$, where for each $\kappa \in \mathbb{N}$, each $f \in \mathcal{F}_\kappa$ is a (description of a) function on $(\mathcal{M}_\kappa)^n$, is given by a set of algorithms with the following syntax:

- $\mathsf{MIFE.Setup}(1^\kappa, 1^n)$: A PPT algorithm taking the security parameter κ and number of users n, and outputting the master secret key MSK and encryption keys $(\mathsf{EK}_1, \ldots, \mathsf{EK}_n)$.
- $\mathsf{MIFE.KeyGen}(\mathsf{MSK}, f)$: A PT algorithm taking a master secret key MSK, a function $f \in \mathcal{F}_\kappa$ and outputting a corresponding decryption key DK_f.
- $\mathsf{MIFE.Enc}(\mathsf{EK}, \mathbf{x})$: A PPT algorithm taking an encryption key EK and a message $\mathbf{x} \in \mathcal{M}_\kappa$, and outputting a ciphertext c.
- $\mathsf{MIFE.Dec}(\mathsf{DK}_f, (c_1, \ldots, c_n))$: A PT algorithm taking decryption key DK_f and vector of ciphertexts $(c_1, \ldots, c_n)$, and outputting a string y.

Correctness

We say that MIFE is *correct* if for all $\kappa, \in \mathbb{N}$, $\mathbf{x}_1 \dots \mathbf{x}_n \in \mathcal{M}_\kappa$ and $f \in \mathcal{F}_\kappa$

$$\Pr\left[y = f(\mathbf{x}_1, \cdots, \mathbf{x}_n) \middle| \begin{array}{l} ((\mathsf{EK}_1, \dots, \mathsf{EK}_n), \mathsf{MSK}) \leftarrow\!\!\$\ \mathsf{MIFE.Setup}(1^\kappa) \\ c_i \leftarrow\!\!\$\ \mathsf{MIFE.Enc}(\mathsf{EK}_i, \mathbf{x}_i) \quad \forall i \in [n] \\ \mathsf{DK}_f \leftarrow\!\!\$\ \mathsf{MIFE.KeyGen}(\mathsf{MSK}, f) \\ y \leftarrow \mathsf{MIFE.Dec}(\mathsf{DK}_f, (c_1, \dots, c_n)) \end{array} \right] = 1 \ .$$

▶ Remark 7. We remark that our formulation of MIFE assumes that the senders (as in our application an encryptor is referred to as a sender or source) are ordered. This allows for consistency with our formulation of ad hoc MIFE and allows us to simplify exposition versus [45].

Indistinguishability-Based Security

For an n-input FE scheme MIFE as above and adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2)$, consider the experiment in Figure 6.

Experiment $\mathbf{IND}^{\mathsf{MIFE}}_{\mathcal{A}}(1^\kappa)$
$(I, st) \leftarrow\!\!\$\ \mathcal{A}_0(1^\kappa)$
$b \leftarrow\!\!\$\ \{0,1\}$
$((\mathsf{EK}_1, \dots, \mathsf{EK}_n), \mathsf{MSK})$
$\leftarrow\!\!\$\ \mathsf{MIFE.Setup}(1^\kappa)$
$st \leftarrow\!\!\$\ \mathcal{A}_1^{\mathcal{O}_{\mathsf{enc}}(\cdot,\cdot,\cdot), \mathcal{O}_{\mathsf{kg}}(\cdot)}(st)$
$b' \leftarrow\!\!\$\ \mathcal{A}_2^{\mathcal{O}_{\mathsf{enc}}(\cdot,\cdot,\cdot), \mathcal{O}_{\mathsf{kg}}(\cdot)}((\mathsf{EK}_i)_{i \in I}, st)$
Return $(b = b')$

Oracle $\mathcal{O}_{\mathsf{enc}}(i, \mathbf{x}_0, \mathbf{x}_1)$
If $\mathbf{x}_0, \mathbf{x}_1 \in \mathcal{M}_\kappa$ and $|\mathbf{x}_0| = |\mathbf{x}_1|$
$c_b \leftarrow\!\!\$\ \mathsf{MIFE.Enc}(\mathsf{EK}_i, \mathbf{x}_b)$
Return c_b
Else Return $\perp$

Oracle $\mathcal{O}_{\mathsf{kg}}(f)$
If $f \in \mathcal{F}_\kappa$
$\mathsf{DK}_f \leftarrow\!\!\$\ \mathsf{MIFE.KeyGen}(\mathsf{MSK}, f)$
Return DK_f
Else return $\perp$

Figure 6 Experiment for IND-security of standard MIFE.

We call $\mathcal{A}$ *legitimate* if for all $\kappa \in \mathbb{N}$, in all transcripts $\mathbf{IND}^{\mathsf{MIFE}}_{\mathcal{A}}(1^\kappa)$ it holds that for every key generation query f there does not exist two sequences $(y_{1,0}, \cdots, y_{n,0})$ and $(y_{1,1}, \cdots, y_{n,1})$ such that

$$f(y_{1,0}, \cdots, y_{n,0}) \neq f(y_{1,1}, \cdots, y_{n,1})$$

and for every $j \in [n]$

- $j \in I$, *i.e.* j is corrupted (so there is no restriction on $y_{j,0}, y_{j,1}$ above), or
- there is an encryption query $(j, \mathbf{x}_0, \mathbf{x}_1)$ such that $y_{j,0} = \mathbf{x}_0$ and $y_{j,1} = \mathbf{x}_1$.

We assume adversaries are legitimate unless otherwise stated. We call $\mathcal{A}$ *passive* if $I = \emptyset$. We call $\mathcal{A}$ *selective* if $\mathcal{A}_2$ makes no queries. We say that MIFE is IND-secure if for any adversary $\mathcal{A}$

$$|\Pr\left[\mathbf{IND}^{\mathsf{MIFE}}_{\mathcal{A}}(\cdot) \text{ outputs } 1\right] - 1/2| = \mathsf{negl}(\cdot) \ .$$

Simulation-Based Security

For an MIFE scheme MIFE as above, adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2)$, and simulator $\mathcal{S} = (\widetilde{\mathsf{Setup}}, \widetilde{\mathsf{Enc}}, \widetilde{\mathsf{KeyGen}})$, consider the experiments in Figure 7. Here $q_{\mathsf{enc},i}$ is the number of encryption queries for user $i \in [n]$. We say that $\mathcal{A}$ is *q-bounded* if $q_{\mathsf{enc},i} \leq q$ for all $i \in [n]$. We say that MIFE is SEL-SIM-secure if for any adversary $\mathcal{A}$ there is a simulator $\mathcal{S}$ such that $\mathbf{REAL}^{\mathsf{MIFE}}_{\mathcal{A}}(\cdot)$ and $\mathbf{IDEAL}^{\mathsf{MIFE}}_{\mathcal{A},\mathcal{S}}(\cdot)$ are computationally indistinguishable.

▶ Remark 8. For consistency with our formulation of simulation-based security for ad hoc MIFE, we restrict the above definition to the selective-security and passive (no sender corruption) setting.

Experiment $\mathbf{REAL}^{\mathsf{MIFE}}_{\mathcal{A}}(1^\kappa)$
$((\mathsf{EK}_1, \ldots, \mathsf{EK}_n), \mathsf{MSK}) \leftarrow\!\!\$\ \mathsf{MIFE.Setup}(1^\kappa)$
$(m_{i,j})_{i\in[n], j\in[q_{\mathsf{enc},i}]} \leftarrow\!\!\$\ \mathcal{A}_1(1^\kappa)$
For all $i \in [n], j \in [q_{\mathsf{enc},i}]$ do:
  $c_{i,j} \leftarrow\!\!\$\ \mathsf{MIFE.Enc}(\mathsf{MSK}_i, m_{i,j})$
$\alpha \leftarrow\!\!\$\ \mathcal{A}_2^{\mathcal{O}_{\mathsf{kg}}(\cdot)}((\mathsf{EK}_i)_{i\in[n]}, (c_{i,j})_{i\in[n],j\in[q_{\mathsf{enc},i}]})$
Return $((m_{i,j})_{i\in[n],j\in[q_{\mathsf{enc},i}]}, \alpha)$

Oracle $\mathcal{O}_{\mathsf{kg}}(f)$ // $f \in \mathcal{F}_\kappa$
$\mathsf{DK}_f \leftarrow\!\!\$\ \mathsf{MIFE.KeyGen}(\mathsf{MSK}, f)$
Return DK_f

Experiment $\mathbf{IDEAL}^{\mathsf{MIFE}}_{\mathcal{A},\mathcal{S}}(1^\kappa)$
$((\widetilde{\mathsf{EK}_1}, \ldots, \widetilde{\mathsf{EK}_n}), \widetilde{\mathsf{MSK}}) \leftarrow\!\!\$\ \widetilde{\mathsf{Setup}}(1^\kappa)$
$(m_{i,j})_{i\in[n], j\in[q_{\mathsf{enc},i}]} \leftarrow\!\!\$\ \mathcal{A}_1(st)$
For all $i \in [n], j \in [q_{\mathsf{enc},i}]$ do:
  $\widetilde{c_{i,j}} \leftarrow\!\!\$\ \widetilde{\mathsf{Enc}}(\widetilde{\mathsf{MSK}}, |m_{i,j}|)$
$\alpha \leftarrow\!\!\$\ \mathcal{A}_2^{\mathcal{O}_{\mathsf{kg}}(\cdot,\cdot,\cdot)}((\widetilde{\mathsf{EK}_i})_{i\in[n]}, (\widetilde{c_{i,j}}))$
Return $((m_{i,j})_{i\in[n],j\in[q_{\mathsf{enc},i}]}, \alpha)$

Oracle $\widetilde{\mathcal{O}_{\mathsf{kg}}}(f)$ // $f \in \mathcal{F}_\kappa$
$\widetilde{\mathsf{DK}_f} \leftarrow \widetilde{\mathsf{KeyGen}}(f, (f(m_{i,j_i}))_{j_i\in[q_{\mathsf{enc},i}]}, \widetilde{\mathsf{MSK}})$
Return $\widetilde{\mathsf{DK}_f}$

Figure 7 Experiments for SIM-security of standard MIFE.

B.5 Function-Private Functional Encryption

A functional encryption scheme, denoted as FE [20], is a tuple of algorithms $\mathsf{FE} = (\mathsf{FE.Setup}, \mathsf{FE.KeyGen}, \mathsf{FE.Enc}, \mathsf{FE.Dec})$ for a message space $\{\mathcal{M}_\kappa\}_{\kappa\in\mathbb{N}}$ and a functionality $\{\mathcal{F}_\kappa\}_{\kappa\in\mathbb{N}}$, where for each $\kappa \in \mathbb{N}$, each $f \in \mathcal{F}_\kappa$ is a (description of a) function on $\mathcal{M}_\kappa$. The syntax is the same as for a 1-input MIFE scheme where $\mathsf{EK}_1 = \mathsf{MSK}$ and $I = \emptyset$. The correctness requirement remains the same, as well the notion of indistinguishability based security (which we refer to as "message privacy").

Function Privacy

We additionally define the notion of function privacy as follows. For an FE scheme FE as above and adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2)$, consider the experiment in Figure 8.

We call $\mathcal{A}$ *legitimate* if for all $\kappa \in \mathbb{N}$, in all transcripts $\mathbf{FP}^{\mathsf{FE}}_{\mathcal{A}}(1^\kappa)$ it holds that for every key generation query f_0, f_1 there does not exist an encryption query $\mathbf{x} \in \mathcal{M}_\kappa$ such that

$$f_0(\mathbf{x}) \neq f_1(\mathbf{x}) .$$

We assume adversaries are legitimate unless otherwise stated. We say that FE is FP-secure if for any adversary $\mathcal{A}$

$$|\Pr\left[\mathbf{FP}^{\mathsf{FE}}_{\mathcal{A}}(\cdot) \text{ outputs } 1\right] - 1/2| = \mathsf{negl}(\cdot) .$$

Experiment $\mathbf{FP}^{\mathsf{FE}}_{\mathcal{A}}(1^\kappa)$
$b \leftarrow\!\!\$\ \{0,1\}$
$\mathsf{MSK} \leftarrow\!\!\$\ \mathsf{FE.Setup}(1^\kappa)$
$b' \leftarrow\!\!\$\ \mathcal{A}_2^{\mathcal{O}_{\mathsf{enc}}(\cdot),\mathcal{O}_{\mathsf{kg}}(\cdot,\cdot)}(1^\kappa)$
Return $(b = b')$

Oracle $\mathcal{O}_{\mathsf{enc}}(\mathbf{x})$
If $\mathbf{x} \in \mathcal{M}_\kappa$
$c \leftarrow\!\!\$\ \mathsf{MIFE.Enc}(\mathsf{MSK}, \mathbf{x})$
Return c
Else Return $\perp$

Oracle $\mathcal{O}_{\mathsf{kg}}(f_0, f_1)$
If $f_0, f_1 \in \mathcal{F}_\kappa$
$\mathsf{DK}_{f_b} \leftarrow\!\!\$\ \mathsf{MIFE.KeyGen}(\mathsf{MSK}, f_b)$
Return DK_{f_b}
Else return $\perp$

Figure 8 Experiment for FP-security of FE.

C Simulation Based Security for Ad Hoc MIFE

We now present a simulation-based definition of security. The definition has a number of restrictions that we justify below.

For an ad hoc MIFE scheme aMIFE as above, adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2)$, and simulator $\mathcal{S} = (\widetilde{\mathsf{Setup}}, \widetilde{\mathsf{Enc}}, \widetilde{\mathsf{KeyGen}})$, consider the experiments in Figure 9. Here $q_{E,i}$ is the number of encryption queries for user $i \in [n]$. We say that $\mathcal{A}$ is q-*bounded* if $q_{E,i} \le q$ for all $i \in [n]$.

Experiment $\mathbf{REAL}^{\mathsf{aMIFE}}_{\mathcal{A}}(1^\kappa)$
$(1^n, st) \leftarrow\!\!\$\ \mathcal{A}_0(1^\kappa)$
For all $i \in [n]$ do:
$(\mathsf{MSK}_i, \mathsf{PP}_i) \leftarrow\!\!\$\ \mathsf{aMIFE.Setup}(1^\kappa)$
$(m_{i,j})_{i\in[n],j\in[q_{E,i}]} \leftarrow\!\!\$\ \mathcal{A}_1(st)$
For all $i \in [n], j \in [q_{E,i}]$ do:
$c_{i,j} \leftarrow\!\!\$\ \mathsf{aMIFE.Enc}(\mathsf{MSK}_i, m_{i,j})$
$\alpha \leftarrow\!\!\$\ \mathcal{A}_2^{\mathcal{O}_{\mathsf{kg}}(\cdot,\cdot,\cdot)}((\mathsf{PP}_i)_{i\in[n]}, (c_{i,j}))$
Return $(1^n, (m_{i,j})_{i\in[n],j\in[q_{E,i}]}, \alpha)$

Experiment $\mathbf{IDEAL}^{\mathsf{aMIFE}}_{\mathcal{A},\mathcal{S}}(1^\kappa)$
$(1^n, st) \leftarrow\!\!\$\ \mathcal{A}_0(1^\kappa)$ // $I \subset [n]$
For all $i \in [n]$ do:
$(\widetilde{\mathsf{MSK}}_i, \widetilde{\mathsf{PP}}_i) \leftarrow\!\!\$\ \widetilde{\mathsf{Setup}}(1^\kappa)$
$(m_{i,j})_{i\in[n],j\in[q_{E,i}]} \leftarrow\!\!\$\ \mathcal{A}_1(st)$
For all $i \in [n], j \in [q_{E,i}]$ do:
$\widetilde{c_{i,j}} \leftarrow\!\!\$\ \widetilde{\mathsf{Enc}}(\widetilde{\mathsf{MSK}}_i, |m_{i,j}|)$
$\alpha \leftarrow\!\!\$\ \mathcal{A}_2^{\mathcal{O}_{\mathsf{kg}}(\cdot,\cdot,\cdot)}((\widetilde{\mathsf{PP}}_i)_{i\in[n]}, (\widetilde{c_{i,j}}))$
Return $(1^n, (m_{i,j})_{i\in[n],j\in[q_{E,i}]}, \alpha)$

Oracle $\mathcal{O}_{\mathsf{kg}}(i, (i_1, \ldots, i_\ell), f)$
If $i \in \{i_1, \cdots, i_\ell\}$, and $f \in \mathcal{F}_\kappa$ of arity ℓ
$\mathsf{PDK}_{i,f} \leftarrow$
$\mathsf{aMIFE.KeyGen}(i, \mathsf{MSK}_i, (\mathsf{PP}_{i_j})_{j\in[\ell]}, f)$
Return $\mathsf{PDK}_{i,f}$
Else return $\perp$

Oracle $\widetilde{\mathcal{O}_{\mathsf{kg}}}(i, (i_1, \ldots, i_\ell), f)$
If $i \in \{i_1, \cdots, i_\ell\}$, and $f \in \mathcal{F}_\kappa$ of arity ℓ
$\widetilde{q} \leftarrow (i, (i_1, \ldots, i_\ell), f)$
$\widetilde{\mathsf{PDK}_{i,f}} \leftarrow$
$\widetilde{\mathsf{KeyGen}}(\widetilde{q}, (f(m_{i_k,j_{i_k}}))_{j_{i_k}\in[q_{E,i_k}]}, (\widetilde{\mathsf{MSK}_{i_j}}))$
Return $\widetilde{\mathsf{PDK}_{i,f}}$
Else return $\perp$

Figure 9 Experiments for SIM-security of ad hoc MIFE.

We say that aMIFE is xxx-SEL-SIM-secure if for any adversary $\mathcal{A}$ of type xxx there is a simulator $\mathcal{S}$ such that $\mathbf{REAL}^{\mathsf{aMIFE}}_{\mathcal{A}}(\cdot)$ and $\mathbf{IDEAL}^{\mathsf{aMIFE}}_{\mathcal{A},\mathcal{S}}(\cdot)$ are computationally indistinguishable.

▶ Remark 9. The above definition has a number of restrictions that we now justify:

- The adversary is passive and does not corrupt any sender. This is justified because otherwise such a scheme implies virtual black-box obfuscation as in the case of standard MIFE [45], which is impossible [12].

- The adversary is selective and chooses its challenge messages before seeing the public parameters of the users. We focus on this formulation for simplicity and leave the study of adaptive security for ad hoc MIFE in the case of simulation-based security for future work.
- The simulator is black-box. This is for simplicity as it is stronger than allowing non-black-box simulation and our constructions achieve it.

D Proof of Theorem 5

▶ **Theorem 5.** *If* MIFE *is a IND-secure MIFE scheme and* MPC *is a SIM-secure 2-round MPC protocol, then our construction is* sel*-IND-secure.*

Proof. The proof of security makes use of a trapdoor data structure which is defined in Figure 10.

The trapdoor data structure

Here, mode is used to indicate whether we are in the real mode Real or trapdoor mode Trap. CT indicates the hardwired MIFE CT which must be output if the field index equals the counter ctr set in the FE key. The fields val_0 and val_1 are used to indicate the values corresponding to bit 0 and bit 1 respectively, where the latter is used when index $>$ ctr and the former when index $<$ ctr.

mode	CT	index	val_0	val_1

Figure 10 Data Structure Trap used for Proof.

The Hybrids

We prove the theorem via a hybrid argument. We describe our hybrids below.

Hybrid 0: This is the real game in which on every encryption query $(i, \mathbf{x}_0, \mathbf{x}_1)$, $\mathbf{x}_0$ is encrypted.

Suppose there are Q_c encryption queries (made selectively). For each $k \in [Q_c]$, let i be the party index queried, $\mathbf{x}_0$ and $\mathbf{x}_1$ the challenge plaintexts, let T be the tag used during encryption and I be the set of users corrupted by the adversary. We use these definitions in the remainder of the proof.

Hybrid 1: The change in this hybrid is twofold.

1. **Simulate the Public Parameters.** We set the first-round protocol message $\rho_i^{(1)}$ for MPC in the public parameters to the output of the simulator $\widetilde{\mathsf{Sim}}_1$ for each uncorrupted user $i \notin I$.
2. **Simulate the Function Key.** For each key generation query $(i, (i_1, \ldots, i_\ell), f)$, we do the following.
   a. Let $J \triangleq I \cap \{i_1, \ldots, i_\ell\}$ be the subset of corrupted users and $\bar{J} = \{i_1, \ldots, i_\ell\} \setminus J$ be the subset of honest users.
   b. We use the simulator's $\widetilde{\mathsf{Extract}}$ algorithm to compute $\mathbf{x}_j \leftarrow \widetilde{\mathsf{Extract}}(\rho_j^{(1)})$ for each corrupted party $j \in J$ where $\mathsf{PP}_j = \rho_j^{(1)}$, then compute $y = \mathsf{GenKeys}_f(\mathbf{x}_{i_1}, \ldots, \mathbf{x}_{i_\ell})$ where $\mathbf{x}_j$ for $j \in \bar{J}$ is (honest) party j's input to the MPC protocol.

c. Compute $(\rho_j^{(2)})_{j\in\bar{I}} \leftarrow \widetilde{\mathsf{Sim}_2}((x_j)_{j\in J}, y)$ and $s \leftarrow \mathsf{PRF.Eval}(K_i, 0 \,\|\, (\mathsf{PP}_{i_j})_{j\in[\ell]} \| f)$. Return $(\rho_i^{(2)}, s)$.

See Figure 11 for a formal description.

Indistinguishability of the hybrids follows from the SIM-security of the MPC protocol. As we see in Figure 11, the only difference from Hybrid 0 is that the inputs of the corrupt parties are extracted using the $\widetilde{\mathsf{Extract}}$ algorithm and the protocol transcript is generated using the MPC simulator. Hence, an adversary who distinguishes between Hybrids 0 and 1 implies an adversary against the MPC protocol by a standard reduction.

Hybrid 1:
$(1^n, I, (\mathsf{PP}_i)_{i\in I}, st) \leftarrow\!\!\$\ \mathcal{A}_0(1^\kappa)$
$b \leftarrow\!\!\$\ \{0,1\}$
$\forall i \notin I$:
$(\mathsf{PP}_{\mathsf{FE}_i}, \mathsf{MSK}_{\mathsf{FE}_i}) \leftarrow\!\!\$\ \mathsf{FE.Setup}(1^\kappa)$
$K_i \leftarrow\!\!\$\ \mathcal{K}_\kappa$
$K_i^{\mathsf{punc}} \leftarrow\!\!\$\ \mathcal{K}_\kappa$
$(\rho_i^{(1)})_{i\notin I} \leftarrow\!\!\$\ \widetilde{\mathsf{Sim}_1}()$
$\mathsf{PP}_i \leftarrow \rho_i^{(1)}$

$st \leftarrow\!\!\$\ \mathcal{A}_1^{\mathcal{O}_{\mathsf{enc}}(\cdot,\cdot,\cdot),\mathcal{O}_{\mathsf{kg}}(\cdot,\cdot,\cdot)}(st)$
$b' \leftarrow\!\!\$\ \mathcal{A}_2^{\mathcal{O}_{\mathsf{enc}}(\cdot,\cdot,\cdot),\mathcal{O}_{\mathsf{kg}}(\cdot,\cdot,\cdot)}((\mathsf{PP}_i)_{i\notin I}, st)$
Return $(b = b')$

Oracle $\mathcal{O}_{\mathsf{enc}}(i, \mathbf{x}_0, \mathbf{x}_1)$
$T \leftarrow\!\!\$\ \{0,1\}^\kappa$
Return $\mathsf{FE.Enc}(\mathsf{MSK}_{\mathsf{FE}_i}, (\mathbf{x}_0, T, K_i^{\mathsf{punc}}, \mathsf{Trap}))$

Oracle $\mathcal{O}_{\mathsf{kg}}(i, (i_1, \ldots, i_\ell), f)$
$J \leftarrow I \cap \{i_1, \ldots, i_\ell\}$; $\bar{J} \leftarrow \{i_1, \ldots, i_\ell\} \setminus J$
$(K_j, \mathsf{MSK}_{\mathsf{FE}_j}) \leftarrow \widetilde{\mathsf{Extract}}(\rho_j^{(1)}) \quad \forall j \in J$
$y \leftarrow \mathsf{GenKeys}_f((K_{i_1}, \mathsf{MSK}_{\mathsf{FE}_{i_1}}), \ldots, (K_{i_\ell}, \mathsf{MSK}_{\mathsf{FE}_{i_\ell}}))$
$(\rho_j^{(2)})_{j\in\bar{J}} \leftarrow\!\!\$\ \widetilde{\mathsf{Sim}_2}(K_j, \mathsf{MSK}_{\mathsf{FE}_j})_{j\in J}, y)$
$s \leftarrow \mathsf{PRF.Eval}(K_i, 0 \,\|\, (\mathsf{PP}_{i_j})_{j\in[\ell]} \,\|\, f)$
Return $(\rho_i^{(2)}, s)$.

Figure 11 Hybrid 1.

Hybrid 2: In this hybrid, we replace the outputs of the PRF on key K_i for the honest users i with uniformly random strings in the function $\mathsf{GenKeys}$ described in Figure 2. For every key query pertaining to parties $(i_1, \ldots, i_\ell)$ and function f and every honest party i, we replace $\mathsf{PRF.Eval}(K_i, k \,\|\, (\mathsf{PP}_{i_j})_{j\in[\ell]} \,\|\, f)$ for $k \in \{0,1,2\}$ as in Hybrid 1 with a fresh uniformly random string. The changes are formally described in Figure 12 wherein the algorithm R is used to generate random strings and keep track of those previously generated.

Indistinguishability of Hybrid 1 and Hybrid 2 follows from the security of the PRF. More precisely, a standard argument iterates through sub-hybrids for each honest party, replacing the PRF outputs with uniformly random strings.

Hybrid 3: In this hybrid, we change how y (the target output passed to $\widetilde{\mathsf{Sim2}}$) is generated in each query $((i_1, \ldots, i_\ell), f)$. In this hybrid, y is randomly sampled. Furthermore for a pair $((i_1, \ldots, i_\ell), f)$ that is *fully queried* i.e. a partial decryption query $(i, (i_1, \ldots, i_\ell), f)$ is made for each $i \in \{i_1, \ldots, i_\ell\} \setminus I$, the final partial decryption key that is issued has its masking value, i.e. the second component of the partial decryption key, generated differently. It is generated as

$$s \leftarrow y \oplus S_1 \oplus S_2 \oplus \mathsf{GenKeys}''((r'_1, \mathsf{MSK}_{\mathsf{FE}_1}), \ldots, (r'_\ell, \mathsf{MSK}_{\mathsf{FE}_\ell}))$$

Here S_1 and S_2 are computed so as to satisfy requisite dependencies. Figure 13 captures these changes formally.

Hybrid 2:

$(1^n, I, (\mathsf{PP}_i)_{i\in I}, st) \leftarrow\!\!\$\ \mathcal{A}_0(1^\kappa)$
$b \leftarrow\!\!\$\ \{0,1\}$
$\forall i \notin I$:
$(\mathsf{PP}_{\mathsf{FE}_i}, \mathsf{MSK}_{\mathsf{FE}_i}) \leftarrow\!\!\$\ \mathsf{FE.Setup}(1^\kappa)$
~~$K_i \leftarrow\!\!\$\ \mathcal{K}_\kappa$~~
$K_i^{\mathsf{punc}} \leftarrow\!\!\$\ \mathcal{K}_\kappa$
$\Gamma \leftarrow \emptyset$
$(\rho_i^{(1)})_{i\notin I} \leftarrow\!\!\$\ \widetilde{\mathsf{Sim}_1}()$
$\mathsf{PP}_i \leftarrow \rho_i^{(1)}$

$st \leftarrow\!\!\$\ \mathcal{A}_1^{\mathcal{O}_{\mathsf{enc}}(\cdot,\cdot,\cdot),\mathcal{O}_{\mathsf{kg}}(\cdot,\cdot,\cdot)}(st)$
$b' \leftarrow\!\!\$\ \mathcal{A}_2^{\mathcal{O}_{\mathsf{enc}}(\cdot,\cdot,\cdot),\mathcal{O}_{\mathsf{kg}}(\cdot,\cdot,\cdot)}((\mathsf{PP}_i)_{i\notin I}, st)$
Return $(b = b')$

Oracle $\mathcal{O}_{\mathsf{enc}}(i, \mathbf{x}_0, \mathbf{x}_1)$
$T \leftarrow\!\!\$\ \{0,1\}^\kappa$
Return $\mathsf{FE.Enc}(\mathsf{MSK}_{\mathsf{FE}_i}, (\mathbf{x}_0, T, K_i^{\mathsf{punc}}, \mathsf{Trap}))$

Oracle $\mathcal{O}_{\mathsf{kg}}(i, (i_1, \ldots, i_\ell), f)$
$J \leftarrow I \cap \{i_1, \ldots, i_\ell\}$; $\bar{J} \leftarrow \{i_1, \ldots, i_\ell\} \setminus J$
$(K_j, \mathsf{MSK}_{\mathsf{FE}_j}) \leftarrow \widetilde{\mathsf{Extract}}(\rho_j^{(1)}) \quad \forall j \in J$
$(\gamma_j^{(0)}, \gamma_j^{(1)}, \gamma_j^{(2)}) \leftarrow\!\!\$\ R_\Gamma(j, (i_1, \ldots, i_\ell), f) \quad \forall j \in \bar{J}$
$\forall m \in \{0,1,2\}, j \in J$:
$\gamma_j^{(m)} \leftarrow \mathsf{PRF.Eval}(K_j, m \,\|\, (\mathsf{PP}_{i_k})_{k\in[\ell]} \,\|\, f)$
$S \leftarrow \bigoplus_{j\in J\cup\bar{J}} \gamma_j^{(0)}$
$y \leftarrow S \oplus \mathsf{GenKeys}'(\{(r_k, r'_l, \mathsf{MSK}_{\mathsf{FE}_k})\}_{k\in[\ell]})$
$(\rho_j^{(2)})_{j\in\bar{J}} \leftarrow\!\!\$\ \widetilde{\mathsf{Sim}_2}(K_j, \mathsf{MSK}_{\mathsf{FE}_j})_{j\in J}, y)$
Return $(\rho_i^{(2)}, s := \gamma_i^{(0)})$

Algorithm $\mathsf{GenKeys}'(\{(r_k, r'_l, \mathsf{MSK}_{\mathsf{FE}_k})\}_{k\in[\ell]})$
$(\{\mathsf{EK}_k\}_{k\in[\ell]}, \mathsf{MSK}) \leftarrow \mathsf{MIFE.Setup}(1^\kappa; \bigoplus_{k\in[\ell]} r_k)$
$\mathsf{SK}_f \leftarrow \mathsf{MIFE.KeyGen}(\mathsf{MSK}, f)$
$\forall i \in [\ell]$:
$\mathsf{SK}_{\mathsf{FE}_i} \leftarrow \mathsf{FE.KeyGen}(\mathsf{MSK}_{\mathsf{FE}_i}, \mathsf{ReEnc}_{\mathsf{EK}_i,\perp}; r'_i)$
Return $(\mathsf{SK}_f, \mathsf{SK}_{\mathsf{FE}_1}, \ldots, \mathsf{SK}_{\mathsf{FE}_\ell})$

Algorithm $R_\Gamma(i, (i_1, \ldots, i_\ell, f)$
If $(i, (i_1, \ldots, i_\ell), f, \gamma^{(0)}, \gamma^{(1)}, \gamma^{(2)}) \in \Gamma$
Return $(\gamma^{(0)}, \gamma^{(1)}, \gamma^{(2)})$
Else
$\gamma^{(j)} \leftarrow\!\!\$\ \{0,1\}^{\mathsf{rp}} \quad \forall j \in \{0,1,2\}$
// where rp is the range of the PRF
$\Gamma \leftarrow \Gamma \cup \{(i, (i_1, \ldots, i_\ell), f, \gamma^{(0)}, \gamma^{(1)}, \gamma^{(2)})\}$
Return $(\gamma^{(0)}, \gamma^{(1)}, \gamma^{(2)})$

Figure 12 Hybrid 2.

Hybrid 2 and Hybrid 3 are distributed identically. First, y is distributed uniformly in both hybrids for all group-function pairs $((i_1, \ldots, i_\ell), f)$ that are not *fully queried*. In the case of a group-function pair $((i_1, \ldots, i_\ell), f)$ that is fully queried, each partial decryption key $(\rho_i^{(2)}, s_i)$ for $i \in \{i_1, \ldots, i_\ell\}$ is such that

$$y \oplus \bigoplus_{i\in[\ell]} s_i = \mathsf{GenKeys}''((r'_1, \mathsf{MSK}_{\mathsf{FE}_1}), \ldots, (r'_\ell, \mathsf{MSK}_{\mathsf{FE}_\ell}))$$

This is distributed the same as the output of $\mathsf{GenKeys}'$ in the previous hybrid. Hence the y values are distributed identically in both hybrids.

Hybrid 4: Let Q_k be the number of subset-function pairs that are *fully queried*. In this hybrid, the key generation algorithm keeps track of the query number $\mathsf{ctr} \in [\mathsf{Q_k}]$ in the ad hoc MIFE function keys. In more detail, the algorithm $\mathsf{GenKeys}''$ invoked for query index j for all $j \in [Q_k]$ is modified to invoke ReEnc with parameter $\mathsf{ctr} = \mathsf{j}$ (please refer to Figure 3) instead of $\perp$.

We claim that by function hiding of FE, the two hybrids are indistinguishable. To see this, note that since $\mathsf{mode} = R$ in all the FE ciphertexts, changing the ctr value in the FE key has no effect on the decryption value obtained, since this field is only relevant in the trapdoor mode, i.e. when $\mathsf{mode} = T$. Hence, the decryption values for both keys remain exactly the same. Then, by security of FE, we have that Hybrids 3 and 4 are indistinguishable. The formal reduction is standard, and constructs everything except the FE ciphertexts and FE function keys as in the previous hybrid, which are obtained using the FE challenger.

Hybrid 3:
$(1^n, I, (\mathsf{PP}_i)_{i\in I}, st) \leftarrow\!\!\$\ \mathcal{A}_0(1^\kappa)$
$b \leftarrow\!\!\$\ \{0,1\}$
$\forall i \notin I$:
$K_i^{\mathsf{punc}} \leftarrow\!\!\$\ \mathcal{K}_\kappa$
$(\mathsf{PP}_{\mathsf{FE}_i}, \mathsf{MSK}_{\mathsf{FE}_i}) \leftarrow\!\!\$\ \mathsf{FE.Setup}(1^\kappa)$
$\Gamma \leftarrow \emptyset$
$Y \leftarrow \emptyset$
$Q \leftarrow \emptyset$
$(\rho_i^{(1)})_{i\notin I} \leftarrow\!\!\$\ \widetilde{\mathsf{Sim}_1}()$
$\mathsf{PP}_i \leftarrow \rho_i^{(1)} \quad \forall i \notin I$

$st \leftarrow\!\!\$\ \mathcal{A}_1^{\mathcal{O}_{\mathsf{enc}}(\cdot,\cdot,\cdot),\mathcal{O}_{\mathsf{kg}}(\cdot,\cdot,\cdot)}(st)$
$b' \leftarrow\!\!\$\ \mathcal{A}_2^{\mathcal{O}_{\mathsf{enc}}(\cdot,\cdot,\cdot),\mathcal{O}_{\mathsf{kg}}(\cdot,\cdot,\cdot)}((\mathsf{PP}_i)_{i\notin I}, st)$
Return $(b = b')$

Algorithm $\mathsf{GenKeys}''_f(\{(r'_k, \mathsf{MSK}_{\mathsf{FE}_k})\}_{k\in[\ell]})$
$((\mathsf{EK}_1, \ldots, \mathsf{EK}_\ell), \mathsf{MSK}) \leftarrow \mathsf{MIFE.Setup}(1^\kappa)$
$\mathsf{SK}_f \leftarrow \mathsf{MIFE.KeyGen}(\mathsf{MSK}, f)$
$\forall i \in [\ell]$, do:
$\mathsf{SK}_{\mathsf{FE}_i} \leftarrow \mathsf{FE.KeyGen}(\mathsf{MSK}_{\mathsf{FE}_i}, \mathsf{ReEnc}_{\mathsf{EK}_i,\perp}; r'_i)$
Return $(\mathsf{SK}_f, \mathsf{SK}_{\mathsf{FE}_1}, \ldots, \mathsf{SK}_{\mathsf{FE}_\ell})$

Oracle $\mathcal{O}_{\mathsf{enc}}(i, \mathbf{x}_0, \mathbf{x}_1)$
$T \leftarrow\!\!\$\ \{0,1\}^\kappa$
Return $\mathsf{FE.Enc}(\mathsf{MSK}_{\mathsf{FE}_i}, (\mathbf{x}_0, T, K_i^{\mathsf{punc}}, \mathsf{Trap}))$

Oracle $\mathcal{O}_{\mathsf{kg}}(i, (i_1, \ldots, i_\ell), f)$
$J \leftarrow I \cap \{i_1, \ldots, i_\ell\}$; $\bar{J} \leftarrow \{i_1, \ldots, i_\ell\} \setminus J$
$(\alpha_j, r_j, K_j) \leftarrow \widetilde{\mathsf{Extract}}(\rho_j^{(1)}) \quad \forall j \in J$
$y \leftarrow\!\!\$\ R_Y((i_1, \ldots, i_\ell), f)$
If $((i_1, \ldots, i_\ell), f, \bar{J}') \notin Q$
$\bar{J}' \leftarrow \emptyset$
$\bar{J}' \leftarrow \bar{J}' \cup \{i\}$
$Q \leftarrow Q \cup ((i_1, \ldots, i_\ell), f, \bar{J}')$
$(\rho_j^{(2)})_{j\in\bar{J}} \leftarrow\!\!\$\ \widetilde{\mathsf{Sim}_2}((\alpha_j, r_j, K_j)_{j\in J}, y)$
$\forall j \in \bar{J}$:
$(\gamma_j^{(0)}, \gamma_j^{(1)}, \gamma_j^{(2)}) \leftarrow\!\!\$\ R_\Gamma(j, (i_1, \ldots, i_\ell), f)$
$\forall m \in \{0,1,2\}, j \in J$,
$\gamma_j^{(m)} \leftarrow \mathsf{PRF.Eval}(K_j, m \,\|\, (\mathsf{PP}_{i_k})_{k\in[\ell]} \,\|\, f)$
If $\bar{J}' = \bar{J}$ // if *fully queried*
$S_1 \leftarrow\!\!\$\ \bigoplus_{j\in\bar{J}, j\neq i} \gamma_j^{(0)}$
$S_2 \leftarrow\!\!\$\ \bigoplus_{j\in J} \gamma_j^{(0)}$
$s \leftarrow y \oplus S_1 \oplus S_2 \oplus$
$\mathsf{GenKeys}''(\{(\gamma_{i_k}^{(2)}, \mathsf{MSK}_{\mathsf{FE}_{i_k}})\}_{k\in[\ell]})$
Else
$s \leftarrow \gamma_i^{(0)}$
Return $(\rho_i^{(2)}, s)$

Algorithm $R_\Gamma(i, (i_1, \ldots, i_\ell, f)$
If $(i, (i_1, \ldots, i_\ell), f, \gamma^{(0)}, \gamma^{(1)}, \gamma^{(2)}) \in \Gamma$
Return $(\gamma^{(0)}, \gamma^{(1)}, \gamma^{(2)})$
Else
$\gamma^{(j)} \leftarrow\!\!\$\ \{0,1\}^{\mathsf{rp}} \quad \forall j \in \{0,1,2\}$
// where rp is the range of the PRF
$\Gamma \leftarrow \Gamma \cup \{(i, (i_1, \ldots, i_\ell), f, \gamma^{(0)}, \gamma^{(1)}, \gamma^{(2)})\}$
Return $(\gamma^{(0)}, \gamma^{(1)}, \gamma^{(2)})$

Algorithm $R_Y((i_1, \ldots, i_\ell, f)$
If $((i_1, \ldots, i_\ell), f, y) \in Y$
Return y
Else
$y \leftarrow\!\!\$\ \{0,1\}^{\mathsf{rp}}$
$Y \leftarrow Y \cup \{((i_1, \ldots, i_\ell), f, y)\}$
Return y

Figure 13 Hybrid 3.

In more detail, we have a series of subhybrids, one for each $i \notin I$, where in Hybrid $4, i$, the change above is made to the function key associated with the FE instance for party i. Let Hybrid $4, 0$ denote Hybrid 3 and let Hybrid $|n \setminus I|$ denote Hybrid 4. We now give the formal reduction to FE function hiding for distinguishing Hybrid $4, i-1$ and Hybrid $4, i$ (for ease of notation, we assume that the indices i are consecutive). The simulator $\mathcal{B}$ is defined as follows. First $\mathcal{B}$ receives the public parameters PP from the FE challenger. Then it receives $(1^\kappa, I, (\mathsf{PP}_j)_{j\in I})$ from the hybrid distinguisher $\mathcal{A}$. Next it runs Step 2 to Step 10 on the left hand side of Figure 14 and passes $(\mathsf{PP}_j)_{j\notin I}$ (see Step 10) to $\mathcal{A}$. It handles encryption and key generation queries as follows. On an encryption query $(i', \mathbf{x}_0, \mathbf{x}_1)$ with $i' \neq i$, the query is handled the same as $\mathcal{O}_{\mathsf{enc}}$ in Figure 14. On an encryption query $(i, \mathbf{x}_0, \mathbf{x}_1)$, a tag $T \leftarrow\!\!\$\ \{0,1\}^\kappa$ is sampled and $\mathcal{B}$ makes a call to the FE encryption oracle with message $(\mathbf{x}_0, T, K_i^{\mathsf{punc}}, \mathsf{Trap} := (\mathsf{mode} := R, \perp, \perp, \perp, \perp))$ and returns the returned ciphertext. Key generation queries are handled as in $\mathcal{O}_{\mathsf{kg}}$ in Figure 14 with one exception, namely the secret keys $\mathsf{SK}_{\mathsf{FE}_{i'}}$ are computed as in $\mathsf{GenKeys}'''$ except for the case $i' = i$; the secret

Hybrid 4:
$(1^n, I, (\mathsf{PP}_i)_{i\in I}, st) \leftarrow\$ \mathcal{A}_0(1^\kappa)$
$b \leftarrow\$ \{0,1\}$
$\forall i \notin I$:
$K_i^{\mathsf{punc}} \leftarrow\$ \mathcal{K}_\kappa$
$(\mathsf{PP}_{\mathsf{FE}_i}, \mathsf{MSK}_{\mathsf{FE}_i}) \leftarrow\$ \mathsf{FE.Setup}(1^\kappa)$
$\Gamma \leftarrow \emptyset$
$Y \leftarrow \emptyset$
$Q \leftarrow \emptyset$
$q \leftarrow 0$
$(\rho_i^{(1)})_{i\notin I} \leftarrow\$ \widetilde{\mathsf{Sim}_1}()$
$\mathsf{PP}_i \leftarrow \rho_i^{(1)} \quad \forall i \notin I$

$st \leftarrow\$ \mathcal{A}_1^{\mathcal{O}_{\mathsf{enc}}(\cdot,\cdot,\cdot),\mathcal{O}_{\mathsf{kg}}(\cdot,\cdot,\cdot)}(st)$
$b' \leftarrow\$ \mathcal{A}_2^{\mathcal{O}_{\mathsf{enc}}(\cdot,\cdot,\cdot),\mathcal{O}_{\mathsf{kg}}(\cdot,\cdot,\cdot)}((\mathsf{PP}_i)_{i\notin I}, st)$
Return $(b = b')$

Algorithm $\mathsf{GenKeys}_f'''((r'_k, \mathsf{MSK}_{\mathsf{FE}_k})_{k\in[\ell]}, \mathsf{ctr})$
$((\mathsf{EK}_1, \ldots, \mathsf{EK}_\ell), \mathsf{MSK}) \leftarrow$
$\mathsf{MIFE.Setup}(1^\kappa)$
$\mathsf{SK}_f \leftarrow \mathsf{MIFE.KeyGen}(\mathsf{MSK}, f)$
$\forall i \in [\ell]$, do:
$\mathsf{SK}_{\mathsf{FE}_i} \leftarrow \mathsf{FE.KeyGen}(\mathsf{MSK}_{\mathsf{FE}_i}, \mathsf{ReEnc}_{\mathsf{EK}_i,\mathsf{ctr}}; r'_i)$
Return $(\mathsf{SK}_f, \mathsf{SK}_{\mathsf{FE}_1}, \ldots, \mathsf{SK}_{\mathsf{FE}_\ell})$

Oracle $\mathcal{O}_{\mathsf{enc}}(i, \mathbf{x}_0, \mathbf{x}_1)$
$T \leftarrow\$ \{0,1\}^\kappa$
Return $\mathsf{FE.Enc}(\mathsf{MSK}_{\mathsf{FE}_i}, (\mathbf{x}_0, T, K_i^{\mathsf{punc}}, \mathsf{Trap}))$

Oracle $\mathcal{O}_{\mathsf{kg}}(i, (i_1, \ldots, i_\ell), f)$
$J \leftarrow I \cap \{i_1, \ldots, i_\ell\}$; $\bar{J} \leftarrow \{i_1, \ldots, i_\ell\} \setminus J$
$(\alpha_j, r_j, K_j) \leftarrow \widetilde{\mathsf{Extract}}(\rho_j^{(1)}) \quad \forall j \in J$
$y \leftarrow\$ R_Y((i_1, \ldots, i_\ell), f)$
If $((i_1, \ldots, i_\ell), f, \bar{J}') \notin Q$
$\bar{J}' \leftarrow \emptyset$
$\bar{J}' \leftarrow \bar{J}' \cup \{i\}$
$Q \leftarrow Q \cup ((i_1, \ldots, i_\ell), f, \bar{J}')$
$(\rho_j^{(2)})_{j\in\bar{J}} \leftarrow\$ \widetilde{\mathsf{Sim}_2}((\alpha_j, r_j, K_j)_{j\in J}, y)$
$\forall j \in \bar{J}$:
$(\gamma_j^{(0)}, \gamma_j^{(1)}, \gamma_j^{(2)}) \leftarrow\$ R_\Gamma(j, (i_1, \ldots, i_\ell), f)$
$\forall m \in \{0,1,2\}, j \in J$,
$\gamma_j^{(m)} \leftarrow \mathsf{PRF.Eval}(K_j, m \,\|\, (\mathsf{PP}_{i_k})_{k\in[\ell]} \,\|\, f)$
If $\bar{J}' = \bar{J}$ // if *fully queried*
$q \leftarrow q + 1$
$S_1 \leftarrow\$ \bigoplus_{j\in\bar{J}, j\neq i} \gamma_j^{(0)}$
$S_2 \leftarrow\$ \bigoplus_{j\in J} \gamma_j^{(0)}$
$s \leftarrow y \oplus S_1 \oplus S_2 \oplus$
$\mathsf{GenKeys}'''(\{(\gamma_{i_k}^{(2)}, \mathsf{MSK}_{\mathsf{FE}_{i_k}})\}_{k\in[\ell]}, q)$
Else
$s \leftarrow \gamma_i^{(0)}$
Return $(\rho_i^{(2)}, s)$

Algorithm $R_\Gamma(i, (i_1, \ldots, i_\ell, f)$
If $(i, (i_1, \ldots, i_\ell), f, \gamma^{(0)}, \gamma^{(1)}, \gamma^{(2)}) \in \Gamma$
Return $(\gamma^{(0)}, \gamma^{(1)}, \gamma^{(2)})$
Else
$\gamma^{(j)} \leftarrow\$ \{0,1\}^{\mathsf{rp}} \quad \forall j \in \{0,1,2\}$
// where rp is the range of the PRF
$\Gamma \leftarrow \Gamma \cup \{(i, (i_1, \ldots, i_\ell), f, \gamma^{(0)}, \gamma^{(1)}, \gamma^{(2)})\}$
Return $(\gamma^{(0)}, \gamma^{(1)}, \gamma^{(2)})$

Algorithm $R_Y((i_1, \ldots, i_\ell, f)$
If $((i_1, \ldots, i_\ell), f, y) \in Y$
Return y
Else
$y \leftarrow\$ \{0,1\}^{\mathsf{rp}}$
$Y \leftarrow Y \cup \{((i_1, \ldots, i_\ell), f, y)\}$
Return y

Figure 14 Hybrid 4.

key $\mathsf{SK}_{\mathsf{FE}_i}$ is obtained by making a call to the FE key generation oracle with functions $(\mathsf{ReEnc}_{i,\perp}, \mathsf{ReEnc}_{i,\mathsf{ctr}})$. If the FE challenger's bit is 0, then $\mathcal{B}$ perfectly simulates Hybrid $4, i-1$ and if the FE challenger's bit is 1, then $\mathcal{B}$ perfectly simulates Hybrid $4, i$.

For $j \in [Q_k]$, we define:

Hybrid $5_{\mathsf{j},1}$: In this hybrid, we hardwire all the Q_c MIFE CTs that are output by the j^{th} function query in the corresponding single input FE ciphertexts in the field Trap.CT and set mode $= T$.

In more detail, for key query j, we generate the encryption keys exactly as in Figure 2. Now, encryptor $i \in [n]$ computes Q_c ciphertexts as follows:

1. For $k \in [Q_c]$, let $r_{i,j,k} \leftarrow \mathsf{PRF.Eval}(K_i^{\mathsf{punc}}, j \,\|\, \mathsf{EK}_{i,j} \,\|\, T_{i,k})$
2. For $k \in [Q_c]$, let $\psi_{i,j,k} = \mathsf{MIFE.Enc}(\mathsf{EK}_{i,j}, \mathbf{x}_{i,k}; r_{i,j,k})$

For $k \in [Q_c]$, encryptor $i \in [n]$ sets Trap so as to program it for the j^{th} function query as follows:

$$\mathsf{Trap.mode} = T, \quad \mathsf{Trap.CT} = \psi_{i,j,k}, \quad \mathsf{Trap.index} = j, \quad \mathsf{val}_0 = \mathbf{x}_{0,i,k}, \quad \mathsf{val}_1 = \mathbf{x}_{1,i,k}$$

Please see Figure 15 for the complete description.

Hybrid $5_{j,1}$:

$(1^n, I, (\mathsf{PP}_i)_{i\in I}, st) \leftarrow\!\!\$\ \mathcal{A}_0(1^\kappa)$
$b \leftarrow\!\!\$\ \{0,1\}$
$\forall i \notin I$:
$K_i^{\mathsf{punc}} \leftarrow\!\!\$\ \mathcal{K}_\kappa$
$(\mathsf{PP}_{\mathsf{FE}_i}, \mathsf{MSK}_{\mathsf{FE}_i}) \leftarrow\!\!\$\ \mathsf{FE.Setup}(1^\kappa)$
$\Gamma \leftarrow \emptyset$
$Y \leftarrow \emptyset$
$Q \leftarrow \emptyset$
$q \leftarrow 0$
$\mathsf{mkeys} := ((\mathsf{EK}_{1,j}, \ldots, \mathsf{EK}_{n,j}), \mathsf{MSK}_j) \leftarrow\!\!\$\ \mathsf{MIFE.Setup}(1^\kappa)$
$(\rho_i^{(1)})_{i\notin I} \leftarrow\!\!\$\ \widetilde{\mathsf{Sim}_1}()$
$\mathsf{PP}_i \leftarrow \rho_i^{(1)} \quad \forall i \notin I$

$st \leftarrow\!\!\$\ \mathcal{A}_1^{\mathcal{O}_{\mathsf{enc}}(\cdot,\cdot,\cdot),\mathcal{O}_{\mathsf{kg}}(\cdot,\cdot,\cdot)}(st)$
$b' \leftarrow\!\!\$\ \mathcal{A}_2^{\mathcal{O}_{\mathsf{enc}}(\cdot,\cdot,\cdot),\mathcal{O}_{\mathsf{kg}}(\cdot,\cdot,\cdot)}((\mathsf{PP}_i)_{i\notin I}, st)$
Return $(b = b')$

Algorithm $\mathsf{GenKeys}''''_f((r'_k, \mathsf{MSK}_{\mathsf{FE}_k})_{k\in[\ell]}, \mathsf{ctr}, \mathsf{mkeys})$
If $\mathsf{ctr} = \mathsf{j}$
$((\mathsf{EK}_1, \ldots, \mathsf{EK}_\ell), \mathsf{MSK}) \leftarrow \mathsf{mkeys}$
Else
$((\mathsf{EK}_1, \ldots, \mathsf{EK}_\ell), \mathsf{MSK}) \leftarrow \mathsf{MIFE.Setup}(1^\kappa)$
$\mathsf{SK}_f \leftarrow \mathsf{MIFE.KeyGen}(\mathsf{MSK}, f)$
$\forall i \in [\ell]$, do:
$\mathsf{SK}_{\mathsf{FE}_i} \leftarrow \mathsf{FE.KeyGen}(\mathsf{MSK}_{\mathsf{FE}_i}, \mathsf{ReEnc}_{\mathsf{EK}_i,\mathsf{ctr}}; r'_i)$
Return $(\mathsf{SK}_f, \mathsf{SK}_{\mathsf{FE}_1}, \ldots, \mathsf{SK}_{\mathsf{FE}_\ell})$

Oracle $\mathcal{O}_{\mathsf{enc}}(i, \mathbf{x}_0, \mathbf{x}_1)$
$T \leftarrow\!\!\$\ \{0,1\}^\kappa$
$r \leftarrow \mathsf{PRF.Eval}(K_i^{\mathsf{punc}}, \mathsf{j} \parallel \mathsf{EK}_{i,\mathsf{j}} \parallel T)$
$\psi \leftarrow \mathsf{MIFE.Enc}(\mathsf{EK}_{i,\mathsf{j}}, \mathbf{x}_0; r)$
$\mathsf{Trap} \leftarrow (\mathsf{mode} := \mathsf{Trap}, \mathsf{CT} := \psi, \mathsf{index} := \mathsf{j}, \mathsf{val}_0 := \mathbf{x}_0, \mathsf{val}_1 := \mathbf{x}_1)$
Return $\mathsf{FE.Enc}(\mathsf{MSK}_{\mathsf{FE}_i}, (\mathbf{x}_0, T, K_i^{\mathsf{punc}}, \mathsf{Trap}))$

Oracle $\mathcal{O}_{\mathsf{kg}}(i, (i_1, \ldots, i_\ell), f)$
$J \leftarrow I \cap \{i_1, \ldots, i_\ell\}$; $\bar{J} \leftarrow \{i_1, \ldots, i_\ell\} \setminus J$
$(\alpha_j, r_j, K_j) \leftarrow \widetilde{\mathsf{Extract}}(\rho_j^{(1)}) \quad \forall j \in J$
$y \leftarrow\!\!\$\ R_Y((i_1, \ldots, i_\ell), f)$
If $((i_1, \ldots, i_\ell), f, \bar{J}') \notin Q$
$\bar{J}' \leftarrow \emptyset$
$\bar{J}' \leftarrow \bar{J}' \cup \{i\}$
$Q \leftarrow Q \cup ((i_1, \ldots, i_\ell), f, \bar{J}')$
$(\rho_j^{(2)})_{j\in\bar{J}} \leftarrow\!\!\$\ \widetilde{\mathsf{Sim}_2}((\alpha_j, r_j, K_j)_{j\in J}, y)$
$\forall j \in \bar{J}$:
$(\gamma_j^{(0)}, \gamma_j^{(1)}, \gamma_j^{(2)}) \leftarrow\!\!\$\ R_\Gamma(j, (i_1, \ldots, i_\ell), f)$
$\forall m \in \{0,1,2\}, j \in J$,
$\gamma_j^{(m)} \leftarrow \mathsf{PRF.Eval}(K_j, m \parallel (\mathsf{PP}_{i_k})_{k\in[\ell]} \parallel f)$
If $\bar{J}' = \bar{J}$ // if *fully queried*
$q \leftarrow q + 1$
$S_1 \leftarrow\!\!\$\ \bigoplus_{j\in\bar{J}, j\neq i} \gamma_j^{(0)}$
$S_2 \leftarrow\!\!\$\ \bigoplus_{j\in J} \gamma_j^{(0)}$
$s \leftarrow y \oplus S_1 \oplus S_2 \oplus$
$\mathsf{GenKeys}''''(\{(\gamma_{i_k}^{(2)}, \mathsf{MSK}_{\mathsf{FE}_{i_k}})\}_{k\in[\ell]}, q, \mathsf{mkeys})$
Else
$s \leftarrow \gamma_i^{(0)}$
Return $(\rho_i^{(2)}, s)$

Algorithm $R_\Gamma(i, (i_1, \ldots, i_\ell, f)$
If $(i, (i_1, \ldots, i_\ell), f, \gamma^{(0)}, \gamma^{(1)}, \gamma^{(2)}) \in \Gamma$
Return $(\gamma^{(0)}, \gamma^{(1)}, \gamma^{(2)})$
Else
$\gamma^{(j)} \leftarrow\!\!\$\ \{0,1\}^{\mathsf{rp}} \quad \forall j \in \{0,1,2\}$
// where rp is the range of the PRF
$\Gamma \leftarrow \Gamma \cup \{(i, (i_1, \ldots, i_\ell), f, \gamma^{(0)}, \gamma^{(1)}, \gamma^{(2)})\}$
Return $(\gamma^{(0)}, \gamma^{(1)}, \gamma^{(2)})$

Algorithm $R_Y((i_1, \ldots, i_\ell, f)$
If $((i_1, \ldots, i_\ell), f, y) \in Y$
Return y
Else
$y \leftarrow\!\!\$\ \{0,1\}^{\mathsf{rp}}$
$Y \leftarrow Y \cup \{((i_1, \ldots, i_\ell), f, y)\}$
Return y

Figure 15 Hybrid $5_{j,1}$.

Note that the hardwired ciphertext is only output for query j, the outputs for the other queries are exactly equal to those in the previous hybrid. Now, for query j, the ciphertext is hardwired and output is set to be equal to what was output in the previous hybrid. It follows that the output of FE decryption remains exactly the same as in the previous hybrid. Thus, by security of FE, we have that the two hybrids are indistinguishable.

In more detail, we have a series of subhybrids, one for each $i \notin I$, where in Hybrid $5_{j,1}, i$, the change above is made to the ciphertext associated with the FE instance for party i. Let Hybrid $5_{j,1}, 0$ denote Hybrid 4 and let Hybrid $5_{j,1}, |n \setminus I|$ denote Hybrid $5_{j,1}$. We now give the formal reduction to FE semantic security for distinguishing Hybrid $5_{j,1}, i-1$ and Hybrid $5_{j,1}, i$ (for ease of notation, we assume that the indices i are consecutive). The simulator $\mathcal{B}$ is defined as follows. First $\mathcal{B}$ receives the public parameters PP from the FE challenger. Then it receives $(1^\kappa, I, (\mathsf{PP}_j)_{j\in I})$ from the hybrid distinguisher $\mathcal{A}$. Next it runs

Step 2 to Step 11 on the left hand side of Figure 15 and passes $(\mathsf{PP}_j)_{j\notin I}$ (see Step 11) to $\mathcal{A}$. It handles encryption and key generation queries as follows. On an encryption query $(i', \mathbf{x}_0, \mathbf{x}_1)$ with $i' \neq i$, the query is handled the same as $\mathcal{O}_{\mathsf{enc}}$ in Figure 15. On an encryption query $(i, \mathbf{x}_0, \mathbf{x}_1)$, a tag $T \leftarrow\!\!\$\ \{0,1\}^\kappa$ is sampled, then $r \leftarrow \mathsf{PRF.Eval}(K_i^{\mathsf{punc}}, j \parallel \mathsf{EK}_{i,j} \parallel T)$, $\psi \leftarrow \mathsf{MIFE.Enc}(\mathsf{EK}_{i,j}, \mathbf{x}_0; r)$, $\mathsf{Trap}_0 \leftarrow (\mathsf{mode} := R, \perp, \perp, \perp, \perp)$ and $\mathsf{Trap}_1 \leftarrow (\mathsf{mode} := \mathsf{Trap}, \psi, j, \mathbf{x}_0, \mathbf{x}_1)$ are computed and $\mathcal{B}$ makes a call to the FE encryption oracle with messages $(\mathbf{x}_0, T, K_i^{\mathsf{punc}}, \mathsf{Trap}_0)$ and $(\mathbf{x}_0, T, K_i^{\mathsf{punc}}, \mathsf{Trap}_1)$, and returns the returned ciphertext. Key generation queries are handled as in $\mathcal{O}_{\mathsf{kg}}$ in Figure 15 with one exception, namely the secret keys $\mathsf{SK}_{\mathsf{FE}_{i'}}$ are computed as in $\mathsf{GenKeys}''''$ except for the case $i' = i$; the secret key $\mathsf{SK}_{\mathsf{FE}_i}$ is obtained by making a call to the FE key generation oracle for function $\mathsf{ReEnc}_{i,\mathsf{ctr}}$. If the FE challenger's bit is 0, then $\mathcal{B}$ perfectly simulates Hybrid $5_{j,1}, i-1$ and if the FE challenger's bit is 1, then $\mathcal{B}$ perfectly simulates Hybrid $5_{j,1}, i$.

Hybrid $5_{\mathbf{j},\mathbf{2}}$: In this hybrid, we use a punctured PRF to generate the MIFE CTs in the ReEnc functionality encoded in the FE function keys. The PRF for party i is punctured at prefix j so that the randomness $r_{i,j,k}$ defined above, for $i \in [n]$, $k \in [Q_c]$ cannot be generated. All MIFE ciphertexts corresponding to other function queries can be generated as before.

In more detail:

1. For $i \in [n]$, party i samples $K_{i,j}^{\mathsf{punc}} \leftarrow \mathsf{puncF}(K_i^{\mathsf{punc}}, j)$.
2. The i^{th} encryptor computes $\mathsf{FE.enc}\big(\mathsf{MSK}_{\mathsf{FE}}, (\mathbf{x}_{0,i,k}, T_{i,k}, K_{i,j}^{\mathsf{punc}}, \mathsf{Trap})\big)$ where all other fields are set as in the previous hybrid.

During FE decryption, for any query $j' \neq j$, we now obtain:

$$r_{i,j',k} \leftarrow \mathsf{puncF}(K_{i,j}^{\mathsf{punc}}, j' \parallel \mathsf{EK}_{i,j'} \parallel T_{i,k})$$

for $k \in [Q_c]$. Everything else is as in the previous hybrid. For query j, the hardwired CT is output, and the punctured PRF key is not used to generate randomness. Please see Figure 16 for the complete description.

We have by correctness of the punctured PRF that for any $j' \neq j$, all the computed $r_{i,j',k}$ are exactly equal to those computed in the previous hybrid, where the normal PRF key was used. For query j, the PRF is not used and the hardwired value is output in both hybrids. Hence, the outputs of FE decryption are equal in both hybrids. Thus, indistinguishability follows from security of FE.

In more detail, we have a series of subhybrids, one for each $i \notin I$, where in Hybrid $5_{j,2}, i$, the change above is made to the ciphertext associated with the FE instance for party i. Let Hybrid $5_{j,2}, 0$ denote Hybrid $5_{j,1}$ and let Hybrid $5_{j,2}, |n \setminus I|$ denote Hybrid $5_{j,2}$. We now give the formal reduction to FE semantic security for distinguishing Hybrid $5_{j,2}, i-1$ and Hybrid $5_{j,2}, i$ (for ease of notation, we assume that the indices i are consecutive). The simulator $\mathcal{B}$ is defined as follows. First $\mathcal{B}$ receives the public parameters PP from the FE challenger. Then it receives $(1^\kappa, I, (\mathsf{PP}_j)_{j\in I})$ from the hybrid distinguisher $\mathcal{A}$. Next it runs Step 2 to Step 12 on the left hand side of Figure 16 and passes $(\mathsf{PP}_j)_{j\notin I}$ (see Step 12) to $\mathcal{A}$. It handles encryption and key generation queries as follows. On an encryption query $(i', \mathbf{x}_0, \mathbf{x}_1)$ with $i' \neq i$, the query is handled the same as $\mathcal{O}_{\mathsf{enc}}$ in Figure 16. On an encryption query $(i, \mathbf{x}_0, \mathbf{x}_1)$, a tag $T \leftarrow\!\!\$\ \{0,1\}^\kappa$ is sampled, then $r \leftarrow \mathsf{PRF.Eval}(K_i^{\mathsf{punc}}, j \parallel \mathsf{EK}_{i,j} \parallel T)$, $\psi \leftarrow \mathsf{MIFE.Enc}(\mathsf{EK}_{i,j}, \mathbf{x}_0; r)$ and $\mathsf{Trap} \leftarrow (\mathsf{mode} := \mathsf{Trap}, \psi, j, \mathbf{x}_0, \mathbf{x}_1)$ are computed and $\mathcal{B}$ makes a call to the FE encryption oracle with messages $(\mathbf{x}_0, T, K_i^{\mathsf{punc}}, \mathsf{Trap})$ and $(\mathbf{x}_0, T, K_{i,j}^{\mathsf{punc}}, \mathsf{Trap})$, and returns the returned ciphertext. Note that the punctured key $K_{i,j}^{\mathsf{punc}}$ is derived in Step 4 on the left hand side of Figure 16. Key generation queries are handled as in $\mathcal{O}_{\mathsf{kg}}$ in Figure 15 with one exception, namely the secret keys $\mathsf{SK}_{\mathsf{FE}_{i'}}$ are computed as in $\mathsf{GenKeys}''''$ except for the case $i' = i$; the

Hybrid $5_{\mathsf{j},2}$:

$(1^n, I, (\mathsf{PP}_i)_{i\in I}, st) \leftarrow\!\!\$\ \mathcal{A}_0(1^\kappa)$
$b \leftarrow\!\!\$\ \{0,1\}$
$\forall i \notin I$:
$K_i^{\mathsf{punc}} \leftarrow\!\!\$\ \mathcal{K}_\kappa$
$K_{i,\mathsf{j}}^{\mathsf{punc}} \leftarrow \mathsf{PRF.Punc}(K_i^{\mathsf{punc}}, j)$
$(\mathsf{PP}_{\mathsf{FE}_i}, \mathsf{MSK}_{\mathsf{FE}_i}) \leftarrow\!\!\$\ \mathsf{FE.Setup}(1^\kappa)$
$\Gamma \leftarrow \emptyset$
$Y \leftarrow \emptyset$
$Q \leftarrow \emptyset$
$q \leftarrow 0$
$\mathsf{mkeys} := ((\mathsf{EK}_{1,\mathsf{j}}, \ldots, \mathsf{EK}_{n,\mathsf{j}}), \mathsf{MSK}_\mathsf{j})$
$\leftarrow\!\!\$\ \mathsf{MIFE.Setup}(1^\kappa)$
$(\rho_i^{(1)})_{i\notin I} \leftarrow\!\!\$\ \widetilde{\mathsf{Sim}_1}()$
$\mathsf{PP}_i \leftarrow \rho_i^{(1)} \quad \forall i \notin I$

$st \leftarrow\!\!\$\ \mathcal{A}_1^{\mathcal{O}_{\mathsf{enc}}(\cdot,\cdot,\cdot),\mathcal{O}_{\mathsf{kg}}(\cdot,\cdot,\cdot)}(st)$
$b' \leftarrow\!\!\$\ \mathcal{A}_2^{\mathcal{O}_{\mathsf{enc}}(\cdot,\cdot,\cdot),\mathcal{O}_{\mathsf{kg}}(\cdot,\cdot,\cdot)}((\mathsf{PP}_i)_{i\notin I}, st)$
Return $(b = b')$

Algorithm $\mathsf{GenKeys}_f''''((r'_k, \mathsf{MSK}_{\mathsf{FE}_k})_{k\in[\ell]}, \mathsf{ctr}, \mathsf{mkeys})$
If $\mathsf{ctr} = \mathsf{j}$
$((\mathsf{EK}_1, \ldots, \mathsf{EK}_\ell), \mathsf{MSK}) \leftarrow \mathsf{mkeys}$
Else
$((\mathsf{EK}_1, \ldots, \mathsf{EK}_\ell), \mathsf{MSK}) \leftarrow \mathsf{MIFE.Setup}(1^\kappa)$
$\mathsf{SK}_f \leftarrow \mathsf{MIFE.KeyGen}(\mathsf{MSK}, f)$
$\forall i \in [\ell]$, do:
$\mathsf{SK}_{\mathsf{FE}_i} \leftarrow \mathsf{FE.KeyGen}(\mathsf{MSK}_{\mathsf{FE}_i}, \mathsf{ReEnc}_{\mathsf{EK}_i,\mathsf{ctr}}; r'_i)$
Return $(\mathsf{SK}_f, \mathsf{SK}_{\mathsf{FE}_1}, \ldots, \mathsf{SK}_{\mathsf{FE}_\ell})$

Oracle $\mathcal{O}_{\mathsf{enc}}(i, \mathbf{x}_0, \mathbf{x}_1)$
$T \leftarrow\!\!\$\ \{0,1\}^\kappa$
$r \leftarrow \mathsf{PRF.Eval}(K_i^{\mathsf{punc}}, \mathsf{j} \parallel \mathsf{EK}_{i,\mathsf{j}} \parallel T)$
$\psi \leftarrow \mathsf{MIFE.Enc}(\mathsf{EK}_{i,\mathsf{j}}, \mathbf{x}_0; r)$
$\mathsf{Trap} \leftarrow (\mathsf{mode} := \mathsf{Trap}, \mathsf{CT} := \psi,$
$\mathsf{index} := \mathsf{j}, \mathsf{val}_0 := \mathbf{x}_0, \mathsf{val}_1 := \mathbf{x}_1)$
Return $\mathsf{FE.Enc}(\mathsf{MSK}_{\mathsf{FE}_i}, (\mathbf{x}_0, T, K_{i,\mathsf{j}}^{\mathsf{punc}}, \mathsf{Trap}))$

Oracle $\mathcal{O}_{\mathsf{kg}}(i, (i_1, \ldots, i_\ell), f)$
$J \leftarrow I \cap \{i_1, \ldots, i_\ell\}$; $\bar{J} \leftarrow \{i_1, \ldots, i_\ell\} \setminus J$
$(\alpha_j, r_j, K_j) \leftarrow \widetilde{\mathsf{Extract}}(\rho_j^{(1)}) \quad \forall j \in J$
$y \leftarrow\!\!\$\ R_Y((i_1, \ldots, i_\ell), f)$
If $((i_1, \ldots, i_\ell), f, \bar{J}') \notin Q$
$\bar{J}' \leftarrow \emptyset$
$\bar{J}' \leftarrow \bar{J}' \cup \{i\}$
$Q \leftarrow Q \cup ((i_1, \ldots, i_\ell), f, \bar{J}')$
$(\rho_j^{(2)})_{j\in\bar{J}} \leftarrow\!\!\$\ \widetilde{\mathsf{Sim}_2}((\alpha_j, r_j, K_j)_{j\in J}, y)$
$\forall j \in \bar{J}$:
$(\gamma_j^{(0)}, \gamma_j^{(1)}, \gamma_j^{(2)}) \leftarrow\!\!\$\ R_\Gamma(j, (i_1, \ldots, i_\ell), f)$
$\forall m \in \{0,1,2\}, j \in J,$
$\gamma_j^{(m)} \leftarrow \mathsf{PRF.Eval}(K_j, m \parallel (\mathsf{PP}_{i_k})_{k\in[\ell]} \parallel f)$
If $\bar{J}' = \bar{J}$ // if *fully queried*
$q \leftarrow q + 1$
$S_1 \leftarrow\!\!\$\ \bigoplus_{j\in\bar{J}, j\neq i} \gamma_j^{(0)}$
$S_2 \leftarrow\!\!\$\ \bigoplus_{j\in J} \gamma_j^{(0)}$
$s \leftarrow y \oplus S_1 \oplus S_2 \oplus$
$\mathsf{GenKeys}''''(\{(\gamma_{i_k}^{(2)}, \mathsf{MSK}_{\mathsf{FE}_{i_k}})\}_{k\in[\ell]}, q, \mathsf{mkeys})$
Else
$s \leftarrow \gamma_i^{(0)}$
Return $(\rho_i^{(2)}, s)$

Algorithm $R_\Gamma(i, (i_1, \ldots, i_\ell, f)$
If $(i, (i_1, \ldots, i_\ell), f, \gamma^{(0)}, \gamma^{(1)}, \gamma^{(2)}) \in \Gamma$
Return $(\gamma^{(0)}, \gamma^{(1)}, \gamma^{(2)})$
Else
$\gamma^{(j)} \leftarrow\!\!\$\ \{0,1\}^{\mathsf{rp}} \quad \forall j \in \{0,1,2\}$
// where rp is the range of the PRF
$\Gamma \leftarrow \Gamma \cup \{(i, (i_1, \ldots, i_\ell), f, \gamma^{(0)}, \gamma^{(1)}, \gamma^{(2)})\}$
Return $(\gamma^{(0)}, \gamma^{(1)}, \gamma^{(2)})$

Algorithm $R_Y((i_1, \ldots, i_\ell, f)$
If $((i_1, \ldots, i_\ell), f, y) \in Y$
Return y
Else
$y \leftarrow\!\!\$\ \{0,1\}^{\mathsf{rp}}$
$Y \leftarrow Y \cup \{((i_1, \ldots, i_\ell), f, y)\}$
Return y

Figure 16 Hybrid $5_{j,2}$.

secret key $\mathsf{SK}_{\mathsf{FE}_i}$ is obtained by making a call to the FE key generation oracle for function $\mathsf{ReEnc}_{i,\mathsf{ctr}}$. If the FE challenger's bit is 0, then $\mathcal{B}$ perfectly simulates Hybrid $5_{j,2}, i-1$ and if the FE challenger's bit is 1, then $\mathcal{B}$ perfectly simulates Hybrid $5_{j,2}, i$.

Hybrid $5_{\mathsf{j},3}$: In this hybrid, we switch the randomness used in the hardwired MIFE CT to be true randomness. That is, $r_{i,j,k}$ is sampled uniformly at random for $i \in [n]$, $k \in [Q_c]$. We have by the security of the punctured PRF that given the punctured key, the PRF evaluations at the punctured points are indistinguishable from random. Hence, indistinguishability follows from security of punctured PRF.

Please see Figure 17 for the detailed description.

Hybrid $5_{\mathsf{j},3}$:
$(1^n, I, (\mathsf{PP}_i)_{i\in I}, st) \leftarrow\!\!\$\ \mathcal{A}_0(1^\kappa)$
$b \leftarrow\!\!\$\ \{0,1\}$
$\forall i \notin I$:
$K_i^{\mathsf{punc}} \leftarrow\!\!\$\ \mathcal{K}_\kappa$
$K_{i,\mathsf{j}}^{\mathsf{punc}} \leftarrow \mathsf{PRF.Punc}(K_i^{\mathsf{punc}}, j)$
$(\mathsf{PP}_{\mathsf{FE}_i}, \mathsf{MSK}_{\mathsf{FE}_i}) \leftarrow\!\!\$\ \mathsf{FE.Setup}(1^\kappa)$
$\Gamma \leftarrow \emptyset$
$Y \leftarrow \emptyset$
$Q \leftarrow \emptyset$
$q \leftarrow 0$
$\mathsf{mkeys} := ((\mathsf{EK}_{1,\mathsf{j}}, \ldots, \mathsf{EK}_{n,\mathsf{j}}), \mathsf{MSK}_\mathsf{j}) \leftarrow\!\!\$\ \mathsf{MIFE.Setup}(1^\kappa)$
$(\rho_i^{(1)})_{i\notin I} \leftarrow\!\!\$\ \widetilde{\mathsf{Sim}_1}()$
$\mathsf{PP}_i \leftarrow \rho_i^{(1)} \quad \forall i \notin I$

$st \leftarrow\!\!\$\ \mathcal{A}_1^{\mathcal{O}_{\mathsf{enc}}(\cdot,\cdot,\cdot),\mathcal{O}_{\mathsf{kg}}(\cdot,\cdot,\cdot)}(st)$
$b' \leftarrow\!\!\$\ \mathcal{A}_2^{\mathcal{O}_{\mathsf{enc}}(\cdot,\cdot,\cdot),\mathcal{O}_{\mathsf{kg}}(\cdot,\cdot,\cdot)}((\mathsf{PP}_i)_{i\notin I}, st)$
Return $(b = b')$

Algorithm $\mathsf{GenKeys}_f''''((r'_k, \mathsf{MSK}_{\mathsf{FE}_k})_{k\in[\ell]}, \mathsf{ctr}, \mathsf{mkeys})$
If $\mathsf{ctr} = \mathsf{j}$
$((\mathsf{EK}_1, \ldots, \mathsf{EK}_\ell), \mathsf{MSK}) \leftarrow \mathsf{mkeys}$
Else
$((\mathsf{EK}_1, \ldots, \mathsf{EK}_\ell), \mathsf{MSK}) \leftarrow \mathsf{MIFE.Setup}(1^\kappa)$
$\mathsf{SK}_f \leftarrow \mathsf{MIFE.KeyGen}(\mathsf{MSK}, f)$
$\forall i \in [\ell]$, do:
$\mathsf{SK}_{\mathsf{FE}_i} \leftarrow \mathsf{FE.KeyGen}(\mathsf{MSK}_{\mathsf{FE}_i}, \mathsf{ReEnc}_{\mathsf{EK}_i,\mathsf{ctr}}; r'_i)$
Return $(\mathsf{SK}_f, \mathsf{SK}_{\mathsf{FE}_1}, \ldots, \mathsf{SK}_{\mathsf{FE}_\ell})$

Oracle $\mathcal{O}_{\mathsf{enc}}(i, \mathbf{x}_0, \mathbf{x}_1)$
$T \leftarrow\!\!\$\ \{0,1\}^\kappa$
$r \leftarrow\!\!\$\ \{0,1\}^{rp}$
// where rp is the range of the PRF
$\psi \leftarrow \mathsf{MIFE.Enc}(\mathsf{EK}_{i,\mathsf{j}}, \mathbf{x}_0; r)$
$\mathsf{Trap} \leftarrow (\mathsf{mode} := \mathsf{Trap}, \mathsf{CT} := \psi, \mathsf{index} := \mathsf{j}, \mathsf{val}_0 := \mathbf{x}_0, \mathsf{val}_1 := \mathbf{x}_1)$
Return $\mathsf{FE.Enc}(\mathsf{MSK}_{\mathsf{FE}_i}, (\mathbf{x}_0, T, K_{i,\mathsf{j}}^{\mathsf{punc}}, \mathsf{Trap}))$

Oracle $\mathcal{O}_{\mathsf{kg}}(i, (i_1, \ldots, i_\ell), f)$
$J \leftarrow I \cap \{i_1, \ldots, i_\ell\}$; $\bar{J} \leftarrow \{i_1, \ldots, i_\ell\} \setminus J$
$(\alpha_j, r_j, K_j) \leftarrow \widetilde{\mathsf{Extract}}(\rho_j^{(1)}) \quad \forall j \in J$
$y \leftarrow\!\!\$\ R_Y((i_1, \ldots, i_\ell), f)$
If $((i_1, \ldots, i_\ell), f, \bar{J}') \notin Q$
$\bar{J}' \leftarrow \emptyset$
$\bar{J}' \leftarrow \bar{J}' \cup \{i\}$
$Q \leftarrow Q \cup ((i_1, \ldots, i_\ell), f, \bar{J}')$
$(\rho_j^{(2)})_{j\in\bar{J}} \leftarrow\!\!\$\ \widetilde{\mathsf{Sim}_2}((\alpha_j, r_j, K_j)_{j\in J}, y)$
$\forall j \in \bar{J}$:
$(\gamma_j^{(0)}, \gamma_j^{(1)}, \gamma_j^{(2)}) \leftarrow\!\!\$\ R_\Gamma(j, (i_1, \ldots, i_\ell), f)$
$\forall m \in \{0,1,2\}, j \in J$,
$\gamma_j^{(m)} \leftarrow \mathsf{PRF.Eval}(K_j, m \parallel (\mathsf{PP}_{i_k})_{k\in[\ell]} \parallel f)$
If $\bar{J}' = \bar{J}$ // if *fully queried*
$q \leftarrow q + 1$
$S_1 \leftarrow\!\!\$\ \bigoplus_{j\in\bar{J}, j\neq i} \gamma_j^{(0)}$
$S_2 \leftarrow\!\!\$\ \bigoplus_{j\in J} \gamma_j^{(0)}$
$s \leftarrow y \oplus S_1 \oplus S_2 \oplus$
$\mathsf{GenKeys}''''(\{(\gamma_{i_k}^{(2)}, \mathsf{MSK}_{\mathsf{FE}_{i_k}})\}_{k\in[\ell]}, q, \mathsf{mkeys})$
Else
$s \leftarrow \gamma_i^{(0)}$
Return $(\rho_i^{(2)}, s)$

Algorithm $R_\Gamma(i, (i_1, \ldots, i_\ell), f)$
If $(i, (i_1, \ldots, i_\ell), f, \gamma^{(0)}, \gamma^{(1)}, \gamma^{(2)}) \in \Gamma$
Return $(\gamma^{(0)}, \gamma^{(1)}, \gamma^{(2)})$
Else
$\gamma^{(j)} \leftarrow\!\!\$\ \{0,1\}^{\mathsf{rp}} \quad \forall j \in \{0,1,2\}$
// where rp is the range of the PRF
$\Gamma \leftarrow \Gamma \cup \{(i, (i_1, \ldots, i_\ell), f, \gamma^{(0)}, \gamma^{(1)}, \gamma^{(2)})\}$
Return $(\gamma^{(0)}, \gamma^{(1)}, \gamma^{(2)})$

Algorithm $R_Y((i_1, \ldots, i_\ell), f)$
If $((i_1, \ldots, i_\ell), f, y) \in Y$
Return y
Else
$y \leftarrow\!\!\$\ \{0,1\}^{\mathsf{rp}}$
$Y \leftarrow Y \cup \{((i_1, \ldots, i_\ell), f, y)\}$
Return y

Figure 17 Hybrid $5_{j,3}$.

In more detail, we have a series of subhybrids, one for each $i \notin I$, where in Hybrid $5_{j,3}, i$, the change above is made to the function key associated with the FE instance for party i. Let Hybrid $5_{j,3}, 0$ denote Hybrid $5_{j,2}$ and let Hybrid $5_{j,3}, |n \setminus I|$ denote Hybrid $5_{j,3}$. We now give the formal reduction to PRF security for distinguishing Hybrid $5_{j,3}, i-1$ and Hybrid $5_{j,3}, i$ (for ease of notation, we assume that the indices i are consecutive). The simulator $\mathcal{B}$ is defined as follows. First $\mathcal{B}$ receives $(1^\kappa, I, (\mathsf{PP}_j)_{j\in I})$ from the hybrid distinguisher $\mathcal{A}$. Next it runs

- For all $i' \notin I$, $i' \neq i$:
  - $K_{i'}^{\mathsf{punc}} \leftarrow\!\!\$\ \mathcal{K}_\kappa$
  - $K_{i',j}^{\mathsf{punc}} \leftarrow \mathsf{PRF.Punc}(K_{i'}^{\mathsf{punc}}, j)$

Then it sends a challenge prefix j to the PRF challenger and receives a punctured key; call this $K^{\mathsf{punc}}_{i,j}$. Next $\mathcal{B}$ runs Step 5 to Step 12 on the left hand side of Figure 17 and passes $(\mathsf{PP}_j)_{j\notin I}$ (see Step 12) to $\mathcal{A}$. It handles encryption and key generation queries as follows. Key generation queries are handled as in $\mathcal{O}_{\mathsf{kg}}$ in Figure 17. On an encryption query $(i', \mathbf{x}_0, \mathbf{x}_1)$ with $i' \neq i$, the query is handled the same as $\mathcal{O}_{\mathsf{enc}}$ in Figure 17. On an encryption query $(i, \mathbf{x}_0, \mathbf{x}_1)$, the query is handled the same as $\mathcal{O}_{\mathsf{enc}}$ in Figure 17 except Step 2 of $\mathcal{O}_{\mathsf{enc}}$ where r is computed. To obtain r, the PRF evaluation oracle is queried at the point $j \parallel \mathsf{EK}_{i,j} \parallel T$ where T is derived in Step 1 of $\mathcal{O}_{\mathsf{enc}}$. Now if the PRF challenger's bit is 0, the string r will be computed using the PRF and $\mathcal{B}$ perfectly simulates $5_{j,3}, i-1$. On the other hand, if the PRF challenger's bit is 1, the string r will be uniformly random because the queried evaluation point begins with the challenge prefix j (i.e. the PRF is punctured at that point) and so $\mathcal{B}$ perfectly simulates Hybrid $5_{j,3}, i$.

Hybrid $\mathbf{5_{j,4}}$: In this Hybrid, the protocol $\mathsf{GenKeys}''''$ is modified further so that for key j, $\mathsf{MIFE.Setup}$ or $\mathsf{MIFE.KeyGen}$ are not invoked. Rather, the output of the $\mathsf{MIFE.KeyGen}$ algorithm is hardwired and output for key j. The hardwired value is exactly the same as in the previous hybrid, hence indistinguishability holds by security of MPC. In more detail, the MPC simulator receives identical inputs in both hybrids and hence produces indistinguishable outputs in the two hybrids.

Hybrid $\mathbf{5_{j,5}}$: In this hybrid, we switch the hardwired MIFE CTs within the FE CTs to use bit $b = 1$. Indistinguishability follows from MIFE security. In more detail, we fix query j. For party $i \in [n]$ and ciphertext query $k \in [Q_c]$, we construct a reduction which plays against the MIFE challenger as below. The reduction computes everything as in the previous hybrid except that the hardwired ciphertexts for the j^{th} copy of MIFE, which it receives from the MIFE challenger as below:

1. The reduction requests for MIFE key corresponding to function f_j which it receives. It hardwires this into functionality $\mathsf{GenKeys}$.
2. For $i \in [n]$, $k \in [Q_c]$, the challenge ciphertexts for party i are set as $(\mathbf{x}_{0,i,k}, T_{i,k}, K^{\mathsf{punc}}_{i,j})$ and $(\mathbf{x}_{1,i,k}, T_{i,k}, K^{\mathsf{punc}}_{i,j})$.
3. The reduction receives $\psi_{i,j,k}$ for $i \in [n]$ and $k \in [Q_c]$. It hardwires these into the FE ciphertexts as discussed above.

It is evident that if $b = 0$ then we are in Hybrid $\mathbf{5_{j,4}}$ and if $b = 1$ we are in Hybrid $\mathbf{5_{j,5}}$. Thus, an adversary that distinguishes between these two hybrids can be used to break the security of the j^{th} MIFE scheme.

Now that the bit b has been switched for query j, we roll back our changes. Arguments of indistinguishability are analogous to the above and are skipped.

Hybrid $\mathbf{5_{j,6}}$: Change $\mathsf{GenKeys}''$ to invoke $\mathsf{MIFE.Setup}$ and $\mathsf{MIFE.KeyGen}$ as before.

Hybrid $\mathbf{5_{j,7}}$: Switch randomness in the hardwired CT back to PRF randomness.

Hybrid $\mathbf{5_{j,8}}$: Switch punctured PRF key back to normal PRF key.

Hybrid $\mathbf{5_{j,9}}$: Increment $\mathsf{Trap.index}$ by 1. At this point, for key j, we have $\mathsf{ctr_j} < \mathsf{Trap_{index}}$, hence by the design of the ReEnc algorithm, we have that the bit $b = 1$ is used for MIFE encryption. This is indistinguishable from the previous hybrid by security of FE because decryption values are exactly the same in both the hybrids.

Hybrid $\mathbf{5_{j+1,1}}$: This Hybrid is analogous to Hybrid $\mathbf{5_{j,1}}$. Indistinguishability follows by security of FE as discussed above. Finally, in Hybrid $\mathbf{5_{Q_k,9}}$, all the keys are outputting MIFE CTs corresponding to $b = 1$.

Hybrid 6: In this hybrid, use message corresponding to $b = 1$ and $\mathsf{mode} = R$. Again, indistinguishability follows by security of FE since the outputs are the same.

Hybrid 7: Undo the changes made in Hybrid 4, namely the algorithm $\mathsf{GenKeys}''$ invoked for query index j for all $j \in [Q_k]$ is modified to invoke ReEnc with parameter $\mathsf{ctr} = \bot$. Indistinguishability follows analogously to the transition between Hybrid 3 and Hybrid 4.

Hybrid 8: Undo the changes made in Hybrid 3. Specifically, we generate y (the target output passed to $\widetilde{\mathsf{Sim2}}$) and s (the masking value component of the partial decryption key) the same as in Hybrid 2 for each query $((i_1, \ldots, i_\ell), f)$. Indistinguishability follows analogously to the transition between Hybrid 2 and Hybrid 3.

Hybrid 9: Undo the changes made in Hybrid 2. More precisely, we replace the uniformly random strings used in the function GenKeys for the honest users i with the outputs of the PRF. Indistinguishability follows analogously to the transition between Hybrid 1 and Hybrid 2.

Hybrid 10: Undo the changes made in Hybrid 1, that is, we generate the first-round protocol messages $\rho_i^{(1)}$ (in the public parameters) and second-round protocol messages $\rho_i^{(2)}$ (in the partial decryption keys) for MPC as in the real system for each uncorrupted user $i \notin I$. Indistinguishability follows analogously to the transition between Hybrid 0 and Hybrid 1.

Hybrid 10 is the real world with bit $b = 1$. ◀

E Proof of Security for Ad Hoc MIFE for Inner Products

In this section, we provide the proof of security for our ad hoc MIFE for inner products.

▶ **Theorem 10.** *If the* MIFE *constructed by [2] is a* xxx*-IND-secure MIFE scheme and* MPC *is an* xxx*-SIM-secure 2-round MPC protocol, then our construction is* sel*-IND-secure.*

Proof. The proof follows easily from the proof of theorem 5. For simplicity, we describe the proof for the case of a single key query. The case of multiple queries is handled exactly as in the proof of theorem 5. In more detail, we define:

Hybrid 0: This is the real game in which on every encryption query $(i, \mathbf{x}_0, \mathbf{x}_1)$, $\mathbf{x}_0$ is encrypted.

Hybrid 1: Exactly as in the proof of theorem 5, the MPC transcript in this hybrid is simulated. Indistinguishability follows as in the proof of theorem 5.

Hybrid 2: In this hybrid, we switch the bit in the MIFE ciphertexts to 1. Indistinguishability follows via a reduction, in which the MIFE function key and ciphertexts are obtained from the MIFE challenger. The MIFE function key is input to the MPC simulator and the MPC transcript and MIFE ciphertexts are returned to the adversary.

To support multiple keys, we proceed as in the proof of theorem 5 and change the bit used in the MIFE encryption from 0 to 1 key by key. ◀

Unexpected Power of Random Strings

Shuichi Hirahara
National Institute of Informatics, Tokyo, Japan
s_hirahara@nii.ac.jp

Abstract

There has been a line of work trying to characterize BPP (the class of languages that are solvable by efficient *randomized* algorithms) by efficient nonadaptive reductions to the set of Kolmogorov-*random* strings: Buhrman, Fortnow, Koucký, and Loff (CCC 2010 [10]) showed that every language in BPP is reducible to the set of random strings via a polynomial-time nonadaptive reduction (irrespective of the choice of a universal Turing machine used to define Kolmogorov-random strings). It was conjectured by Allender (CiE 2012 [1]) and others that their lower bound is tight when a reduction works for every universal Turing machine; i.e., "the only way to make use of *random* strings by a nonadaptive polynomial-time algorithm is to *derandomize* BPP."

In this paper, we *refute* this conjecture under the plausible assumption that the exponential-time hierarchy does not collapse, by showing that the exponential-time hierarchy EXPH can be solved in exponential time by nonadaptively asking the oracle whether a string is Kolmogorov-random or not. In addition, we provide an exact characterization of $\mathsf{S}_2^{\mathrm{exp}}$ in terms of exponential-time-computable nonadaptive reductions to arbitrary dense subsets of random strings.

2012 ACM Subject Classification Theory of computation → Complexity classes; Theory of computation → Problems, reductions and completeness

Keywords and phrases Kolmogorov-Randomness, Nonadaptive Reduction, BPP, Symmetric Alternation

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.41

Related Version Part of the results of this paper appeared in `https://eccc.weizmann.ac.il/report/2019/025/`.

Funding *Shuichi Hirahara*: ACT-I, JST and JSPS KAKENHI Grant Number 18H04090.

Acknowledgements I thank Eric Allender, Michal Koucký, and Osamu Watanabe for helpful discussions, and thank anonymous reviewers for helpful comments. I got the ideas of this work during the joint work with Osamu Watanabe.

1 Introduction

Randomness arises everywhere in the theory of computation, and there are at least three different notions of randomness: (1) BPP, one of the most important complexity classes, is the class of languages solvable by a two-sided-error randomized polynomial-time algorithm with high probability. (2) Pseudorandom generators enable us to derandomize BPP. Its one-sided-error version is called a *hitting set generator*. (3) The notion of *Kolmogorov complexity* enables us to quantify the amount of randomness in a finite string from the perspective of compressibility. In this study, we explore the relationship existing among these different notions of "randomness" – efficient randomized algorithms, pseudorandomness, and Kolmogorov-randomness. We present the interplay among these notions of randomness, while reviewing the literature in the subsequent section.

1.1 Pseudorandomness and Kolmogorov-Randomness

The Kolmogorov complexity of a string x can informally expressed as the length of the shortest program that prints x. More formally, the Kolmogorov complexity of $x \in \{0,1\}^*$ with respect to a Turing machine U is defined as

$$\mathrm{K}_U(x) := \min\{\, |d| \mid U \text{ outputs } x \text{ on input } d \,\}.$$

We choose a "universal" Turing machine U so that the Kolmogorov complexity is smallest up to an additive constant. A Turing machine U is said to be *universal* if, for every Turing machine M, there exists a constant c such that $\mathrm{K}_U(x) \leq \mathrm{K}_M(x)+c$ for every string $x \in \{0,1\}^*$. Note that there exists a universal Turing machine because one can take a Turing machine U that, given an input (M,x), simulates a Turing machine M on input x.

Kolmogorov complexity enables us to define the notion of Kolmogorov-randomness. A string x is said to be *(Kolmogorov-)random* if there exists no shorter program that prints x. More generally, given a threshold $s\colon \mathbb{N} \to \mathbb{N}$, we say that $x \in \{0,1\}^\ell$ is s-random if $\mathrm{K}_U(x) \geq s(\ell)$. Throughout this paper, we fix any reasonable threshold s (e.g., $s(\ell) := \ell/2$) and denote by R_{K_U} the set of s-random strings, i.e., $R_{\mathrm{K}_U} := \{\, x \mid \mathrm{K}_U(x) \geq s(|x|) \,\}$.

An important property of the set R_{K_U} of random strings is that R_{K_U} is an extremely powerful distinguisher for any computable hitting set generator. More specifically, there is a simple but powerful connection between pseudorandomness and Kolmogorov-randomness: Consider any "pseudorandom" string x that is generated by a computable process from a short seed. By definition, the string x is not Kolmogorov-random; hence R_{K_U} distinguishes any pseudorandom distribution from the uniform distribution.

A function $G\colon \{0,1\}^{s(\ell)} \to \{0,1\}^\ell$ $(\, s(\ell) < \ell \,)$ is called a *hitting set generator* secure against a class $\mathcal{C}$ if there exists no $\mathcal{C}$-algorithm R that avoids G; here we say that R *avoids* G if R rejects every string in the range of G, and R accepts at least a half of inputs of length ℓ. The notion of hitting set generator naturally arises when one tries to derandomize one-sided-error randomized algorithms: For instance, if there is an efficient hitting set generator of seed length $s(\ell) = O(\log \ell)$ and secure against linear-size circuits, then by exhaustively trying all the seeds of G, one can completely derandomize one-sided-error randomized algorithms, i.e., $\mathsf{RP} = \mathsf{P}$.

What can be efficiently solved by asking the oracle R_{K_U} whether a string is random or not? As mentioned earlier, it is easy to observe that R_{K_U} avoids any computable family of functions $G = \{G_\ell\colon \{0,1\}^{s(\ell)-O(\log \ell)} \to \{0,1\}^\ell\}_{\ell \in \mathbb{N}}$.[1] More generally, any "dense" subset R of random strings R_{K_U} avoids any computable hitting set generator. (Here we say that $R \subseteq \{0,1\}^*$ is *dense* if R contains at least a half of inputs for each input length.) By exploiting this property, a line of work exhibited the power of R_{K_U} as an oracle. Allender, Buhrman, Koucký, van Melkebeek, and Ronneburger [5] proved that $\mathsf{PSPACE} \subseteq \mathsf{P}^{R_{\mathrm{K}_U}}$. Allender, Buhrman, Koucký [4] showed that $\mathsf{NEXP} \subseteq \mathsf{NP}^{R_{\mathrm{K}_U}}$. In the case of nonadaptive reductions, Buhrman, Fortnow, Koucký, and Loff [10] showed that $\mathsf{BPP} \subseteq \mathsf{P}_{\parallel}^{R_{\mathrm{K}_U}}$, where $\mathsf{P}_{\parallel}^{R_{\mathrm{K}_U}}$ denotes the class of languages solvable in polynomial time with nonadaptive oracle access to R_{K_U}.[2] Here, a reduction is said to be *nonadaptive* (also known as a truth-table or parallel-query reduction) if the queries of the reduction do not depend on previous answers from the oracle.

---

[1] Specifically, for every $\ell \in \mathbb{N}$ and seed $z \in \{0,1\}^{s(\ell)-O(\log \ell)}$, the Kolmogorov complexity of $G_\ell(z)$ is less than $s(\ell)$; thus $G_\ell(z) \notin R_{\mathrm{K}_U}$. On the other hand, a simple counting argument shows that $|R_{\mathrm{K}_U} \cap \{0,1\}^\ell| \geq 2^{\ell-1}$ for every ℓ.

[2] The subscript $\parallel$ stands for parallel queries.

1.2 BPP and Kolmogorov-Randomness

Kolmogorov-randomness and BPP originated from different disciplines. The notion of Kolmogorov-randomness usually sits within the realm of *computability theory* as R_{K_U} is not computable, whereas *complexity theory* deals with efficient computations, such as BPP. In particular, it is obvious that there is no computable upper bound on the class $\mathsf{P}_{\|}^{R_{\mathrm{K}_U}}$, since R_{K_U} itself is not computable.

Nevertheless, a surprising interdisciplinary connection between complexity classes and Kolmogorov-random strings has been uncovered in the line of work (e.g., [4, 7, 13]). When a reduction works no matter which universal Turing machine U is used in the definition of R_{K_U}, a computable upper bound can be obtained. Cai, Downey, Epstein, Lempp, and Miller [13] showed that a language reducible to R_{K_U} for every prefix-free universal Turing machine U is computable; Allender, Friedman, and Gasarch [7] showed that any computable language L that reduces to R_{K_U} for every prefix-free universal Turing machine U via a polynomial-time nonadaptive reduction is in PSPACE. Therefore:

▶ **Theorem 1** ([10, 7, 13]).

$$\mathsf{BPP} \subseteq \bigcap_U \mathsf{P}_{\|}^{R_{\mathrm{K}_U}} \subseteq \mathsf{PSPACE},$$

where the intersection is taken over all prefix-free universal Turing machines U.

For some technical reasons, the upper bounds of [7, 13] are known to hold only for "prefix-free" universal Turing machines. A similar upper bound can be obtained for plain universal Turing machines by imposing a super-constant minimum query length restriction on reductions (cf. Hirahara and Kawamura [16]). Thus, we herein focus on (plain) universal Turing machines for the sake of simplicity.

At this point, a natural question arises: What is the exact computational power of R_{K_U} under polynomial-time nonadaptive reductions? Intuitively, any polynomial-time nonadaptive reduction cannot make any use of the set of Kolmogorov-random strings of length larger than $O(\log n)$ because the Kolmogorov complexity of any query that the reduction can make on input 1^n is at most $O(\log n)$. It was argued in [3] that, intuitively, short queries to Kolmogorov-random strings would only be used as a source of pseudorandomness. Allender [1] thus conjectured that the lower bound of Theorem 1 is exactly tight, and then a fair amount of effort has been made to verify the conjecture.

▶ **Conjecture 2** ([10, 1, 6, 3, 16]). $\mathsf{BPP} = \bigcap_U \mathsf{P}_{\|}^{R_{\mathrm{K}_U}}$, *where the intersection is taken over all universal Turing machines.*

Beyond its curiosity, such a characterization of BPP might make it possible to study BPP by using the techniques from computability theory, and could be a completely new approach for resolving the $\mathsf{P} = \mathsf{BPP}$ conjecture (as speculated in [1, 6]). Moreover, Conjecture 2 is interesting from the viewpoint of the study of the Minimum Circuit Size Problem (MCSP [20]). In some technical sense, Kolmogorov complexity can be seen as the minimum size of a circuit with oracle access to the halting problem (cf. [5]); thus R_{K_U} can be regarded as a computability-theoretic analogue of MCSP. In the light of this, Conjecture 2 states non-NP-hardness of R_{K_U} under nonadaptive polynomial-time reductions (unless $\mathsf{NP} \subseteq \mathsf{BPP}$), and thus would make it possible to obtain some new insights about NP-hardness of MCSP, which has been the focus of many recent studies on MCSP (e.g. [22, 17, 18, 9, 8]).

The "evidence" in favor of Conjecture 2 seemed to be pilling up, by considering various restrictions on the reductions. For example, Allender, Buhrman, Koucký [4] showed that conjunctive nonadaptive reductions from computable languages to R_{K_U} can be simulated

in P; Hirahara and Kawamura [16] showed that some restricted variant of $\bigcap_U \mathsf{P}_{\|}^{R_{\mathrm{K}_U}}$ lies between BPP and $\Sigma_2^{\mathrm{p}} \cap \Pi_2^{\mathrm{p}} \cap \mathsf{P/poly}$. We refer interested readers to the survey of Allender [2] for detailed background on MCSP and R_{K_U}.[3]

1.3 Our Results: Unexpected Power of Random Strings

In this work, we disprove Conjecture 2 under the plausible assumption that the exponential-time hierarchy does not collapse to BPEXP, by presenting unexpected power of R_{K_U}. Not only do we disprove Conjecture 2 itself, but also argue that the intuition of [3] is not correct.

Our main new insight is to consider a tally language in $\bigcap_U \mathsf{P}_{\|}^{R_{\mathrm{K}_U}}$. Recall that the intuition behind Conjecture 2 is that

1. On input 1^n, any long query q to R_{K_U} is answered as "No," since a nonadaptive polynomial-time machine cannot make any query whose Kolmogorov complexity is larger than $O(\log n)$, and that
2. Any query of length $O(\log n)$ *should be* useful only as a source of "pseudorandomness" [3]. We interpret this as follows: Short queries to R_{K_U} should be replaceable with queries to any oracle R that avoids any computable hitting set generator (i.e., R is a dense subset of random strings).

In order to closely examine these intuitions, let us focus on tally languages $L \subseteq \{1^n\}$. By a standard padding argument, this is essentially equivalent to considering an exponential-time analogue of Conjecture 2. More specifically, if Conjecture 2 is true, then a padding argument shows that

$$\mathsf{BPEXP} = \bigcap_U \mathsf{EXP}_{\|}^{R_{\mathrm{K}_U}}$$

is also true.[4] In order to examine the latter intuition, we also focus on an arbitrary oracle R that avoids U. Specifically, Conjecture 2 implies that the subclass $\bigcap_R \mathsf{EXP}_{\|}^{R}$ of $\bigcap_U \mathsf{EXP}_{\|}^{R_{\mathrm{K}_U}}$ is also contained in BPEXP, i.e.,

$$\bigcap_R \mathsf{EXP}_{\|}^{R} \subseteq \mathsf{BPEXP},$$

where the intersection is taken over all dense subsets R of random strings; thus, we first study the subclass $\bigcap_R \mathsf{EXP}_{\|}^{R}$.

We completely settle what can be solved by a nonadaptive exponential-time reduction to dense subsets R of random strings.

▶ **Theorem 3** (Characterizing $\mathsf{S}_2^{\mathrm{exp}}$ Using Efficient Reductions to Dense Subsets of Random Strings)**.** *For any universal Turing machine U, it holds that*

$$\mathsf{S}_2^{\mathrm{exp}} = \bigcap_R \mathsf{EXP}_{\|}^{R},$$

where the intersection is taken over all the dense subsets R of R_{K_U}. (The condition of R can be equivalently stated as follows: R avoids a universal Turing machine U regarded as a family of functions $U = \{U_\ell \colon \{0,1\}^{s(\ell)} \to \{0,1\}^{\ell}\}_{\ell \in \mathbb{N}}$.)

[3] Some unpublished results mentioned there are included in this paper.

[4] In fact, this padding argument together with [4] already shows that Conjecture 2 is false unless NEXP $\subseteq$ BPEXP, which was completely overlooked before this work. Our results significantly weaken the hypothesis.

Here $\mathsf{S}_2^{\mathrm{exp}}$ represents an exponential-time analogue of $\mathsf{S}_2^{\mathrm{p}}$ (the second level of the symmetric hierarchy [23, 14]), and is the class of languages which admit a "competing-two-prover" system: an exponential-time verifier must compute the correct answer when one of the provers presents a correct certificate.

Theorem 3 improves the result of [10] (conjectured to be tight) in two ways: First, it is known that $\mathsf{BPEXP} \subseteq \mathsf{S}_2^{\mathrm{exp}}$, and the inclusion is likely to be proper (e.g., note that $\mathsf{NEXP} \subseteq \mathsf{EXP}^{\mathsf{NP}} \subseteq \mathsf{S}_2^{\mathrm{exp}}$ [23]); thus, Theorem 3 refutes Conjecture 2 unless $\mathsf{S}_2^{\mathrm{exp}} \subseteq \mathsf{BPEXP}$. Second, unlike the proof of [10], our proof does not exploit any specific property of R_{K_U}; we only rely on the pseudorandomness property of R_{K_U} (i.e., the property that R_{K_U} is a distinguisher for any computable hitting set generator).

It is also worthy of note that our $\mathsf{S}_2^{\mathrm{exp}}$ upper bound of Theorem 3 significantly improves the previous $\mathsf{EXPSPACE}$ upper bound of Allender, Friedman, and Gasarch [7]: They proved $\bigcap_R \mathsf{EXP}_{\|}^R \subseteq \mathsf{EXPSPACE}$ as a corollary of their PSPACE upper bound of Theorem 1.

Let us take a look at the original conjecture again. In Conjecture 2, we are allowed to exploit a property of R_{K_U}, instead of just a property that an oracle avoids a computable hitting set generator. In this case, it is possible to reduce the entire exponential-time hierarchy to R_{K_U} via a nonadaptive exponential-time reduction, or even more efficiently, via a PH-Turing reduction.

▶ **Theorem 4.** *For every universal Turing machine U,*

$$\mathsf{EXPH} \subseteq \mathsf{PH}^{R_{\mathrm{K}_U}} \subseteq \mathsf{EXP}_{\|}^{R_{\mathrm{K}_U}}.$$

Theorem 4 shows that the class $\bigcap_U \mathsf{EXP}_{\|}^{R_{\mathrm{K}_U}}$ is much closer to the $\mathsf{EXPSPACE}$ upper bound of [7] than previously conjectured. To be specific, it implies that Conjecture 2 is false unless $\mathsf{EXPH} \subseteq \mathsf{BPEXP}$.

1.4 Proof Ideas

We briefly mention our proof techniques for proving the lower bounds of Theorem 3 and Theorem 4. It is known that the halting problem is reducible to dense subsets R of random strings via a *nonuniform* polynomial-time reduction [5]. Therefore, by exhaustively searching polynomial-size R-oracle circuits (in exponential time), we can find a "succinct witness" for $\mathsf{S}_2^{\mathrm{exp}}$, whose correctness can be verified in exponential time. In order to extend this argument to EXPH, recall that EXPH can be regarded as a constant-round two-player game whose winner can be computed in exponential time. A winning strategy of round $k+1$ must depend on previous strategies (R_{K_U}-oracle circuits) of rounds $1, \cdots, k$. Using the fact that an R_{K_U}-oracle circuit can be evaluated with oracle access to the halting problem, it can be argued that there exists a polynomial-size R-oracle circuit which encodes a winning strategy for round $k+1$.

In order to prove the $\mathsf{S}_2^{\mathrm{exp}}$ upper bound of Theorem 3, we simulate an exponential-time reduction M by an $\mathsf{S}_2^{\mathrm{exp}}$ algorithm as follows: The ith competing prover sends a set $\bar{R}_i$ for each $i \in \{1,2\}$; the honest prover sends the set $\{0,1\}^* \setminus R_{\mathrm{K}_U}$ of nonrandom strings. While we cannot know which of $\bar{R}_1$ or $\bar{R}_2$ is correct, $\{0,1\}^* \setminus (\bar{R}_1 \cup \bar{R}_2)$ is guaranteed to be a dense subset of R_{K_U}, to which the reduction M works correctly.

The rest of this paper is organized as follows. After reviewing some background in Section 2, Theorem 3 is proved in Section 3. Theorem 4 is proved in Section 4. In Appendix A, the $\mathsf{EXP}_{\|}$ reduction of Theorem 3 is improved.

2 Preliminaries

We identify a language $L \subseteq \{0,1\}^*$ with its characteristic function $L\colon \{0,1\}^* \to \{0,1\}$. Let $[n] := \{1,\dots,n\}$ for $n \in \mathbb{N}$. We say that $R \subseteq \{0,1\}^*$ is *dense* if $|R \cap \{0,1\}^n| \geq 2^{n-1}$ for every $n \in \mathbb{N}$. For a Boolean function $f\colon \{0,1\}^n \to \{0,1\}$, we denote by $\mathtt{tt}(f)$ the truth table of f, i.e., the concatenation of $f(x)$ for all $x \in \{0,1\}^n$ in the lexicographical order. We often identify a Boolean circuit C with the function computed by C.

2.1 On the Threshold of Kolmogorov-Random Strings

Recall that a string $x \in \{0,1\}^*$ is said to be $s(\text{-})$-*random* if $\mathrm{K}_U(x) \geq s(|x|)$. Throughout this paper, we fix any reasonable threshold $s(\text{-})$ of randomness so that there exists a universal constant $\epsilon > 0$ satisfying $n^\epsilon \leq s(n) \leq n-2$ for any $n \in \mathbb{N}$, and denote by R_{K_U} the set of $s(\text{-})$-random strings, i.e., $R_{\mathrm{K}_U} := \{\, x \in \{0,1\}^* \mid \mathrm{K}_U(x) \geq s(|x|) \,\}$. The upper bound of the threshold is for assuming that the set of random strings is sufficiently dense:

▶ **Fact 5.** *$|\{0,1\}^n \setminus R_{\mathrm{K}_U}| \leq 2^{n-2}$ for any $n \in \mathbb{N}$.*

Proof. The number of nonrandom strings is bounded by the number of programs of length less than $s(n)$, which is at most $\sum_{i=0}^{s(n)-1} 2^i \leq 2^{s(n)} \leq 2^{n-2}$. ◀

2.2 Symmetric Alternation

$\mathsf{S}_2^{\mathrm{p}}$ denotes the class of languages that admit a competing-two-prover system. More formally:[5]

▶ **Definition 6** ([14, 23])**.** *$\mathsf{S}_2^{\mathrm{p}}$ is the class of languages L such that there exist a polynomial $p(n) = n^{O(1)}$ and a polynomial-time algorithm V such that, for every input $x \in \{0,1\}^*$,*

1. *$\exists y \in \{0,1\}^{p(|x|)}, \forall z \in \{0,1\}^{p(|x|)},\ V(x,y,z) = L(x)$, and*
2. *$\exists z \in \{0,1\}^{p(|x|)}, \forall y \in \{0,1\}^{p(|x|)},\ V(x,y,z) = L(x)$.*

$\mathsf{S}_2^{\mathrm{exp}}$ is an exponential-time analogue of $\mathsf{S}_2^{\mathrm{p}}$: the definition of $\mathsf{S}_2^{\mathrm{exp}}$ is obtained by allowing $p(n)$ to be exponential in the definition above (i.e., $p(n) = 2^{n^{O(1)}}$). For an oracle A, the relativized version of $\mathsf{S}_2^{\mathrm{p}}$ is denoted by S_2^A; that is, S_2^A is the class of languages that admit a competing-two-prover system with a polynomial-time A-oracle verifier.

2.3 Exponential-Time Hierarchy

EXPH denotes the exponential-time analogue of the polynomial-time hierarchy (PH). That is, a language L is in EXPH if and only if there exist a constant k, an exponential bound $e(n) = 2^{n^{O(1)}}$ and a polynomial-time algorithm M such that, for every input $x \in \{0,1\}^*$,

$$x \in L \iff \exists y_1, \forall y_2, \exists y_3, \forall y_4, \cdots, \exists y_k,\ M(x, y_1, \cdots, y_k) = 1,$$

where $y_i \in \{0,1\}^{e(n)}$ for any $i \in [k]$.

[5] We follow the definition of Canetti [14], which is equivalent to the definition of [23] (cf. [11]).

3 Reductions to Dense Subsets of Random Strings

In this section, we give the characterization of $\mathsf{S}_2^{\mathrm{exp}}$ by exponential-time nonadaptive reductions to arbitrary dense subsets of random strings.

▶ **Reminder of Theorem 3.** Fix any universal Turing machine U.

- *Lower bound:* $\mathsf{S}_2^{\mathrm{exp}} \subseteq \mathsf{EXP}_{\|}^{R}$ for any dense subset R of R_{K_U}.
- *Upper bound:* $\{ L \subseteq \{0,1\}^* \mid L \in \mathsf{EXP}_{\|}^{R}$ for every dense $R \subseteq R_{\mathrm{K}_U} \} \subseteq \mathsf{S}_2^{\mathrm{exp}}$.

3.1 Lower Bound

In order to prove the lower bound on the complexity class $\bigcap_R \mathsf{EXP}_{\|}^{R}$, we use the fact that the halting problem HALT is reducible to any dense subset of random strings via a polynomial-size oracle circuit. The halting problem HALT is the problem of taking as input a description of a Turing machine M and deciding whether M halts.

▶ **Theorem 7** ([5]). $\mathsf{HALT} \in \mathsf{P}^{R}/\mathsf{poly}$ *for any dense subset R of random strings.*

Proof Sketch. It was shown in [5, Corollary 32] that HALT is reducible to R_{K_U} via a nonadaptive P/poly reduction. The proof only uses the fact that R_{K_U} is a distinguisher for some hitting set generator; thus HALT is also reducible to any dense subset R of random strings. ◀

Proof of the Lower Bound of Theorem 3. Let $L \in \mathsf{S}_2^{\mathrm{exp}}$ and V be a polynomial-time verifier associated with L that takes certificates of length $e(n) = 2^{n^{O(1)}}$ on inputs of length n. We claim that $L \in \mathsf{EXP}_{\|}^{R}$ for every dense subset R of random strings. Our $\mathsf{EXP}_{\|}^{R}$ algorithm M exhaustively searches "succinct witnesses", i.e., circuits that encode a correct certificate.

Specifically, for some polynomial $q(n)$ that is chosen later, let $\mathcal{C}_n$ be the set of all the $q(n)$-size $\log e(n)$-input oracle circuit. On input $x \in \{0,1\}^*$, the algorithm M exhaustively searches all the circuits $Y, Z \in \mathcal{C}_{|x|}$ and accepts if and only if there exists an oracle circuit $Y \in \mathcal{C}_{|x|}$ such that, for every oracle circuit $Z \in \mathcal{C}_{|x|}$, $V(x, \mathtt{tt}(Y^R), \mathtt{tt}(Z^R))$ accepts. Since the number $|\mathcal{C}_n|$ of polynomial-size oracle circuits is at most exponential in n, M halts in exponential time. Moreover, M is a nonadaptive reduction to R because the length of any query is bounded by some polynomial $q(n)$ (thus, in exponential time, one can ask every possible query beforehand).

We claim the correctness of M. By the definition of $\mathsf{S}_2^{\mathrm{exp}}$, for any input $x \in \{0,1\}^*$,

1. if $x \in L$, there exists a certificate $y \in \{0,1\}^{e(n)}$ such that $V(x,y,z) = 1$ for every $z \in \{0,1\}^{e(n)}$, and
2. if $x \notin L$, there exists a certificate $z \in \{0,1\}^{e(n)}$ such that $V(x,y,z) = 0$ for every $y \in \{0,1\}^{e(n)}$.

Consider the function η that maps (x,i) to the ith bit of the lexicographically first certificate $y \in \{0,1\}^{e(|x|)}$ such that $V(x,y,z) = 1$ for every $z \in \{0,1\}^{e(|x|)}$ (if exists), where $i \in \{0,1\}^{\log e(|x|)}$. Since η is computable, by Theorem 7, there exists a polynomial-size oracle circuit Y' such that $Y'^R(x,i) = \eta(x,i)$ for any (x,i). Therefore, for every input $x \in L$, by fixing x we obtain a polynomial-size circuit $Y_x := Y'(x,\text{-})$ such that $V(x, \mathtt{tt}(Y_x^R), z) = 1$ for every z. By choosing the bound $q(n)$ large enough, we obtain $Y_x \in \mathcal{C}$, and thus the algorithm M accepts. Similarly, one can prove that, for every input $x \notin L$, there exists a circuit $Z_x \in \mathcal{C}$ such that $V(x, y, \mathtt{tt}(Z_x^R)) = 0$ for every y. ◀

3.2 Upper Bound

Next, we prove the $\mathsf{S}_2^{\mathrm{exp}}$ upper bound on $\bigcap_R \mathsf{EXP}_{\|}^R$. Take an arbitrary language $L \in \bigcap_R \mathsf{EXP}_{\|}^R$. This means that, for every dense subset R of random strings, there exists an exponential-time oracle machine M_R such that M_R solves L given oracle access to R. That is, the reduction M_R is allowed to depend on the oracle R, which makes it hard to simulate the reduction. We first show, by using a diagonalization argument, that there exists a *single* machine that works for every oracle R.

▶ **Lemma 8.** *For any language L, the following are equivalent:*

1. *$L \in \bigcap_R \mathsf{EXP}_{\|}^R$, where the intersection is taken over all dense subsets R of R_{K_U}.*
2. *There exists an exponential-time nonadaptive oracle machine M such that, for every dense subset R of R_{K_U}, for every input $x \in \{0,1\}^*$, $M^R(x) = L(x)$.*

Proof. The direction from the second item to the first item is obvious. We prove below the contrapositive of the other direction.

Suppose that, for any exponential-time nonadaptive oracle machine M, there exists a dense subset R of R_{K_U} such that $M^R(x) \neq L(x)$ for some $x \in \{0,1\}^*$. We claim that there exists a *single* dense subset $R \subseteq R_{\mathrm{K}_U}$ such that $L \notin \mathsf{EXP}_{\|}^R$.

To this end, let $\{M_e\}_{e\in\mathbb{N}}$ be the enumeration of all exponential-time nonadaptive oracle machines. We will construct some dense subset $R_e \subseteq R_{\mathrm{K}_U}$ and input x_e (and $\ell_e \in \mathbb{N}$) by induction on $e \in \mathbb{N}$, so that M_e fails to compute L on input x_e, given oracle access to R_{e+1}; then we will define $R := \bigcup_{e\in\mathbb{N}} R_e$. Let us start with $R_0 := \varnothing$ and $\ell_0 := 0$.

At stage $e \in \mathbb{N}$, we claim that there exists some dense subset $R'_{e+1} \subseteq R_{\mathrm{K}_U}$ and some input $x_e \in \{0,1\}^*$ such that

- $M_e^{R'_{e+1}}(x_e) \neq L(x_e)$, and
- $q \in R_e$ if and only if $q \in R'_{e+1}$ for any string q of length $< \ell_e$.

Indeed, for any oracle Q, let $Q' := \{\, q \in Q \mid |q| \geq \ell_e \,\} \cup \{\, q \in R_e \mid |q| < \ell_e \,\}$. Consider an exponential-time nonadaptive oracle machine M'_e such that $M_e'^Q$ simulates $M_e^{Q'}$; that is, M'_e is hardwired with the set $\{\, q \in R_e \mid |q| < \ell_e \,\}$, and simulates M_e and answer any query q of length $< \ell_e$ by using the hardwired information. By our assumption, there exists some dense subset $\hat{R}_{e+1} \subseteq R_{\mathrm{K}_U}$ such that $M_e'^{\hat{R}_{e+1}}(x_e) \neq L(x_e)$ for some $x_e \in \{0,1\}^*$; by the definition of M'_e, we obtain $M_e^{R'_{e+1}}(x_e) \neq L(x_e)$ for $R'_{e+1} := \{\, q \in \hat{R}_{e+1} \mid |q| \geq \ell_e \,\} \cup \{\, q \in R_e \mid |q| < \ell_e \,\}$, which is again a dense subset of R_{K_U}. This completes the proof of the claim above. Now define $\ell_{e+1} \in \mathbb{N}$ as a large enough integer so that $\ell_{e+1} \geq \ell_e$ and the machine M_e on input x_e does not query any string of length $\geq \ell_{e+1}$, and define an oracle $R_{e+1} := \{\, q \in R'_{e+1} \mid |q| < \ell_{e+1} \,\}$, which completes the construction of stage $e \in \mathbb{N}$.

Define $R := \bigcup_{e\in\mathbb{N}} R_e$, which is a dense subset of R_{K_U} by the construction above. By the choice of $(\ell_e)_{e\in\mathbb{N}}$, we have

$$M_e^R(x_e) = M_e^{R_{e+1}}(x_e) \neq L(x_e),$$

for every exponential-time nonadaptive oracle machine M_e. Thus $L \notin \mathsf{EXP}_{\|}^R$. ◀

We are ready to present the $\mathsf{S}_2^{\mathrm{exp}}$ algorithm that simulates an exponential-time nonadaptive reduction to dense subsets of random strings, which completes the proof of Theorem 3.

Proof of the Upper Bound of Theorem 3. Take any language $L \in \bigcap_R \mathsf{EXP}_{\|}^R$. By Lemma 8, there exists an exponential-time nonadaptive oracle machine M such that, for every dense subset R of R_{K_U}, for every input $x \in \{0,1\}^*$, $M^R(x) = L(x)$. Our goal is to simulate M in $\mathsf{S}_2^{\mathrm{exp}}$.

First, we can assume, without loss of generality, that the length of any query that M makes on input x is bounded by some polynomial $|x|^{O(1)}$. Indeed, since M is a nonadaptive reduction, the set of all the queries $Q(x)$ that M makes on input x can be computed in exponential time; thus, for each query $q \in Q(x)$, its Kolmogorov complexity $\mathrm{K}_U(q)$ is at most $|x| + \log |Q(x)| + O(1) \leq |x|^{O(1)}$, since q is described by x and the index of q in $Q(x)$. Therefore, for any query q whose length is larger than $|x|^{O(1)}$, the answer from the oracle R is "No" because $q \notin R_{\mathrm{K}_U}$ $(\supseteq R)$.

Now we present an $\mathsf{S}_2^{\mathrm{exp}}$ algorithm that simulate M. The idea is that two competing provers send the set of nonrandom strings. Given two possible sets of nonrandom strings $\bar{R}_1, \bar{R}_2 \subseteq \{0,1\}^*$ (one of which is guaranteed to be correct), $R := \{0,1\}^* \setminus (\bar{R}_0 \cup \bar{R}_1)$ is a subset of R_{K_U}. Moreover, since the number of nonrandom strings is small, the set R is dense enough. Details follow.

Let $p(n)$ be a polynomial that upper-bounds the length of any query that M makes on inputs of length n. Our $\mathsf{S}_2^{\mathrm{exp}}$ algorithm operates as follows: Fix any input x of length n. The ith prover ($i \in \{1,2\}$) sends, for each $\ell \leq p(n)$, a subset $\bar{R}_{i,\ell} \subseteq \{0,1\}^\ell$ of size at most $2^{\ell-2}$; an honest prover sets $\bar{R}_{i,\ell} := \{0,1\}^\ell \setminus R_{\mathrm{K}_U}$. Define $\bar{R}_i := \bigcup_{\ell \leq p(n)} \bar{R}_{i,\ell}$. Note that such subsets can be encoded as a string of exponential length. The verifier sets $R := \{0,1\}^* \setminus (\bar{R}_1 \cup \bar{R}_2)$, and accept if and only if $M^R(x)$ accepts.

We claim the correctness of the $\mathsf{S}_2^{\mathrm{exp}}$ algorithm. Assume that the i^*th prover is honest. For each length $\ell \leq p(n)$ and $i \in \{1,2\}$, we assumed that $|\bar{R}_{i,\ell}| \leq 2^{\ell-2}$; thus $|\bar{R}_{1,\ell} \cup \bar{R}_{2,\ell}| \leq 2^{\ell-1}$, from which it follows that R is dense. Moreover, since the number of nonrandom strings of length ℓ is at most $2^{\ell-2}$ (Fact 5)[6], the i^*th prover can set $\bar{R}_{i^*,\ell} := \{0,1\}^\ell \setminus R_{\mathrm{K}_U}$. Define $R' := \{\, q \in R \mid |q| \leq p(n) \,\} \cup \{\, q \in R_{\mathrm{K}_U} \mid |q| > p(n) \,\}$. Then, from the argument above, R' is a dense subset of R_{K_U}, and hence $M^{R'}(x) = L(x)$. Since M does not make any query of length $> p(n)$, it follows that $M^R(x) = M^{R'}(x) = L(x)$. ◀

In Appendix A, we will mention that the reducibility notion of Theorem 3 can be significantly improved from $\mathsf{EXP}_{\|}$ to $\mathsf{S}_2^{\mathrm{p}}$ (at the cost of a slight loss in the lower bound).

4 The Unexpected Power of Kolmogorov-Random Strings

In this section, we show that every language in the exponential-time hierarchy is reducible to the set of Kolmogorov-random strings under PH reductions.

▶ **Reminder of Theorem 4.** For every universal Turing machine U,

$$\mathsf{EXPH} \subseteq \mathsf{PH}^{R_{\mathrm{K}_U}}.$$

For the purpose of refuting Allender's conjecture (Conjecture 2), it is enough to show the weaker statement that $\mathsf{EXPH} \subseteq \mathsf{EXP}_{\|}^{R_{\mathrm{K}_U}}$. Indeed, we observe that a simple padding argument essentially refutes Conjecture 2:

▶ **Corollary 9.** $\mathsf{BPP} \neq \bigcap_U \mathsf{P}_{\|}^{R_{\mathrm{K}_U}}$ *unless* $\mathsf{EXPH} \subseteq \mathsf{BPEXP}$*, where the intersection is taken over all universal Turing machines.*

Proof. Assume that $\bigcap_U \mathsf{P}_{\|}^{R_{\mathrm{K}_U}} \subseteq \mathsf{BPP}$. Then by the standard padding argument, we also obtain $\bigcap_U \mathsf{EXP}_{\|}^{R_{\mathrm{K}_U}} \subseteq \mathsf{BPEXP}$. (Specifically, take any language $L \in \bigcap_U \mathsf{EXP}_{\|}^{R_{\mathrm{K}_U}}$, and define a padded version $L' := \{\, x 10^{2^{p(|x|)}} \mid x \in L \,\}$ for some large enough polynomial p so that $L' \in \bigcap_U \mathsf{P}_{\|}^{R_{\mathrm{K}_U}}$. By the assumption, we have $L' \in \mathsf{BPP}$, which implies that $L \in \mathsf{BPEXP}$.) It follows from Theorem 4 that $\mathsf{EXPH} \subseteq \bigcap_U \mathsf{EXP}_{\|}^{R_{\mathrm{K}_U}} \subseteq \mathsf{BPEXP}$. ◀

[6] This is because of our choice of the threshold for Kolmogorov-randomness.

Now we proceed to a proof of Theorem 4. We first observe that any language computable with oracle access to HALT is in $\mathsf{P}^{\mathsf{HALT}}$.

▶ **Lemma 10.** *Let M be any oracle machine that, on inputs of length n, halts in finite steps and makes a query of length at most $n^{O(1)}$. Then the language decided by M^{HALT} is in $\mathsf{P}^{\mathsf{HALT}}$.*

Proof Sketch. The proof is essentially the same with [5, Theorem 27], and thus we just give a proof sketch. The idea is to decide M^{HALT} in the following two steps: First, by using a binary search and oracle access to HALT, one can decide the number of all the strings in HALT of length at most $n^{O(1)}$ in polynomial time. Then, given as advice the number of strings in HALT of length at most $n^{O(1)}$, the computation of M^{HALT} becomes now computable, and hence it reduces to HALT. ◀

While the ingredients above are enough to obtain EXP reductions, in order to obtain PH reductions, we make use of the efficient proof system of PH given by Kiwi, Lund, Spielman, Russell, and Sundaram [21]. For simplicity, we state their results in the case of the number of alternation is 2, but their results hold for every constant number of alternation. We also state their results in terms of Σ_2^{EXP} instead of Σ_2^{P}.

▶ **Theorem 11** (Kiwi, Lund, Spielman, Russell, and Sundaram [21])**.** *For every language L in Σ_2^{EXP}, there exists a randomized polynomial-time verifier such that,*

1. *for every input $x \in L$, there exists an oracle A such that for any oracle B, $V^{A,B}(x)$ accepts with probability 1, and*
2. *for every input $x \notin L$, for all oracles A, there exists an oracle B, $V^{A,B}(x)$ accepts with probability at most $\frac{1}{2}$.*

We are now ready to present a proof of Theorem 4.

Proof of Theorem 4. The main idea is that, given oracle access to the set of random strings, Theorem 7 tells us that there is a "succinct witness" for any exponential-time computation. However, unlike the proof of Theorem 3, here we need to claim that there exists a succinct witness that encodes some winning strategy of two player games, which may depend on a R_{K_U}-oracle circuit that encodes the opponent's strategy. For simplicity, we will only give a detailed proof for $\Sigma_2^{\mathsf{EXP}} \subseteq \mathsf{PH}^{R_{\mathrm{K}_U}}$, since it is straightforward to extend the proof.

First, we present a proof of $\Sigma_2^{\mathsf{EXP}} \subseteq \mathsf{EXP}_{\|}^{R_{\mathrm{K}_U}}$. Let V be a polynomial-time verifier for $L \in \Sigma_2^{\mathsf{EXP}}$, and $e(n) = 2^{n^{O(1)}}$ be an exponential bound such that for every input x of length n, it holds that $x \in L$ if and only if there exists a certificate y of length $e(n)$ such that $V(x, y, z)$ accepts for all z of length $e(n)$; Note that V runs in time $2^{n^{O(1)}}$. We regard the computation as a game between the first player y and the second player z.

Our $\mathsf{EXP}_{\|}^{R_{\mathrm{K}_U}}$ algorithm operates as follows: Let $s_Y(n)$ and $s_Z(n)$ be some polynomials specified later. Given input x of length n, the algorithm accepts if and only if there exists an oracle circuit Y of size $s_Y(n)$ such that $V(x, \mathtt{tt}(Y^{R_{\mathrm{K}_U}}), \mathtt{tt}(Z^{R_{\mathrm{K}_U}}))$ accepts for all oracle circuits Z of size $s_Z(n)$, where $\mathtt{tt}(Y^{R_{\mathrm{K}_U}})$ denotes the truth table of the function computed by $Y^{R_{\mathrm{K}_U}}$; the algorithm checks this condition by an exhaustive search. Since there are at most exponentially many circuits of polynomial size, this algorithm runs in exponential time.

We claim the correctness of the algorithm. Fix any input $x \in L$ of length n. In this case, the correctness readily follows from the fact that there exists a succinct witness under the oracle R_{K_U}: Indeed, let y_x be the lexicographically first certificate for $x \in L$. Since each bit of y_x is decidable (in the sense that the language $\{\, (x, i) \mid \text{the } i\text{th bit of } y_x \text{ is } 1 \,\}$ is decidable), by Theorem 7, there exists an oracle circuit Y of size $s_Y(n) := \mathsf{poly}(n, \log |y_x|) = \mathsf{poly}(n)$ such that $\mathtt{tt}(Y^{R_{\mathrm{K}_U}}) = y_x$. Thus $V(x, y_x, \mathtt{tt}(Z^{R_{\mathrm{K}_U}}))$ accepts no matter how the adversarial circuit Z is chosen.

Now fix any input $x \notin L$ of length n. This case requires a more delicate argument. Here we need to claim that, for every circuit Y of size $s_Y(n)$, there exists a circuit Z that encodes a strategy that beats the strategy of $\mathtt{tt}(Y^{R_{\mathrm{K}^U}})$. The point is that, given any circuit Y, the lexicographically first winning strategy of the second player is computable with oracle access to HALT. Indeed, let $z_{x,Y}$ denote the lexicographically first string such that $V(x, \mathtt{tt}(Y^{R_{\mathrm{K}^U}}), z)$ rejects. Consider the language $L' := \{ (x, Y, i) \mid \text{the } i\text{th bit of } z_{x,Y} \text{ is } 1 \}$. Since R_{K^U} is reducible to HALT, the language L' is computable with oracle access to HALT. By Lemma 10, $L' \in \mathsf{P}^{\mathsf{HALT}}$; thus by Theorem 7, we obtain $L' \in \mathsf{P}^{R_{\mathrm{K}^U}}/\mathsf{poly}$. This means that for every circuit Y there exists a circuit Z_Y of size $s_Z(n) := \mathsf{poly}(n, |Y|, \log |z_{x,Y}|) = \mathsf{poly}(n, s_Y(n))$ such that $\mathtt{tt}(Z_Y^{R_{\mathrm{K}^U}}) = z_{x,Y}$. Thus, our algorithm rejects. This completes the proof of $\Sigma_2^{\mathsf{EXP}} \subseteq \mathsf{EXP}_{\|}^{R_{\mathrm{K}^U}}$.

In order to extend the proof above to $\Sigma_{2k}^{\mathsf{EXP}}$ for every constant k, we modify the $\mathsf{EXP}^{R_{\mathrm{K}^U}}$ algorithm so that it checks, given input x of length n, whether $\exists$ a circuit C_1 of size $s_1(n)$, $\forall$ a circuit C_2 of size $s_2(n)$, $\cdots$, $\forall$ a circuit C_{2k} of size $s_{2k}(n)$ such that a verifier $V(x, \mathtt{tt}(C_1^{R_{\mathrm{K}^U}}), \cdots, \mathtt{tt}(C_{2k}^{R_{\mathrm{K}^U}}))$ accepts, where $s_1(n), \cdots, s_{2k}(n)$ are some appropriately chosen polynomials.

We now explain how to reduce the complexity of the EXP reduction to PH. For simplicity, we again focus on a proof of $\Sigma_2^{\mathsf{EXP}} \subseteq \mathsf{PH}^{R_{\mathrm{K}^U}}$. Note that, in the proof above, the bottleneck of the computation is the evaluation of $V(x, \mathtt{tt}(Y^{R_{\mathrm{K}^U}}), \mathtt{tt}(Z^{R_{\mathrm{K}^U}}))$, where V runs in time $2^{|x|^{O(1)}}$. We replace V with the randomized polynomial-time verifier of Theorem 11; then we obtain the following $\Sigma_2^{R_{\mathrm{K}^U}}$ algorithm: Existentially guess a circuit Y of size at most $s_Y(n)$, and universally guess a circuit Z of size at most $s_Z(n)$ as well as a coin flip for V, Then accept if and only if $V^{Y,Z}(x)$ accepts on the guessed coin flip sequence. ◀

References

1. Eric Allender. Curiouser and Curiouser: The Link between Incompressibility and Complexity. In *Proceedings of the 8th Conference on Computability in Europe (CiE)*, pages 11–16, 2012. doi:10.1007/978-3-642-30870-3_2.
2. Eric Allender. The Complexity of Complexity. In *Computability and Complexity - Essays Dedicated to Rodney G. Downey on the Occasion of His 60th Birthday*, pages 79–94, 2017. doi:10.1007/978-3-319-50062-1_6.
3. Eric Allender, Harry Buhrman, Luke Friedman, and Bruno Loff. Reductions to the set of random strings: The resource-bounded case. *Logical Methods in Computer Science*, 10(3), 2014. doi:10.2168/LMCS-10(3:5)2014.
4. Eric Allender, Harry Buhrman, and Michal Koucký. What can be efficiently reduced to the Kolmogorov-random strings? *Ann. Pure Appl. Logic*, 138(1-3):2–19, 2006. doi:10.1016/j.apal.2005.06.003.
5. Eric Allender, Harry Buhrman, Michal Koucký, Dieter van Melkebeek, and Detlef Ronneburger. Power from Random Strings. *SIAM J. Comput.*, 35(6):1467–1493, 2006. doi:10.1137/050628994.
6. Eric Allender, George Davie, Luke Friedman, Samuel Hopkins, and Iddo Tzameret. Kolmogorov Complexity, Circuits, and the Strength of Formal Theories of Arithmetic. *Chicago J. Theor. Comput. Sci.*, 2013, 2013. URL: http://cjtcs.cs.uchicago.edu/articles/2013/5/contents.html.
7. Eric Allender, Luke Friedman, and William I. Gasarch. Limits on the computational power of random strings. *Inf. Comput.*, 222:80–92, 2013. doi:10.1016/j.ic.2011.09.008.
8. Eric Allender and Shuichi Hirahara. New Insights on the (Non-)Hardness of Circuit Minimization and Related Problems. *TOCT*, 11(4):27:1–27:27, 2019. doi:10.1145/3349616.

9 Eric Allender, Dhiraj Holden, and Valentine Kabanets. The Minimum Oracle Circuit Size Problem. *Computational Complexity*, 26(2):469–496, 2017. doi:10.1007/s00037-016-0124-0.

10 Harry Buhrman, Lance Fortnow, Michal Koucký, and Bruno Loff. Derandomizing from Random Strings. In *Proceedings of the Conference on Computational Complexity (CCC)*, pages 58–63, 2010. doi:10.1109/CCC.2010.15.

11 Jin-yi Cai. $\mathsf{S}_2^p \subseteq \mathsf{ZPP}^{\mathsf{NP}}$. *J. Comput. Syst. Sci.*, 73(1):25–35, 2007. doi:10.1016/j.jcss.2003.07.015.

12 Jin-yi Cai, Venkatesan T. Chakaravarthy, Lane A. Hemaspaandra, and Mitsunori Ogihara. Competing provers yield improved Karp-Lipton collapse results. *Inf. Comput.*, 198(1):1–23, 2005. doi:10.1016/j.ic.2005.01.002.

13 Mingzhong Cai, Rodney G. Downey, Rachel Epstein, Steffen Lempp, and Joseph S. Miller. Random strings and tt-degrees of Turing complete C.E. sets. *Logical Methods in Computer Science*, 10(3), 2014. doi:10.2168/LMCS-10(3:15)2014.

14 Ran Canetti. More on BPP and the Polynomial-Time Hierarchy. *Inf. Process. Lett.*, 57(5):237–241, 1996. doi:10.1016/0020-0190(96)00016-6.

15 Shuichi Hirahara. Identifying an Honest EXP^NP Oracle Among Many. In *Proceedings of the Conference on Computational Complexity (CCC)*, pages 244–263, 2015. doi:10.4230/LIPIcs.CCC.2015.244.

16 Shuichi Hirahara and Akitoshi Kawamura. On characterizations of randomized computation using plain Kolmogorov complexity. *Computability*, 7(1):45–56, 2018. doi:10.3233/COM-170075.

17 Shuichi Hirahara and Osamu Watanabe. Limits of Minimum Circuit Size Problem as Oracle. In *Proceedings of the Conference on Computational Complexity (CCC)*, pages 18:1–18:20, 2016. doi:10.4230/LIPIcs.CCC.2016.18.

18 John M. Hitchcock and Aduri Pavan. On the NP-Completeness of the Minimum Circuit Size Problem. In *Proceedings of the Conference on Foundation of Software Technology and Theoretical Computer Science (FSTTCS)*, pages 236–245, 2015. doi:10.4230/LIPIcs.FSTTCS.2015.236.

19 Russell Impagliazzo and Avi Wigderson. $P = BPP$ if E Requires Exponential Circuits: Derandomizing the XOR Lemma. In *Proceedings of the Symposium on the Theory of Computing (STOC)*, pages 220–229, 1997. doi:10.1145/258533.258590.

20 Valentine Kabanets and Jin-yi Cai. Circuit minimization problem. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 73–79, 2000. doi:10.1145/335305.335314.

21 Marcos A. Kiwi, Carsten Lund, Daniel A. Spielman, Alexander Russell, and Ravi Sundaram. Alternation in interaction. *Computational Complexity*, 9(3-4):202–246, 2000. doi:10.1007/PL00001607.

22 Cody D. Murray and R. Ryan Williams. On the (Non) NP-Hardness of Computing Circuit Complexity. *Theory of Computing*, 13(1):1–22, 2017. doi:10.4086/toc.2017.v013a004.

23 Alexander Russell and Ravi Sundaram. Symmetric Alternation Captures BPP. *Computational Complexity*, 7(2):152–162, 1998. doi:10.1007/s000370050007.

A Improving the Complexity of the Reductions

In this appendix, we mention that the reducibility notion of Theorem 3 can be significantly improved by using the efficient proof system of [15]:

▶ **Theorem 12.** *For any universal Turing machine U,*

$$\mathsf{EXP}^{\mathsf{NP}} \subseteq \bigcap_R \mathsf{S}_2^R \subseteq \mathsf{S}_2^{\exp},$$

where the intersection is taken over all the dense subsets R of R_{K_U}.

Hirahara [15] introduced the notion of a selector, and showed that $\mathsf{EXP}^{\mathsf{NP}}$ admits a selector.

▶ **Lemma 13** ([15]). *For any $\mathsf{EXP}^{\mathsf{NP}}$-complete language L, there exists a* selector *for L. That is, there exists a randomized polynomial-time oracle machine S such that, for any input $x \in \{0,1\}^*$ and oracles $A_0, A_1 \subseteq \{0,1\}^*$, if $L \in \{A_0, A_1\}$ then $\Pr_S\left[S^{A_0,A_1}(x) = L(x)\right] \geq \frac{2}{3}$.*

We show that any language L with a selector is low for $\mathsf{S}_2^{\mathrm{p}}$ if $L \in \mathsf{P/poly}$.

▶ **Theorem 14.** *Let L be a language with a selector S and R be any oracle. Then,*

$$L \in \mathsf{P}^R/\mathsf{poly} \quad \Longrightarrow \quad \mathsf{S}_2^L \subseteq \mathsf{S}_2^R.$$

This generalizes a lowness result of [12] from any downward self-reducible language L to any language L with a selector.

Proof. The idea is as follows: We request two competing provers of S_2^R to send R-oracle circuits C_0, C_1 that compute L. Then, for every query q of L, one can decide whether $q \stackrel{?}{\in} L$ by running S and using C_0^R and C_1^R as oracles. Details follow.

Take any $A \in \mathsf{S}_2^L$, and let V be an S_2^L-machine that witnesses $A \in \mathsf{S}_2^L$. Take some constants c, d such that V runs in time n^c and S runs in time n^d on inputs of length n.

Now we describe an $\mathsf{S}_2 \cdot \mathsf{BP} \cdot \mathsf{P}^R$ algorithm that computes A: Given an input $x \in \{0,1\}^*$ of length n, for each $i \in \{0,1\}$, the ith competing prover sends an S_2-type certificate y_i for M. Moreover, each prover sends a polynomial-size R-oracle circuit C_i; an honest prover sends a circuit C_i such that C_i^R computes L on every input of length at most n^{cd}. Then simulate V using the two certificates (i.e., run V on input (x, y_0, y_1)), where each query q that V makes is answered with $S^{C_0^R, C_1^R}(q)$. It is easy to see the correctness of this algorithm.

Therefore, we have $A \in \mathsf{S}_2 \cdot \mathsf{BP} \cdot \mathsf{P}^R$, and by [23], we can derandomize the randomized computation by using the power of S_2 and obtain $L \in \mathsf{S}_2 \cdot \mathsf{BP} \cdot \mathsf{P}^R = \mathsf{S}_2^R$. ◀

Proof of Theorem 12. Under any dense subset R of Kolmogorov-random strings, we have $\mathsf{EXP}^{\mathsf{NP}} \subseteq \mathsf{P}^R/\mathsf{poly}$ (by Theorem 7). Thus by taking any $\mathsf{EXP}^{\mathsf{NP}}$-complete problem L, we obtain $\mathsf{EXP}^{\mathsf{NP}} \subseteq \mathsf{S}_2^L \subseteq \mathsf{S}_2^R$ by combining Lemma 13 and Theorem 14. ◀

Finally, we mention that in the case of reductions to the set of random strings, the $\mathsf{S}_2^{\mathrm{p}}$ reductions of Theorem 12 can be derandomized to obtain P^{NP} reductions.

▶ **Theorem 15.** $\mathsf{EXP}^{\mathsf{NP}} \subseteq \mathsf{P}^{\mathsf{NP}^{R_{\mathrm{K}_U}}}$ *for any universal Turing machine U.*

Proof. By Theorem 12, we immediately obtain $\mathsf{EXP}^{\mathsf{NP}} \subseteq \mathsf{S}_2^{R_{\mathrm{K}_U}}$. By the relativized version of Cai's result [11], we have $\mathsf{P}^{\mathsf{NP}^{R_{\mathrm{K}_U}}} \subseteq \mathsf{S}_2^{R_{\mathrm{K}_U}} \subseteq \mathsf{ZPP}^{\mathsf{NP}^{R_{\mathrm{K}_U}}}$; thus it remains to derandomize the computation of ZPP under an $\mathsf{NP}^{R_{\mathrm{K}_U}}$ oracle. One can find the lexicographically first Kolmogorov-random string by a $\mathsf{P}^{\mathsf{NP}^{R_{\mathrm{K}_U}}}$ algorithm. By Lemma 10 and Theorem 7, the circuit complexity relative to an $\mathsf{NP}^{R_{\mathrm{K}_U}}$ oracle of any Kolmogorov-random string of length n is at least $n^{\Omega(1)}$. Thus by using a Kolmogorov-random string as a hard function of the Impagliazzo-Wigderson pseudorandom generator [19], one can derandomize the computation of ZPP under an $\mathsf{NP}^{R_{\mathrm{K}_U}}$ oracle. ◀

Consensus vs Broadcast, with and Without Noise

Andrea Clementi
Università Tor Vergata di Roma, Italy
clementi@mat.uniroma2.it

Luciano Gualà
Università Tor Vergata di Roma, Italy
guala@mat.uniroma2.it

Emanuele Natale
Université Côte d'Azur, Sophia Antipolis, France
emanuele.natale@univ-cotedazur.fr

Francesco Pasquale
Università Tor Vergata di Roma, Italy
pasquale@mat.uniroma2.it

Giacomo Scornavacca
Università degli Studi di Sassari, Italy
giacomoscornavacca@gmail.com

Luca Trevisan
Università Bocconi, Milano, Italy
l.trevisan@unibocconi.it

Abstract

Consensus and Broadcast are two fundamental problems in distributed computing, whose solutions have several applications. Intuitively, Consensus should be no harder than Broadcast, and this can be rigorously established in several models. Can Consensus be *easier* than Broadcast?

In models that allow noiseless communication, we prove a reduction of (a suitable variant of) Broadcast to binary Consensus, that preserves the communication model and all complexity parameters such as randomness, number of rounds, communication per round, etc., while there is a loss in the success probability of the protocol. Using this reduction, we get, among other applications, the first logarithmic lower bound on the number of rounds needed to achieve Consensus in the uniform Gossip model on the complete graph. The lower bound is tight and, in this model, Consensus and Broadcast are equivalent.

We then turn to distributed models with noisy communication channels that have been studied in the context of some bio-inspired systems. In such models, only one noisy bit is exchanged when a communication channel is established between two nodes, and so one cannot easily simulate a noiseless protocol by using error-correcting codes. An $\Omega(\varepsilon^{-2}n)$ lower bound is proved by Boczkowski et al. [PLOS Comp. Bio. 2018] on the convergence time of binary Broadcast in one such model (noisy uniform Pull), where ε is a parameter that measures the amount of noise).

We prove an $O(\varepsilon^{-2}\log n)$ upper bound on the convergence time of binary Consensus in such model, thus establishing an exponential complexity gap between Consensus versus Broadcast. We also prove our upper bound above is tight and this implies, for binary Consensus, a further strong complexity gap between noisy uniform Pull and noisy uniform Push. Finally, we show a $\Theta(\varepsilon^{-2}n\log n)$ bound for Broadcast in the noisy uniform Pull.

2012 ACM Subject Classification Theory of computation → Distributed algorithms; Theory of computation → Random walks and Markov chains; Theory of computation → Random network models

Keywords and phrases Distributed Computing, Consensus, Broadcast, Gossip Models, Noisy Communication Channels

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.42

11th Innovations in Theoretical Computer Science Conference (ITCS 2020).
Editor: Thomas Vidick; Article No. 42; pp. 42:1–42:13

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Category Extended Abstract

Related Version A full version of the paper is available at `https://arxiv.org/abs/1807.05626`.

Funding *Luca Trevisan*: LT was supported by the NSF under grant CCF 1815434 and his work on this project has received funding from the European Research Council (ERC) under the European Union Horizon 2020 research and innovation programme (grant agreement No. 834861).

1 Introduction

In this paper we investigate the relation between Consensus and Broadcast, which are two of the most fundamental algorithmic problems in distributed computing [21, 23, 39, 41], and we study how the presence or absence of communication noise affects their complexity.

In the (Single-Source) *Broadcast* problem, one node in a network has an initial message `msg` and the goal is for all the nodes in the network to receive a copy of `msg`.

In the *Consensus* problem, each of the n nodes of a network starts with an input value (which we will also call an *opinion*), and the goal is for all the nodes to converge to a configuration in which they all have the same opinion (this is the *agreement* requirement) and this shared opinion is one held by at least one node at the beginning (this is the *validity* requirement). In the *Binary* Consensus problem, there are only two possible opinions, which we denote by 0 and 1.

In the (binary) *Majority Consensus* problem [5, 22, 40] we are given the promise that one of the two possible opinions is initially held by at least $n/2 + b(n)$ nodes, where $b(n)$ is a parameter of the problem, and the goal is for the nodes to converge to a configuration in which they all have the opinion that, at the beginning, was held by the majority of nodes. Note that Consensus and Majority Consensus are incomparable problems: a protocol may solve one problem without solving the other.[1] Both the notions of Consensus and Majority Consensus above can be further relaxed to those of δ-*Almost Consensus* and δ-*Almost Majority Consensus*, respectively. According to such weaker notions, we allow the system to converge to an almost-consensus regime where δn *outliers* may have a different opinion from the rest of the nodes.

Motivations for studying the Broadcast problem are self-evident. Consensus and Majority Consensus are simplified models for the way inconsistencies and disagreements are resolved in social networks, biological models and peer-to-peer systems [24, 27, 37].[2]

In distributed model that severely restrict the way in which nodes communicate (to model constraints that arise in peer-to-peer systems or in social or biological networks), upper and lower bounds for the Broadcast problem give insights on the effect of the communication constraints on the way in which information can spread in the network. The analysis of algorithms for Consensus often give insights on how to break symmetry in distributed networks, when looking at how the protocol handles an initial opinion vector in which exactly

[1] A Consensus protocol is allowed to converge to an agreement to an opinion that was initially in the minority (provided that it was held by at least one node), while a Majority Consensus protocol must converge to the initial majority whenever the minority opinion is held by fewer than $n/2 - b$ nodes. On the other hand, a Majority Consensus problem is allowed to converge to a configuration with no agreement if the initial opinion vector does not satisfy the promise, while a Consensus protocol must converge to an agreement regardless of the initial opinion vector.

[2] The Consensus problem is often studied in models in which nodes are subject to malicious faults, and, in that case, one has motivations from network security. In this paper we concentrate on models in which all nodes honestly follow the prescribed protocol and the only possibly faulty devices are the communication channels.

half the nodes have one opinion and half have the other. The analysis of algorithms for Majority Consensus usually hinge on studying the rate at which the number of nodes holding the minority opinion shrinks.

If the nodes are labeled by $\{1, \ldots, n\}$, and each node knows its label, then there is an easy reduction of binary Consensus to Broadcast: node 1 broadcasts its initial opinion to all other nodes, and then all nodes agree on that opinion as the consensus opinion. Even if the nodes do not have known identities, they can first run a *leader election* protocol, and then proceed as above with the leader broadcasting its initial opinion. Even in models where leader election is not trivial, the best known Consensus protocol has, in all the cases that we are aware of, at most the "complexity" (whether it's measured in memory per node, communication per round, number of rounds, etc.) of the best known broadcast protocol.

A first major question that we address in this paper is whether the converse hold, that is, are there ways of obtaining a Broadcast protocol from a Consensus problem or are there gaps, in certain models, between the complexity of the two problems?

We will show that, in the presence of noiseless communication channels, every Consensus protocol can be used to realize a weak form of Broadcast. Since, in many cases, known lower bounds for Broadcast apply also to such weak form, we get new lower bounds for Consensus. In a previously studied, and well motivated, distributed model with noisy communication, namely the noisy Gossip, however, we establish an exponential gap between Consensus and Broadcast.

As a second major question, we investigate the impact of the communication noise on the Consensus problem. More in detail, does this impact strongly depend on the particular noisy Gossip model we adopt? We will give a positive answer to this question by establishing a strong complexity separation between the two most popular versions of the Gossip model, namely, the Pull model and the Push one.

Roadmap of the paper and a remark

In order to formally state and discuss our results, in the next section, we introduce the distributed models and their associated complexity measures our results deal with. In Section 3, we describe our results and their consequences for noiseless communication models and compare them with the related previous work. Section 4 is devoted to our results for the noisy communication models and to their comparison with the related previous work. Finally, in Section 5 we provide a short summary of the obtained results and discuss some related open questions. We remark that, in this version, we only sketch the main ideas of the technical proofs: detailed proofs are given in the full version of the paper [17].

2 Communication and computational models

We study protocols defined on a communication network, described by an undirected graph $G = (V, E)$ where V is the set of nodes, each one running an instance of the distributed algorithm, and E is the set of pairs of nodes between which there is a communication link that allows them to exchange data. When not specified, G is assumed to be the complete graph.

In *synchronous parallel* models, there is a global clock and, at each time step, nodes are allowed to communicate using their links.

In the Local model, there is no restriction on how many neighbors a node can talk to at each step, and no restriction on the number of bits transmitted at each step. There is also no restriction on the amount of memory and computational ability of each node. The only complexity measures is the number of rounds of communication. For example, it is easy to

see that the complexity of Broadcast is the diameter of the graph G. The Congest model is like the Local model but the amount of data that each node can send at each time step is limited, usually to $O(\log n)$ bits.

In the (general) Gossip model [20, 30], at each time step, each node v chooses one of its neighbors c_v and *activates* the communication link (v, c_v), over which communication becomes possible during that time step, allowing v to send a message to c_v and, simultaneously, c_v to send a message to v. We will call v the *caller of* c_v. In the Push variant, each node v sends a message to its chosen neighbor c_v; in the Pull variant, each node sends a message to its callers (if any). Note that, although each node chooses only one neighbor, some nodes may be chosen by several others, and so they may receive several messages in the Push setting, or send a message to several recipients in the Pull setting. In our algorithmic results for the Gossip model, we will assume that each message exchanged in each time step is only one bit, while our negative results for the noiseless setting will apply to the case of messages of unbounded length. In the *uniform* Gossip (respectively Push or Pull) model, the choice of c_v is done uniformly at random among the neighbors of v. This means that uniform models make sense even in anonymous networks, in which nodes are not aware of their identities nor of the identities of their neighbors.[3]

In this work, we are mainly interested in models like Gossip that severely restrict communication [5, 2, 22, 24, 37, 40], both for efficiency consideration and because such models capture aspects of the way consensus is reached in biological population systems, and other domains of interest in network science [4, 6, 23, 12, 24, 25, 27]. Communication capabilities in such scenarios are typically constrained and non-deterministic: both features are well-captured by uniform models.

Asynchronous variants of the Gossip model (such as *Population Protocols* [5, 4]) have also been extensively studied [11, 28, 40]. In this variant, no global clock is available to nodes. Instead, nodes are idle until a single node is activated by a (possibly random) scheduler, either in discrete time or in continuous time. When a node wakes up, it activates one of its incident edges and wakes up the corresponding neighbor. Communication happens only between those two vertices, which subsequently go idle again until the next time they wake up.

Previous studies show that, in both Push and Pull variants of uniform Gossip, (binary) Consensus, Majority Consensus and Broadcast can be solved within logarithmic time (and work per node) in the complete graph, via elementary protocols[4], with high probability (for short *w.h.p.*[5]) [5, 9, 11, 22, 28, 31]. Moreover, efficient protocols have been proposed for Broadcast and Majority Consensus for some restricted families of graphs such as regular expanders and random graphs [1, 15, 14, 19, 29, 36].

However, while for Broadcast $\Omega(\log n)$ time and work are necessary in the complete graph [11, 28, 31], prior to this work, it was still unknown whether a more efficient protocol existed for Consensus and Majority Consensus.

3 Our contribution I: Noiseless communication

Our main result is a reduction of a weak form of Broadcast to Consensus which establishes, among other lower bounds, tight logarithmic lower bounds for Consensus and Majority Consensus both in the uniform Gossip (and hence uniform Pull and Push as well) model and in the general Push model.

[3] In the general Gossip model in which a node can choose which incident edge to activate, a node must, at least, know its degree and have a way to distinguish between its incident edges.
[4] In the case of Majority Consensus, the initial additive bias must have size $\Omega(\sqrt{n \log n})$.
[5] In this paper, we say that an event $\mathcal{E}_n$ holds *w.h.p.* if $\mathbf{P}(\mathcal{E}_n) \geqslant 1 - n^{-\alpha}$, for some $\alpha > 1$.

In order to formally state the reduction, we need to introduce a slightly-different variant of Broadcast where, essentially, it is (only) required that *some* information from the source is spread on the network.

▶ **Definition 1.** *A protocol $\mathcal{P}$ solves the* γ-Infection *problem w.r.t. a source node s if it* infects *at least γn nodes, where we define a node* infected *recursively as follows: initially only s is infected; a node v becomes infected whenever it receives* any *message from an infected node.*

Notice that a protocol $\mathcal{P}$ solving the γ-Infection problem w.r.t. a source node s can be easily turned into a protocol for broadcasting a message `msg` from s to at least γn nodes. Indeed, we give the message `msg` to the source node s, and we simulate $\mathcal{P}$. Every time an infected node sends a message, it appends `msg` to it. Clearly, the size of each message in $\mathcal{P}'$ is increased by the size of `msg`.

This notion is helpful in thinking about upper and lower bounds for Broadcast: any successful broadcast protocol from s needs to infect all nodes from source s, and any protocol that is able to infect all nodes from source s can be used to broadcast from s by appending `msg` to each message originating from an infected node. Thus any lower bound for Infection is also a lower bound for Broadcast, and any protocol for Infection can be converted, perhaps with a small overhead in communication, to a protocol for Broadcast. For example, in the Push model, the number of infected nodes can at most double at each step, because each infected node can send a message to only one other node, and this is the standard argument that proves an $\Omega(\log n)$ lower bound for Broadcast.

In the next theorem, we show that lower bounds for Infection *also give lower bounds for* Consensus. More precisely we prove that if we have a Consensus protocol that, for every initial opinion vector, succeeds in achieving almost consensus with probability $1 - o(1/n)$, then there is an initial opinion vector and a source such that the protocol infects many nodes from that source with probability at least $(1 - o(1))/n$.

▶ **Theorem 2.** *Let $\mathcal{P}$ be a protocol reaching δ-Almost Consensus with probability at least $1 - o(1/n)$. Then, a source node s and an initial opinion vector $\mathbf{x}$ exist such that $\mathcal{P}$, starting from $\mathbf{x}$, solves the $(1 - 2\delta)$-Infection problem w.r.t. s with probability at least $(1 - o(1))/n$.*

Notice that the above result implies that any protocol for Consensus actually solves the Infection problem (when initialized with a certain opinion vector) in a weak sense: the infection is w.r.t. a source that depends on the consensus protocol in a (possibly) uncontrolled manner; and (ii) the success probability of the infection is quite low. However, if we are in a model in which there is no source for which we can have probability, say, $\geqslant 1/(2n)$ of infecting all nodes with certain resources (such as time, memory, communication per node, etc.), then, in the same model, and with the same resources, the above theorem implies that every Consensus protocol has probability $\Omega(1/n)$ of failing. For example, by the above argument, we have an $\Omega(\log n)$ lower bound for Consensus in the Push model (because, in fewer than $\log_2 n$ rounds, the probability of infecting all nodes is zero).

In case of Consensus problem (i.e. $\delta = 0$), our proof for Theorem 2 makes use of a hybrid argument to show that there are two initial opinion vectors $\mathbf{x}$ and $\mathbf{y}$, which are identical except for the initial opinion of a node s, such that there is at least a $(1 - o(1))/n$ difference between the probability of converging to the all-zero configuration starting from $\mathbf{x}$ or from $\mathbf{y}$. Then, we prove that this difference must come entirely from runs of the protocol that fail to achieve consensus (which happens only with $o(1/n)$ probability) or from runs of the protocol in which s infects all other nodes. Thus the probability that s infects all nodes from the initial vector $\mathbf{x}$ has to be $\geqslant (1 - o(1))/n$. Then, to extend the above approach for the Almost Consensus problem (i.e. $\delta > 0$), some additional care and a suitable counting argument are required to manage the unknown set of outliers.

As for Majority Consensus, we have a similar reduction, but from a variant of the infection problem in which there is an initial set of b infected nodes.[6] Formally:

▶ **Theorem 3.** *Let $\mathcal{T}$ be any fixed resource defined on a distributed system $\mathcal{S}$ and suppose there is no Infection protocol that, starting from any subset of n^α nodes with $\alpha < 1$, can inform at least $(1-\delta)n$ nodes by using at most τ^{B} units of $\mathcal{T}$, w.h.p. Then, any protocol $\mathcal{P}$ on this model, reaching δ-*Almost Majority Consensus *w.h.p., must use more than τ^{B} units of $\mathcal{T}$.*

3.1 Some applications

Lower bounds for infection are known in several models in which there are no previous negative results for Consensus. We have not attempted to survey all possible applications of our reductions, but here we enumerate some of them (see the full version for the formal statements of such results):

- In the uniform Gossip model (also known as uniform Push-Pull model), and in the general Push model, tight analysis (see [30, 31]) show that any protocol $\mathcal{P}$ for the complete graph w.h.p. does not complete Broadcast within less than $\beta \log n$ rounds, where β is a sufficiently small constant. Combining this lower bound with our reduction result above, we get an $\Omega(\log n)$ lower bound for Consensus. This is the first known lower bound for Consensus showing a full equivalence between the complexity of Broadcast and Consensus in such models. Regarding Majority Consensus, we also obtain an $\Omega(\log n)$ lower bound for any initial bias $b = O(n^\alpha)$, with $\alpha < 1$.
- In a similar way, we are able to prove a lower bound of $\Omega(n \log n)$ number of steps (and hence $\Omega(\log n)$ parallel time) or $\Omega(\log n)$ number of messages per node for Consensus on an asynchronous variant of the Gossip model, the *Population Protocols* with uniform/probabilistic scheduler, as defined in [5].
- The last application we mention here concerns the synchronous *Radio Network* model [3, 7, 16, 42]. Several optimal bounds have been obtained on the Broadcast time [7, 18, 32, 33, 34] while only few results are known for Consensus time [16, 42]. In particular, we are not aware of better lower bounds other than the trivial $\Omega(D)$ (where D denotes the diameter of the network). Then, by combining a previous lower bound in [3] on Broadcast with our reduction result, we get a new lower bound for Consensus in this model.

We remark that our reduction allows us to prove that some of the above lower bounds hold even if the nodes have unbounded memory and can send/receive messages of unbounded size.

4 Our contribution II: Noisy communication

We now turn to the study of distributed systems in which the communication links between nodes are noisy. We will consider a basic model of high-noise communication: the binary symmetric channel [35] in which each exchanged bit is flipped independently at random with probability $1/2 - \varepsilon$, where $0 \leqslant \varepsilon < 1/2$, and we refer to ε as the *noise* parameter of the model. Then, in the sequel, the version of each model $\mathcal{M}$, in which the presence of communication noise above is introduced, will be shortly denoted as *noisy* $\mathcal{M}$.

[6] Recall that b is the value such that we are promised that the majority opinion is held, initially, by at least $n/2 + b$ nodes.

In models such as Local and Congest, the ability to send messages of logarithmic length (or longer) implies that, with a small overhead, one can encode the messages using error-correcting codes and simulate protocols that assume errorless communication.

In the uniform Gossip model with one-bit messages, however, error-correcting codes cannot be used and, indeed, whenever the number of rounds is sublinear in n, most of the pairs of nodes that ever communicate only exchange a single bit.

The study of fundamental distributed tasks, such as Broadcast and Majority Consensus, has been undertaken in the uniform Gossip model with one-bit messages and noisy links [10, 25] as a way of modeling the restricted and faulty communication that takes place in biological systems, and as a way to understand how information can travel in such systems, and how they can repair inconsistencies. Such investigation falls under the agenda of *natural algorithms*, that is, the investigation of biological phenomena from an algorithmic perspective [13, 38].

As for the uniform Push model with one-bit messages, we first notice that there is a simple local strategy that solves both (binary) Broadcast and Consensus in the noisy Push (this strategy holds even assuming that agents share only a *binary* synchronous clock). For instance, consider binary Consensus: let every node with initial opinion 0 start a broadcast process at even rounds, while the same task is started in odd rounds by nodes with initial opinion 1. When a node receives a bit in any even (odd) round, this bit is always interpreted as 0 (1). Then, at every round, each node updates its output with, for instance, the minimum value it has seen so far (any round).

In [25], the authors consider a restricted, natural class of *symmetric* algorithms where the action of the nodes cannot depend on the value of the exchanged bits. In this setting, they prove that (binary) Broadcast and (binary) Majority Consensus can be solved in time $O(\varepsilon^{-2} \log n)$, where ε is the noise parameter. They also prove a matching lower bound for this class of algorithms. This has been later generalized to non-binary opinions in [26].

In the noisy uniform Pull model however, [10] proves an $\Omega(\varepsilon^{-2}n)$ time lower bound[7]. This lower bound is proved even under assumptions that strengthen the negative result, such as unique node IDs, full synchronization, and shared randomness (see Section 2.4 of [10] for more details on this point).

Such a gap between noisy uniform Push and Pull comes from the fact that, in the Push model, a node is allowed to decline to send a message, and so one can arrange a protocol in which nodes do not start communicating until they have some confidence of the value of the broadcast value. In the Pull model, instead, a called node must send a message, and so the communication becomes polluted with noise from the messages of the non-informed nodes.

What about Consensus and Majority Consensus in the noisy Pull model? Our reduction in Theorem 2 suggests that there could be $\Omega(\varepsilon^{-2}n)$ lower bounds for Consensus and Majority Consensus, but recall that the reduction is to the infection problem, and infection is equivalent to Broadcast only when we have errorless channels.

4.1 Upper bounds in noisy uniform Pull

4.1.1 A protocol for Consensus and its analysis

We devise a simple and natural protocol for Consensus for the noisy uniform Pull model having convergence time $O(\varepsilon^{-2} \log n)$, w.h.p., thus exhibiting an exponential gap between Consensus and Broadcast in the noisy uniform Pull model.

[7] They actually proved a more general result including non-binary noisy channels.

▶ **Theorem 4.** *In the noisy uniform* PULL *model, with noisy parameter ε, a protocol exists that achieves Consensus within $O(\varepsilon^{-2}\log n)$ rounds and communication, w.h.p. The protocol requires $\Theta(\log\log n + \log\varepsilon^{-2})$ local memory.*
Moreover, if the protocol starts from any initial opinion vector with bias $b = \Omega(\sqrt{n\log n})$, then it guarantees Majority Consensus, w.h.p.

The protocol we refer to in the above theorem works in two consecutive phases. Each phase is a simple application of the well-known *k-Majority Dynamics* [8, 9]:

> k-MAJORITY. *At every round, each node samples k neighbours*[8] *independently and u.a.r. (with replacement). Then, the node updates its opinion according to the majority opinion in the sample.*

The protocol is thus the following:

> MAJORITY PROTOCOL. Let α be a sufficiently large positive constant[9]. Every node performs $\alpha\log n$ rounds of k-Majority with $k = \Theta(1/\varepsilon^2)$, followed by one round of the k-Majority with $k = \Theta(\varepsilon^{-2}\log n)$.

Our analysis shows that, w.h.p., at the end of the first phase there is an opinion that is held by at least $n/2+\Omega(n)$ nodes, and that if the initial opinions where unanimous then the initial opinion is the majority opinion after the first phase (notice that the latter fact guarantees the validity property, w.h.p.). Then, in the second phase, despite the communication errors, we show every node has a high probability of seeing the true phase-one majority as the empirical majority in the batch and so all nodes converge to the same valid opinion. To analyze the first phase, we break it out into two sub-phases (this breakdown is only in the analysis, not in the protocol): in a first sub-phase of length $O(\varepsilon^{-2}\log n)$, we prove the protocol "breaks symmetry" w.h.p. and, no matter the initial vector and the presence of communication noise, reaches a configuration in which one opinion is held by $n/2+\Omega(\sqrt{n\log n})$ nodes. In the second sub-phase, also of length $O(\varepsilon^{-2}\log n)$, a configuration of bias $\Omega(\sqrt{n\log n})$ w.h.p. becomes a configuration of bias $\Omega(n)$. The analysis of the first sub-phase is our main technical novelty while the analysis of the second sub-phase for achieving Majority Consensus is similar to that in [25, 26]. If the initial opinion vector is unanimous, then it is not necessary to break up the first phase into sub-phases, and one can directly see that a unanimous configuration maintains a bias $\Omega(n)$, w.h.p., for the duration of the first phase.

A consequence of our analysis is that, if the initial opinion vector has a bias $\Omega(\sqrt{n\log n})$, then the protocol converges to the majority, w.h.p. So, we get a Majority-Consensus protocol for this model under the above condition on the bias.

4.1.2 A protocol for Broadcast

We provide a simple two-phases Broadcast protocol that runs in the noisy uniform PULL model.

Protocol NOISYBROADCAST.

- In the first phase, each non-source node displays 0 (obviously, the source displays its input value), and performs a pull operation for $\Theta(\varepsilon^{-2}n\log n)$ rounds; it then chooses to support value 1 iff the fraction of received messages equal to 1 is at least $\frac{1}{2} - \varepsilon(1-\frac{1}{2n})$, zero otherwise.

[8] In the binary case when k is odd, the k-Majority is stochastically equivalent to the $k+1$-Majority where ties are broken u.a.r. (see Lemma 17 in [26]). For this reason, in this section we assume that k is odd.
[9] The value of α will be fixed later in the analysis.

- In the second phase, nodes run the Majority Consensus protocol of Theorem 4, starting with the value obtained at the end of the first phase.

We prove the following performance of the protocol, nearly matching the $\Omega(\varepsilon^{-2}n)$ lower bound mentioned before [10]:

▶ **Theorem 5.** *Protocol* NoisyBroadcast *solves the Broadcast problem in the noisy uniform* Pull *model in* $\mathcal{O}(\varepsilon^{-2}n\log n)$ *rounds, w.h.p.*

Our proof shows that at the end of the first phase, the fraction of nodes which have obtained a value equal to the source's input is greater than those that failed by at least $\sqrt{n\log n}$ nodes. The latter fact satisfies the hypothesis of Theorem 4 for solving Majority Consensus in $\mathcal{O}(\varepsilon^{-2}\log n)$, which constitutes the second phase.

4.2 Lower bounds in noisy Pull models

We prove that any Almost Consensus protocol with at most δn outliers and with error probability at most δ requires $\Omega\left(\varepsilon^{-2}\log\delta^{-1}\right)$ rounds. Formally:

▶ **Theorem 6.** *Let* δ *be any real such that* $0 < \delta < 1/8$ *and consider any protocol* $\mathcal{P}$ *for the noisy general* Pull *model with noise parameter* ε*. If* $\mathcal{P}$ *solves* δ*-Almost Consensus with probability at least* $1-\delta$*, then it requires at least* $t = \Omega(\varepsilon^{-2}\log\delta^{-1})$ *rounds*[10].

This shows that the complexity $O(\varepsilon^{-2}\log n)$ of our protocol described in Subsection 4.1.1 is tight for protocols that succeed w.h.p. We remark that our result holds for any version (general and uniform) of the noisy Pull model with noise parameter ε, unbounded local memory, even assuming unique node IDs. Recalling the $\Theta(\log n)$ bound that holds for (general) Consensus protocols in the noisy uniform Push (for any value of ε), our lower bound above thus implies a strong separation result between noisy uniform Pull and noisy uniform Push.

The proof of Theorem 6 is one of the main technical contributions of this work and below we provide a short discussion.

In [25], an $\Omega\left(\varepsilon^{-2}\log\delta^{-1}\right)$ round lower bound is proved for Majority Consensus in the uniform Push model, for a restricted class of protocols. Their argument, roughly speaking, is that each node needs to receive a bit of information from the rest of the graph (namely, the majority value in the rest of the graph), and this bit needs to be correctly received with probability $1-\delta$, while using a binary symmetric channel with error parameter ε. It is then a standard fact from information theory that the channel needs to be used $\Omega(\varepsilon^{-2}\log\delta^{-1})$ times. It is not clear how to adapt this argument to the Consensus problem. Indeed, it is not true that every node receives a bit of information with high confidence from the rest of the graph (consider the protocol in which one node broadcasts its opinion), and it is not clear if there is a distribution of initial opinions such that there is a node v whose final opinion has mutual information close to 1 to the global initial opinion vector given the initial opinion of v (the natural generalization of the argument of [25]). Instead, we prove that there are two initial opinion vectors $\mathbf{x}$ and $\mathbf{y}$, a node v, and a bit b, such that the initial opinion of v is the same in $\mathbf{x}$ and $\mathbf{y}$, but the probability that v outputs b is $\leqslant\delta$ when the initial opinion vector is $\mathbf{x}$ and $\geqslant\Omega(1)$ when the initial opinion vector is $\mathbf{y}$. Thus, the rest of the graph is sending v a bit of information (whether the initial opinion vector is $\mathbf{x}$ or $\mathbf{y}$) and the communication

[10] We notice the double role parameter δ has in this statement.

succeeds with probability $\geqslant 1-\delta$ when the bit has one value and with probability $\geqslant 1/3$ if the bit has the other value. Despite this asymmetry, if the communication takes place over a binary symmetric channel with error parameter ε, we use KL divergence to show that the channel has to be used $\Omega(\varepsilon^{-2}\log\delta^{-1})$ times.

4.2.1 An improved lower bound for Broadcast

The $\Omega(\varepsilon^{-2}n)$ lower bound of [10] for Broadcast in the uniform PULL model applies to protocols that have constant probability of correctly performing the broadcast operation. With the following theorem we show a way of modifying their proof (in particular, to derive an $\Omega(\varepsilon^{-2}n\log n)$ for uniform PULL protocols for Broadcast that have high probability of success, matching the $O(\varepsilon^{-2}n\log n)$ round complexity of Theorem 5.

▶ **Theorem 7.** *The Broadcast Problem cannot be solved in the noisy uniform* Pull *model w.h.p. in less than* $\Omega(\varepsilon^{-2}n\log n)$ *rounds.*

5 Conclusions

Table 1 shows the two main separation results that follow from a comparison between some previous bounds and the bounds we obtain in this paper: The complexity gap between Consensus and Broadcast in the presence or absence of noise and the different complexity behaviour of Consensus between noisy uniform Pull and noisy uniform Push. The figure also shows our new lower bounds for Consensus in the noiseless Gossip models.

A further consequence regards a separation between general Pull and Push models as far as Consensus is concerned in the noiseless world. Indeed, if we assume unique IDs, in the general Pull model, Consensus can be easily solved in constant time: every node can copy the opinion of a prescribed node by means of a single pull operation. On the other hand, in the general Push model, our Broadcast-Consensus reduction shows that $\Omega(\log n)$ rounds are actually necessary for solving Consensus.

We considered noisy communication models that assume the presence of a global clock: nodes work in parallel sharing the value of the current round. Our protocols definitely exploit this important property of the model. Then, an interesting open issue is to analyse fundamental tasks, such as Consensus and Broadcast, in asynchronous versions of the Push and Pull models where, as in our setting, communication is noisy and takes place via binary messages only. A further interesting future work we plan to consider is to introduce a (strong) bound on the local memory of the nodes.

Table 1 The updated state-of-art for Broadcast and Consensus in Gossip models with and w/o noise. The results of this paper are highlighted bold, while "folklore" results are marked with * (see also the preliminary discussion in Section 4).

	ε-noisy models		Noiseless models	
	Uniform Pull	**Uniform Push**	**Uniform Pull**	**Uniform Push**
Broadcast	$\Omega\left(n/\varepsilon^2\right)$ [10] $\mathbf{\Theta\left(\frac{n\log n}{\varepsilon^2}\right)}$	$\Theta\left(\log n\right)^*$	$\Theta(\log n)$ [30, 31]	$\Theta(\log n)$ [30, 31]
Consensus	$\mathbf{\Theta\left(\frac{\log n}{\varepsilon^2}\right)}$	$\mathbf{\Omega\left(\log n\right)}$ $\mathcal{O}(\log n)^*$	$\mathbf{\Omega\left(\log n\right)}$ $\mathcal{O}(\log n)$ [30, 31]	$\mathbf{\Omega\left(\log n\right)}$ $\mathcal{O}(\log n)$ [30, 31]

References

1. Mohammed Amin Abdullah and Moez Draief. Majority Consensus on Random Graphs of a Given Degree Sequence. *CoRR*, abs/1209.5025, 2012. arXiv:1209.5025.
2. Mohammed Amin Abdullah and Moez Draief. Global majority consensus by local majority polling on graphs of a given degree sequence. *Discrete Applied Mathematics*, 180:1–10, 2015.
3. Noga Alon, Amotz Bar-Noy, Nathan Linial, and David Peleg. A lower bound for radio broadcast. *Journal of Computer and System Sciences*, 43(2):290–298, 1991. doi:10.1016/0022-0000(91)90015-W.
4. Dana Angluin, James Aspnes, Zoë Diamadi, Michael J. Fischer, and Peralta René. Computation in networks of passively mobile finite-state sensors. *Distributed Computing*, 18(4):235–253, 2006.
5. Dana Angluin, James Aspnes, and David Eisenstat. A Simple Population Protocol for Fast Robust Approximate Majority. *Distributed Computing*, 21(2):87–102, 2008. (Preliminary version in DISC'07).
6. Dana Angluin, Michael J. Fischer, and Hong Jiang. Stabilizing consensus in mobile networks. In *Proc. of Distributed Computing in Sensor Systems (DCOSS'06)*, volume 4026 of *LNCS*, pages 37–50, 2006.
7. Reuven Bar-Yehuda, Oded Goldreich, and Alon Itai. On the time-complexity of broadcast in multi-hop radio networks: An exponential gap between determinism and randomization. *Journal of Computer and System Sciences*, 45(1):104–126, 1992. doi:10.1016/0022-0000(92)90042-H.
8. Luca Becchetti, Andrea Clementi, Emanuele Natale, Francesco Pasquale, and Luca Trevisan. Stabilizing consensus with many opinions. In *Proc. of the 27th Ann. ACM-SIAM Symp. on Discrete algorithms*, pages 620–635. SIAM, 2016.
9. Luca Becchetti, Andrea E.F. Clementi, Emanuele Natale, Francesco Pasquale, Riccardo Silvestri, and Luca Trevisan. Simple dynamics for plurality consensus. In *ACM SPAA'14*, pages 247–256, 2014.
10. Lucas Boczkowski, Emanuele Natale, Ofer Feinerman, and Amos Korman. Limits on reliable information flows through stochastic populations. *PLOS Computational Biology*, 14(6):e1006195, June 2018. doi:10.1371/journal.pcbi.1006195.
11. Stephen Boyd, Arpita Ghosh, Balaji Prabhakar, and Devavrat Shah. Randomized Gossip Algorithms. *IEEE/ACM Transactions on Networking*, 14:2508–2530, 2006. doi:10.1109/TIT.2006.874516.
12. L. Cardelli and A. Csikász-Nagy. The cell cycle switch computes approximate majority. *Scientific Reports*, Vol. 2, 2012.
13. Bernard Chazelle. Natural Algorithms and Influence Systems. *Commun. ACM*, 55(12):101–110, December 2012. doi:10.1145/2380656.2380679.
14. Flavio Chierichetti, Silvio Lattanzi, and Alessandro Panconesi. Almost Tight Bounds for Rumour Spreading with Conductance. In *Proceedings of the Forty-second ACM Symposium on Theory of Computing*, STOC '10, pages 399–408, New York, NY, USA, 2010. ACM. doi:10.1145/1806689.1806745.
15. Flavio Chierichetti, Silvio Lattanzi, and Alessandro Panconesi. Rumor spreading in social networks. *Theoretical Computer Science*, 412(24):2602–2610, 2011. Selected Papers from 36th International Colloquium on Automata, Languages and Programming (ICALP 2009). doi:10.1016/j.tcs.2010.11.001.
16. Gregory Chockler, Murat Demirbas, Seth Gilbert, Nancy Lynch, Calvin Newport, and Tina Nolte. Consensus and collision detectors in radio networks. *Distributed Computing*, 21(1):55–84, June 2008. doi:10.1007/s00446-008-0056-2.
17. Andrea E. F. Clementi, Luciano Gualà, Emanuele Natale, Francesco Pasquale, Giacomo Scornavacca, and Luca Trevisan. Consensus Needs Broadcast in Noiseless Models but can be Exponentially Easier in the Presence of Noise. *CoRR*, abs/1807.05626, 2018. arXiv:1807.05626.

18 Andrea E. F. Clementi, Angelo Monti, and Riccardo Silvestri. Selective Families, Superimposed Codes, and Broadcasting on Unknown Radio Networks. In *Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '01, pages 709–718, Philadelphia, PA, USA, 2001. Society for Industrial and Applied Mathematics. URL: `http://dl.acm.org/citation.cfm?id=365411.365756`.

19 C. Cooper, R. Elsasser, and T. Radzik. The Power of Two Choices in Distributed Voting. In *Proceedings of the 41st International Colloquium on Automata, Languages, and Programming (ICALP'14)*, volume 8573 of *LNCS*, pages 435–446. Springer, 2014.

20 A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry. Epidemic algorithms for replicated database maintenance. In *ACM PODC'87*, 1987.

21 E. W. Dijkstra. Self-stabilizing Systems in Spite of Distributed Control. *Commun. ACM*, 17(11):643–644, 1974. `doi:10.1145/361179.361202`.

22 Benjamin Doerr, Leslie Ann Goldberg, Lorenz Minder, Thomas Sauerwald, and Christian Scheideler. Stabilizing consensus with the power of two choices. In *ACM SPAA'11*, pages 149–158, 2011.

23 S. Dolev. *Self-Stabilization*. The MIT Press, 2000.

24 David Doty. Timing in chemical reaction networks. In *ACM-SIAM SODA'14*, pages 772–784, 2014.

25 Ofer Feinerman, Bernhard Haeupler, and Amos Korman. Breathe Before Speaking: Efficient Information Dissemination Despite Noisy, Limited and Anonymous Communication. *Distributed Computing*, 30(5):239–355, 2017. Ext. Abs. in ACM PODC'14.

26 Pierre Fraigniaud and Emanuele Natale. Noisy rumor spreading and plurality consensus. *Distributed Computing*, pages 1–20, June 2018. `doi:10.1007/s00446-018-0335-5`.

27 Nigel R. Franks, Stephen C. Pratt, Eamonn B. Mallon, Nicholas F. Britton, and David J.T. Sumpter. Information flow, opinion polling and collective intelligence in house–hunting social insects. *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, 357(1427):1567–1583, 2002.

28 George Giakkoupis, Yasamin Nazari, and Philipp Woelfel. How asynchrony affects rumor spreading time. In *35th ACM Symposium on Principles of Distributed Computing (PODC 2016)*, 2016.

29 George Giakkoupis and Thomas Sauerwald. Rumor Spreading and Vertex Expansion. In *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms*, SODA '12, pages 1623–1641, Philadelphia, PA, USA, 2012. Society for Industrial and Applied Mathematics. URL: `http://dl.acm.org/citation.cfm?id=2095116.2095245`.

30 Bernhard Haeupler. Simple, Fast and Deterministic Gossip and Rumor Spreading. *J. ACM*, 62(6):47:1–47:18, December 2015. `doi:10.1145/2767126`.

31 R. Karp, C. Schindelhauer, S. Shenker, and B. Vocking. Randomized rumor spreading. In *IEEE FOCS'00*, pages 565–574, 2000.

32 D. R. Kowalski and A. Pelc. Deterministic broadcasting time in radio networks of unknown topology. In *The 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002. Proceedings.*, pages 63–72, 2002. `doi:10.1109/SFCS.2002.1181883`.

33 Fabian Kuhn, Nancy Lynch, Calvin Newport, Rotem Oshman, and Andrea Richa. Broadcasting in Unreliable Radio Networks. In *Proceedings of the 29th ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*, PODC '10, pages 336–345, New York, NY, USA, 2010. ACM. `doi:10.1145/1835698.1835779`.

34 E. Kushilevitz and Y. Mansour. An $\Omega(D \log(N/D))$ Lower Bound for Broadcast in Radio Networks. *SIAM Journal on Computing*, 27(3):702–712, 1998. `doi:10.1137/S0097539794279109`.

35 David J. C. MacKay. *Information theory, inference, and learning algorithms*. Cambridge University Press, 2003.

36 G. B. Mertzios, S. E. Nikoletseas, C. Raptopoulos, and P. G. Spirakis. Determining Majority in Networks with Local Interactions and very Small Local Memory. In *Proceedings of the 41st International Colloquium on Automata, Languages, and Programming (ICALP'14)*, 2014.

37 Elchanan Mossel, Joe Neeman, and Omer Tamuz. Majority dynamics and aggregation of information in social networks. *Autonomous Agents and Multi-Agent Systems*, 28(3):408–429, 2014.

38 Saket Navlakha and Ziv Bar-Joseph. Distributed information processing in biological and computational systems. *Communications of the ACM*, 58(1):94–102, 2015.

39 Marshall Pease, Robert Shostak, and Leslie Lamport. Reaching agreement in the presence of faults. *Journal of the ACM*, 27(2):228–234, 1980.

40 Etienne Perron, Dinkar Vasudevan, and Milan Vojnovic. Using Three States for Binary Consensus on Complete Graphs. In *IEEE INFOCOM'09*, pages 2527–1535, 2009.

41 Michael O. Rabin. Randomized byzantine generals. In *Proc. of the 24th Ann. Symp. on Foundations of Computer Science (SFCS)*, pages 403–409. IEEE, 1983.

42 N. Santoro and P. Widmayer. Time is Not a Healer. In *Proceedings of the 6th Annual Symposium on Theoretical Aspects of Computer Science on STACS 89*, pages 304–313, New York, NY, USA, 1989. Springer-Verlag New York, Inc. URL: `http://dl.acm.org/citation.cfm?id=73228.73254`.

Testing Linear Inequalities of Subgraph Statistics

Lior Gishboliner
School of Mathematical Sciences, Tel Aviv University, Tel Aviv, 69978, Israel
liorgis1@mail.tau.ac.il

Asaf Shapira
School of Mathematical Sciences, Tel Aviv University, Tel Aviv 69978, Israel
asafico@tau.ac.il

Henrique Stagni
Departamento de Ciencia da Computacao, Instituto de Matematica e Estatistica,
Universidade de Sao Paulo, Brazil
stagni@gmail.com

Abstract

Property testers are fast randomized algorithms whose task is to distinguish between inputs satisfying some predetermined property $\mathcal{P}$ and those that are far from satisfying it. Since these algorithms operate by inspecting a small randomly selected portion of the input, the most natural property one would like to be able to test is whether the input does not contain certain forbidden small substructures. In the setting of graphs, such a result was obtained by Alon et al., who proved that for any finite family of graphs $\mathcal{F}$, the property of being induced $\mathcal{F}$-free (i.e. not containing an induced copy of any $F \in \mathcal{F}$) is testable.

It is natural to ask if one can go one step further and prove that more elaborate properties involving induced subgraphs are also testable. One such generalization of the result of Alon et al. was formulated by Goldreich and Shinkar who conjectured that for any finite family of graphs $\mathcal{F}$, and any linear inequality involving the densities of the graphs $F \in \mathcal{F}$ in the input graph, the property of satisfying this inequality can be tested in a certain restricted model of graph property testing. Our main result in this paper disproves this conjecture in the following strong form: some properties of this type are not testable even in the classical (i.e. unrestricted) model of graph property testing.

The proof deviates significantly from prior non-testability results in this area. The main idea is to use a linear inequality relating induced subgraph densities in order to encode the property of being a pseudo-random graph.

2012 ACM Subject Classification Mathematics of computing → Discrete mathematics

Keywords and phrases graph property testing, subgraph statistics

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.43

Funding *Asaf Shapira*: Supported in part by ISF Grant 1028/16 and ERC Starting Grant 633509.

1 Introduction

Property testers are fast randomized algorithms that distinguish between objects satisfying a certain property and objects that are "far" from the property. The systematic study of such problems originates in the seminal papers of Rubinfeld and Sudan [10] and Goldreich, Goldwasser and Ron [4], and has since become a very active area of research. We refer the reader to the book of Goldreich [3] for more background and references on the subject.

In this paper we study property testing of graph properties in the *dense graph model*. In this model, a graph is given as an $n \times n$ adjacency matrix. An n-vertex graph G is said to be ε-*far* from a graph property Π, if one has to change at least εn^2 entries in the adjacency matrix of G in order to turn it into a graph satisfying Π. A *tester* for Π is a (randomized) algorithm that, given a proximity parameter $\varepsilon \in (0,1)$ and a graph G, accepts if G satisfies Π and rejects if G is ε-far from Π, with success probability at least $\frac{2}{3}$ in both cases. The

tester is given oracle access to the adjacency matrix of the input, to which it makes queries. We say that a tester has query complexity $q(\varepsilon, n)$ if it makes at most $q(\varepsilon, n)$ queries when invoked with proximity parameter ε on inputs with n vertices. A property Π is called *testable* if it has a tester whose query complexity is bounded by a function of ε alone, that is, it is independent of the size of the input. A tester is *canonical* if it works by sampling a random set of vertices of some size $s(\varepsilon, n)$, querying all pairs among these vertices, and making its decision based on (the isomorphism class) of the sample. The integer $s(\varepsilon, n)$ is called the *sample complexity* of the tester. Goldreich and Trevisan [7, Theorem 2] showed that every tester can be transformed to a canonical one, with the cost of possibly squaring the query complexity. A tester is *size-oblivious* if it does not know n; that is, if its function depends only on the proximity parameter ε (and not on the size of the input). The transformation of Goldreich and Trevisan [7] which turns arbitrary testers into canonical testers, preserves the property of being size-oblivious.

In this paper we study a special kind of testers, called *proximity oblivious testers*, which are defined as follows.

▶ **Definition 1.** *A* proximity oblivious tester *(POT) for a graph property* Π *is an algorithm which makes a* constant *(i.e. independent of* n and ε*) number of queries to the input and satisfies the following. There is a constant* $c \in (0, 1]$ *and a function* $f : (0, 1] \to (0, 1]$ *such that:*

1. *If the input graph satisfies* Π *then the tester accepts with probability at least* c*.*
2. *If the input graph is* ε*-far from* Π *then the tester accepts with probability at most* $c - f(\varepsilon)$*.*

Observe that a POT for Π can be used to obtain a standard tester for Π, by invoking the POT $T = \Theta(1/f(\varepsilon)^2)$ times and accepting if and only if the POT accepted in at least $(c - \frac{f(\varepsilon)}{2})T$ of the tests.

POTs were introduced by Goldreich and Ron [5], who studied *one-sided-error POTs*, namely POTs that accept every input which satisfies the property with probability 1 (this corresponds to having $c = 1$ in Definition 1). Later, Goldreich and Shinkar [6] studied general (two-sided-error) POTs in several settings, including those of general boolean functions, dense graphs and bounded degree graphs. For the dense graph model, they designed a POT for the property of being αn-regular (for a given $\alpha \in (0, 1)$), as well as for several related properties. Moreover, they considered properties of the following form: given graphs H, G, the *density* of H in G, denoted by $p(H, G)$, is the fraction of induced subgraphs of G of order $v(H)$ which are isomorphic to H. Given an integer $h \geq 2$, a rational number b and rational numbers $w_H \geq 0$, where H runs over all h-vertex graphs, the property $\Pi_{h,w,b}$ is defined as the property of all graphs G satisfying

$$\sum_H w_H \cdot p(H, G) \leq b.$$

Throughout this paper, a tuple (h, w, b) will always consist of an integer $h \geq 2$, a rational number b, and a function $w \colon \{H : v(H) = h\} \to \mathbb{Q}^+$ from the set of all h-vertex graphs to the positive rationals. The value assigned by w to a graph H is denoted by w_H.

Since property testing algorithms, and POTs in particular, work by inspecting the subgraph induced by a small sample of vertices, it is natural to ask if the property of not containing an induced copy of a fixed graph H is a testable property. Such a result was obtained by Alon, Fischer, Krivelevich and Szegedy [1] who proved that in fact for every finite family of graphs $\mathcal{F}$, the property of being induced $\mathcal{F}$-free (i.e. not containing an induced copy of F for every $F \in \mathcal{F}$) is testable. It is easy to see that the family of properties $\Pi_{h,w,b}$ forms a strict generalization of the family of properties of being induced $\mathcal{F}$-free, since the

former can encode the latter. (Indeed, if all graphs in $\mathcal{F}$ have the same size h then simply set $b = 0$, $w_H = 1$ for each $H \in \mathcal{F}$, and $w_H = 0$ for each h-vertex graph H which is not in $\mathcal{F}$. If graphs in $\mathcal{F}$ have varying sizes, then take advantage of the fact that for every pair of graphs F, G and $h \geq v(F)$, it holds that $p(F,G) = \sum_H p(F,H) \cdot p(H,G)$, where the sum is over all h-vertex graphs H.)

We now arrive at an important definition.

▶ **Definition 2.** *A tuple (h,w,b) has the* removal property *if there is a function $f : (0,1] \to (0,1]$ such that for every $\varepsilon \in (0,1)$ and for every graph G, if G is ε-far from $\Pi_{h,w,b}$ then*

$$\sum_H w_H \cdot p(H,G) \geq b + f(\varepsilon).$$

As an example, the main result of [1] mentioned above is equivalent to the statement that if $b = 0$ then $\Pi_{h,w,b}$ has the removal property. Goldreich and Shinkar [6] observed that if (h,w,b) has the removal property then $\Pi_{h,w,b}$ admits a size-oblivious POT. Indeed, given an input graph G, the POT works by sampling a random induced subgraph of G of order h, and then rejecting with probability w_H if the sampled subgraph is isomorphic to H, for each H on h vertices. If G satisfies $\Pi_{h,w,b}$ then by the definition of this property, G is rejected with probability $\sum_H w_H \cdot p(H,G) \leq b$. On the other hand, if G is ε-far from $\Pi_{h,w,b}$ then by the removal property, G is rejected with probability $\sum_H w_H \cdot p(H,G) \geq b + f(\varepsilon)$. Thus, Definition 1 is satisfied with $c = 1 - b$.

Our first result in this paper is a converse of the above statement, showing that if $\Pi_{h,w,b}$ has a size-oblivious POT, then (h,w,b) has the removal property.

▶ **Theorem 3.** *For each tuple (h,w,b), if $\Pi_{h,w,b}$ has a size-oblivious POT then (h,w,b) has the removal property.*

As a corollary of the above theorem, we infer that if one "representation" of a property as $\Pi_{h,w,b}$ has the removal property, then all such representations have the removal property. This is stated in the following corollary.

▶ **Corollary 4.** *Let (h,w,b) and (h',w',b') be tuples such that $\Pi_{h,w,b} = \Pi_{h',w',b'}$. Then (h,w,b) has the removal property if and only if (h',w',b') has the removal property.*

▶ Remark 5. Theorem 3 contradicts the statement of Proposition 3.14 in [6], which states that there is a tuple (h,w,b) such that $\Pi_{h,w,b}$ has a POT but (h,w,b) does not have the removal property. We believe that the proof of this proposition is wrong. This will be explained in the full version of this paper.

Goldreich and Shinkar conjectured (see [6, Open Problem 3.11]) that every property of the form $\Pi_{h,w,b}$ has a POT . Our next theorem disproves this conjecture by showing that there are properties $\Pi_{h,w,b}$ that are not testable at all (let alone testable using a POT).

▶ **Theorem 6.** *Let K_4 denote the complete graph on 4 vertices, D_4 the diamond graph (i.e. K_4 minus an edge), P_2 the graph on 4 vertices containing a path of length 2 and an isolated vertex, C_4 the 4-cycle, P_4 the path on 4 vertices and $K_{1,3}$ the star on 4 vertices. Let w_H be the following weight function assigning a non-negative weight to each graph on 4 vertices.*

H :	K_4	$\overline{K_4}$	D_4	$\overline{D_4}$	P_2	$\overline{P_2}$	C_4	$\overline{C_4}$	$K_{1,3}$	$\overline{K_{1,3}}$	P_4
w_H :	1	$\frac{1}{2}$	$\frac{5}{12}$	$\frac{5}{12}$	$\frac{1}{6}$	$\frac{1}{3}$	$\frac{1}{2}$	$\frac{1}{3}$	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$

Then, the property

$$\Pi_{h,w,b} = \left\{ G : \sum_{H:|V(H)|=4} w_H \cdot p(H,G) \leq \frac{5}{16} \right\} \tag{1}$$

is not *testable.*

Given the above theorem it is natural to ask if every property $\Pi_{h,w,b}$ can at least be tested using $o(n^2)$ edge-queries. We leave this as an open problem.

Paper overview

The rest of the paper is organized as follows. In Section 2 we prove Theorem 3. The proof of Theorem 6 appears in Section 3. The main idea behind its proof is to show that there exists a property $\Pi_{h,w,b}$ (specifically, the one defined in (1)) which encodes the property of being a pseudo-random graph. It is then not hard to show that such a property cannot be tested using a constant number of queries.

2 Proof of Theorem 3

In this section we prove Theorem 3 and Corollary 4. We will need the following auxiliary lemma.

▶ **Lemma 7.** *Let* Π *be a graph property, and suppose that* Π *has a canonical size-oblivious* ε*-tester* $\mathcal{T}$ *with sample complexity* $s = s(\varepsilon)$ *(and success probability* $\frac{2}{3}$*). Then for every* $n \geq s^4$ *and for every* n*-vertex graph* G *which is* ε*-far from* Π*, the following holds. For* U *chosen uniformly at random from* $\binom{V(G)}{s^4}$*, we have* $\mathbb{P}[G[U] \in \Pi] \leq e^{-\Omega(s)}$*.*

Proof. Let $\mathcal{A}$ be the family of all s-vertex graphs A such that $\mathcal{T}$ accepts if it sees a subgraph isomorphic to A. For a graph G, we say that a sequence of subsets $S_1, \ldots, S_s \in \binom{V(G)}{s}$ is *good* if $G[S_i] \in \mathcal{A}$ for at least half of the values of $1 \leq i \leq s$; otherwise $S_1, \ldots, S_s$ is *bad*. For a sequence of vertices $W = (x_1, \ldots, x_{s^2})$, we say that W is good (resp. bad) if $\{x_1, \ldots, x_s\}, \{x_{s+1}, \ldots, x_{2s}\}, \ldots, \{x_{s^2-s+1}, \ldots, x_{s^2}\}$ is good (resp. bad). Note that for a random $S \in \binom{V(G)}{s}$, if $G \in \Pi$ then $\mathbb{P}[G[S] \in \mathcal{A}] \geq \frac{2}{3}$, and if G is ε-far from Π then $\mathbb{P}[G[S] \in \mathcal{A}] \leq \frac{1}{3}$. Using a standard Chernoff-type bound, one can show that the following holds for $S_1, \ldots, S_s \in \binom{V(G)}{s}$ chosen uniformly at random and independently.

1. If $G \in \Pi$ then $S_1, \ldots, S_s$ is good with probability at least $1 - e^{-Cs}$.
2. If G is ε-far from Π then $S_1, \ldots, S_s$ is bad with probability at least $1 - e^{-Cs}$.

In both items above, C is an absolute constant.

The probability that $S_i \cap S_j \neq \emptyset$ for some $1 \leq i < j \leq s$ is at most $\binom{s}{2}\frac{s^2}{n} < \frac{1}{2}$, where the inequality follows from the assumption that $n \geq s^4$. So $|S_1 \cup \cdots \cup S_s| = s^2$ with probability at least $\frac{1}{2}$. Conditioned on the event that $S_1, \ldots, S_s$ are pairwise-disjoint, the set $S := S_1 \cup \cdots \cup S_s$ has the distribution of an element of $\binom{V(G)}{s^2}$ chosen uniformly at random. Thus, a random sequence of vertices $W = (x_1, \ldots, x_{s^2})$ chosen *without repetitions* satisfies the following.

1. If G satisfies Π then W is good with probability at least $1 - 2e^{-Cs}$.
2. If G is ε-far from Π then W is bad with probability at least $1 - 2e^{-Cs}$.

Now let G be a graph on $n \geq s^4$ vertices which is ε-far from Π. Consider a random pair (U, W), where U is chosen uniformly at random from $\binom{V(G)}{s^4}$, and $W = (x_1, \ldots, x_{s^2})$ is a sequence of vertices sampled randomly without repetition from U. Note that W is

distributed as a uniform sequence of s^2 vertices of G, sampled without repetition. Thus, $\mathbb{P}[W \text{ is good}] \leq 2e^{-Cs}$. On the other hand, if $G[U] \in \Pi$, then $\mathbb{P}[W \text{ is good} \mid U] \geq 1 - 2e^{-Cs}$. By combining these two facts, we see that

$$\mathbb{P}[G[U] \in \Pi] \leq \frac{2e^{-Cs}}{1-2e^{-Cs}} \leq 4e^{-Cs} = e^{-\Omega(s)}. \qquad \blacktriangleleft$$

Proof of Theorem 3. As mentioned in the introduction, a POT for $\Pi_{h,w,b}$ can be used to obtain a standard tester for $\Pi_{h,w,b}$ by invoking the POT an appropriate number of times. Moreover, it is clear that if the POT is size-oblivious, then so is the resulting tester. Next, we apply the transformation of Goldreich and Trevisan [7] to get a canonical tester $\mathcal{T}$ for $\Pi_{h,w,b}$. Since this transformation preserves the property of being size-oblivious, $\mathcal{T}$ is size-oblivious, and hence satisfies the condition of Lemma 7. Denote by $s = s(\varepsilon)$ the sample complexity of $\mathcal{T}$. We may and will assume that s is large enough as a function of the parameters h and b.

Let us denote $z(G) = \sum_H w_H \cdot p(H, G)$. By multiplying the inequality $\sum_H w_H \cdot p(H, G) \leq b$ by an appropriate integer, we can assume without loss of generality that b is an integer, and that w_H is an integer for every H.

Let G be a graph which is ε-far from $\Pi_{h,w,b}$. Our goal is to show that $z(G) \geq b + f(\varepsilon)$, for a function $f : (0, 1] \to (0, 1]$ to be chosen later. By Lemma 7, a randomly chosen $U \in \binom{V(G)}{s^4}$ satisfies $G[U] \in \Pi_{h,w,b}$ with probability at most $e^{-\Omega(s)}$. Observe that if a k-vertex graph K does not satisfy $\Pi_{h,w,b}$, then necessarily

$$z(K) = \sum_H w_H \cdot p(H, K) \geq b + \binom{k}{h}^{-1} > b + k^{-h},$$

as b and all weights w_H are integers. Thus, if $G[U] \notin \Pi_{h,w,b}$ then

$$z(G[U]) > b + |U|^{-h} = b + s^{-4h}.$$

Observe that $z(G)$ is the average of $z(G[U])$ over all $U \in \binom{V(G)}{s^4}$. Thus, using the guarantees of Lemma 7, we obtain

$$z(G) \geq (1 - e^{-\Omega(s)})(b + s^{-4h}) > b + \frac{1}{2}s^{-4h},$$

where the last inequality holds provided that s is large enough as a function of h and b. So we may take the function f in Definition 2 to be $f(\varepsilon) = \frac{1}{2}s(\varepsilon)^{-4h}$. This completes the proof. $\blacktriangleleft$

Proof of Corollary 4. We have established that (h, w, b) satisfies the removal property if and only if $\Pi_{h,w,b}$ has a size-oblivious POT. The "only if" part was explained in the introduction (see also [6]), and the "if" part is the statement of Theorem 3. Since the existence of a tester (specifically, a size-oblivious POT) does not depend on the specific representation of a given property as $\Pi_{h,w,b}$, it is now clear that the corollary holds. $\blacktriangleleft$

3 Proof of Theorem 6

Let $\Pi_{h,w,b}$ be as in the statement of Theorem 6. Denote

$$z(G) := \sum_{H:|V(H)|=4} w_H \cdot p(H, G)$$

for every graph G of order at least 4. Under this notation, $\Pi_{h,w,b} = \{G : z(G) \leq b\}$. For a pair of graphs H and G, define

$$t_{\text{inj}}(H,G) = \frac{1}{n^{\underline{h}}}|\{\varphi\colon V(H) \to V(G) \text{ injective s.t. } uv \in E(H) \Rightarrow \varphi(u)\varphi(v) \in E(G)\}|,$$

and

$$t_{\text{ind}}(H,G) = \frac{1}{n^{\underline{h}}}|\{\varphi\colon V(H) \to V(G) \text{ injective s.t. } uv \in E(H) \Leftrightarrow \varphi(u)\varphi(v) \in E(G)\}|,$$

where $n^{\underline{h}} = n \cdot (n-1) \cdot \cdots \cdot (n-h+1)$. Note that $t_{\text{ind}}(H,G) = p(H,G) \cdot \text{aut}(H)/h!$, where $\text{aut}(H)$ is the number of automorphisms of H. The following lemma gives a simpler description of $\Pi_{h,w,b}$.

▶ **Lemma 8.** *$\Pi_{h,w,b} = \{G : \phi(G) \leq 0\}$, where $\phi(G) = 2t_{inj}(C_4,G) - t_{inj}(K_2,G) + \frac{3}{8}$.*

Proof. First, note that $t_{inj}(K_2,G) = p(K_2,G)$. Next, we use the fact that

$$\begin{aligned} t_{\text{inj}}(C_4,G) &= t_{\text{ind}}(C_4,G) + 2t_{\text{ind}}(D_4,G) + t_{\text{ind}}(K_4,G) \\ &= \frac{\text{aut}(C_4)}{4!} \cdot p(C_4,G) + 2\frac{\text{aut}(D_4)}{4!} \cdot p(D_4,G) + \frac{\text{aut}(K_4)}{4!} \cdot p(K_4,G) \\ &= \frac{1}{3}p(C_4,G) + \frac{1}{3}p(D_4,G) + p(K_4,G). \end{aligned}$$

Hence,

$$\begin{aligned} \phi(G) &= \frac{2}{3}p(C_4,G) + \frac{2}{3}p(D_4,G) + 2p(K_4,G) - p(K_2,G) + \frac{3}{8} \\ &= \frac{2}{3}p(C_4,G) + \frac{2}{3}p(D_4,G) + 2p(K_4,G) + p(\overline{K_2},G) - \frac{5}{8} \\ &= \frac{2}{3}p(C_4,G) + \frac{2}{3}p(D_4,G) + 2p(K_4,G) + \sum_{H:|V(H)|=4} p(\overline{K_2},H)p(H,G) - \frac{5}{8} \\ &= \sum_{H:|V(H)|=4} 2w_H \cdot p(H,G) - \frac{5}{8}. \end{aligned}$$

Therefore, $\phi(G) \leq 0$ if and only if $\sum_{H:|V(H)|=4} w_H \cdot p(H,G) \leq 5/16$ ◀

An important ingredient in the proof of Theorem 6 is the following lemma, which shows that graphs in $\Pi_{h,w,b}$ are pseudo-random. In what follows, we write $x = y \pm z$ to mean that $x \in [y-z, y+z]$.

▶ **Lemma 9.** *For every $\delta \in (0,1)$ there is $n_0(\delta)$ such that every graph $G \in \Pi_{h,w,b}$ on $n \geq n_0(\delta)$ vertices satisfies the following. For every $U, V \subseteq V(G)$ with $|U|, |V| \geq \delta n$, it holds that*

$$e(U,V) = \left(\frac{1}{2} \pm \delta\right)|U||V|\,.$$

Proof. We start by showing that for every $\gamma \in (0,1)$ there is $n_0(\gamma)$ such that if $G \in \Pi_{h,w,b}$ is a graph on $n \geq n_0(\gamma)$ vertices, then

$$t_{\text{inj}}(K_2,G) = \frac{1}{2} \pm \gamma \text{ and } t_{\text{inj}}(C_4,G) = \frac{1}{16} \pm \gamma. \tag{2}$$

It is a well-known fact (see for instance [11]) that every n-vertex graph G satisfies[1]

$$t_{\text{inj}}(C_4,G) \geq t_{\text{inj}}(K_2,G)^4 - O\left(\frac{1}{n}\right) \geq t_{\text{inj}}(K_2,G)^4 - \frac{\gamma^2}{2}, \tag{3}$$

where the last inequality holds if n is large enough. Now, observe that every $G \in \Pi_{h,w,b}$ satisfies

$$2t_{\text{inj}}(K_2,G)^4 - t_{\text{inj}}(K_2,G) + \frac{3}{8} \leq 2t_{\text{inj}}(C_4,G) + \gamma^2 - t_{\text{inj}}(K_2,G) + \frac{3}{8} = \phi(G) + \gamma^2 \leq \gamma^2, \tag{4}$$

where the last inequality follows from Lemma 8. Note that the function $x \mapsto 2x^4 - x + \frac{3}{8}$ is convex, and attains its minimum at $x = 1/2$. Therefore, if we had $t_{\text{inj}}(K_2,G) > \frac{1}{2} + \gamma$, then we would have

$$2t_{\text{inj}}(K_2,G)^4 - t_{\text{inj}}(K_2,G) + \frac{3}{8} > 2\left(\frac{1}{2}+\gamma\right)^4 - \left(\frac{1}{2}+\gamma\right) + \frac{3}{8} = 2\gamma^4 + 4\gamma^3 + 3\gamma^2 > \gamma^2.$$

Similarly, if we had $t_{\text{inj}}(K_2,G) < \frac{1}{2} - \gamma$, then we would have

$$2t_{\text{inj}}(K_2,G)^4 - t_{\text{inj}}(K_2,G) + \frac{3}{8} > 2\left(\frac{1}{2}-\gamma\right)^4 - \left(\frac{1}{2}-\gamma\right) + \frac{3}{8} = 2\gamma^4 - 4\gamma^3 + 3\gamma^2 > \gamma^2.$$

In any case, we see that $|t_{\text{inj}}(K_2,G) - \frac{1}{2}| > \gamma$ would stand in contradiction to (4). Hence, $t_{\text{inj}}(K_2,G) = \frac{1}{2} \pm \gamma$. By applying Lemma 8 again we get $t_{\text{inj}}(C_4,G) \leq \frac{1}{16} + \gamma$. By using the intermediate inequality in (3) and $t_{\text{inj}}(K_2,G) \geq \frac{1}{2} - \gamma$, we get

$$t_{\text{inj}}(C_4,G) \geq \left(\frac{1}{2}-\gamma\right)^4 - O\left(\frac{1}{n}\right) \geq \frac{1}{16} - \gamma,$$

where the last inequality can be easily verified, assuming that n is large enough, by expanding the binomial expression. We have thus established (2).

A well-known result of Chung, Graham and Wilson [2] states that for every $\delta \in (0,1)$ there is $\gamma = \gamma(\delta)$ such that if a graph G satisfies (2), then for every $U, V \subseteq V(G)$ with $|U|, |V| \geq \delta n$, it holds that $e(U,V) = \left(\frac{1}{2} \pm \delta\right)|U||V|$. The lemma follows by combining this result with the above. ◀

For a family of graphs $\mathcal{F}$ and a graph G, we define

$$p(\mathcal{F},G) = \sum_{F \in \mathcal{F}} p(F,G)\ .$$

It is well-known (see e.g. [2]) that a pseudo-random graph has approximately the same distribution of small subgraphs as a random graph with the same density. By combining this with Lemma 9, we obtain the following corollary. Note that the expected value of $p(F, G(n, \frac{1}{2}))$ is $2^{-\binom{s}{2}} \frac{s!}{\text{aut}(F)}$.

▶ **Corollary 10.** *For every $s \geq 2$ and $\delta \in (0,1)$ there is $n_1 = n_1(s,\delta)$ such that every $G \in \Pi_{h,w,b}$ on $n \geq n_1$ vertices satisfies the following. For every family $\mathcal{F}$ of s-vertex graphs, it holds that*

$$p(\mathcal{F},G) = \sum_{F \in \mathcal{F}} 2^{-\binom{s}{2}} \frac{s!}{aut(F)} \pm \delta.$$

[1] Usually this inequality is stated in terms of the *homomorphic density*, as $\mathrm{t}(C_4,G) \geq \mathrm{t}(K_2,G)^4$ (see e.g. [9]). The error-term $O(\frac{1}{n})$ accounts for the difference between the homomorphic density and the injective density.

We are now ready to prove Theorem 6.

Proof of Theorem 6. We start by showing that $\Pi_{h,w,b}$ is non-empty. More specifically, we prove that for every integer $n \geq 4$, there exists an n-vertex graph satisfying $\Pi_{h,w,b}$. Let $G \sim G(n, \frac{1}{2})$. It is easy to see that $\mathbb{E}[t_{\text{inj}}(K_2, G)] = \frac{1}{2}$ and $\mathbb{E}[t_{\text{inj}}(C_4, G)] = \frac{1}{16}$. Hence,

$$\mathbb{E}[\phi(G)] = 2\mathbb{E}[t_{\text{inj}}(C_4, G)] - \mathbb{E}[t_{\text{inj}}(K_2, G)] + \frac{3}{8} = 0.$$

It follows that there is an n-vertex graph with $\phi(G) \leq 0$, and hence $G \in \Pi_{h,w,b}$ by Lemma 8.

Now suppose by contradiction that $\Pi_{h,w,b}$ is testable. In particular, there exists a 0.1-tester for $\Pi_{h,w,b}$. This implies – by [7, Theorem 2] (see also [8]) – that there is a canonical 0.1-tester $\mathcal{T}$ for $\Pi_{h,w,b}$. Denote by s the sample complexity of $\mathcal{T}$. Then for every $n > 0$ there is a family $\mathcal{F} = \mathcal{F}(n)$ of (rejection) graphs of order s satisfying the following for every n-vertex graph G.

1. $p(\mathcal{F}, G) \leq \frac{1}{3}$ if $G \in \Pi_{h,w,b}$;
2. $p(\mathcal{F}, G) \geq \frac{2}{3}$ if G is 0.1-far from $\Pi_{h,w,b}$.

Let $n' = \max\{9s^2, n_1(s, \frac{1}{9})\}$ and $n = \max\{n_0(\frac{1}{n'}), n_1(s, \frac{1}{9})\}$, where n_1 is from Corollary 10 and n_0 is from Lemma 9. Let G' be an arbitrary n'-vertex graph which satisfies $\Pi_{h,w,b}$, and let G be the $\frac{n}{n'}$-blow-up of G'. Denote by $V_1 \sqcup \cdots \sqcup V_{n'} = V(G)$ the clusters of this blow-up.

We claim that G is 0.1-far from $\Pi_{h,w,b}$. Indeed, fix any $G^* \in \Pi_{h,w,b}$ with n vertices. By our choice of n via Lemma 9, and as $|V_i| = \frac{n}{n'}$, we must have $e_{G^*}(V_i, V_j) = (\frac{1}{2} \pm \frac{1}{n'})(n/n')^2$. But since $e_G(V_i, V_j) \in \{0, (n/n')^2\}$, we must change at least $(\frac{1}{2} - \frac{1}{n'})(n/n')^2 \geq 0.4(n/n')^2$ edges between V_i and V_j for every $1 \leq i < j \leq n'$, in order to turn G into G^*. Therefore, the edit distance between G and G^* is at least $\binom{n'}{2} \cdot 0.4(n/n')^2 \geq 0.1n^2$, as required.

Now, let $S \in \binom{V(G)}{s}$ be chosen uniformly at random, and let $\mathcal{B}$ be the event that there exists $1 \leq i \leq n'$ for which $|S \cap V_i| > 1$. Note that we have $\mathbb{P}(\mathcal{B}) \leq \binom{s}{2}/n' < \frac{1}{9}$, by the choice of n'. Observe that conditioned on $\mathcal{B}^c$, the probability that S is isomorphic to an s-vertex graph F is exactly $p(F, G')$. Hence, setting $\mathcal{F} = \mathcal{F}(n)$ and $\rho = \sum_{F \in \mathcal{F}} 2^{-\binom{s}{2}} \frac{s!}{\text{aut}(F)}$, we have

$$\begin{aligned} p(\mathcal{F}, G) &= \mathbb{P}[G[S] \in \mathcal{F}] = \mathbb{P}(G[S] \in \mathcal{F} \mid B^c) + \mathbb{P}(B) \\ &< \mathbb{P}(G[S] \in \mathcal{F} \mid B^c) + \frac{1}{9} \\ &= p(\mathcal{F}, G') + \frac{1}{9} \\ &\leq \sum_{F \in \mathcal{F}} 2^{-\binom{s}{2}} \frac{s!}{\text{aut}(F)} + \frac{1}{9} + \frac{1}{9} \\ &= \rho + \frac{2}{9}, \end{aligned} \tag{5}$$

where in the last inequality we used our choice of n' via Corollary 10. On the other hand, our choice of n via Corollary 10 implies that every n-vertex graph $G^* \in \Pi_{h,w,b}$ satisfies

$$p(\mathcal{F}, G^*) \geq \sum_{F \in \mathcal{F}} 2^{-\binom{s}{2}} \frac{s!}{\text{aut}(F)} - \frac{1}{9} = \rho - \frac{1}{9}.$$

By combining this with (5), we get $p(\mathcal{F}, G^*) > p(\mathcal{F}, G) - \frac{1}{3}$. But this stands in contradiction to $p(\mathcal{F}, G^*) \leq \frac{1}{3}$ (as $G^* \in \Pi_{h,w,b}$) and $p(\mathcal{F}, G) \geq \frac{2}{3}$ (as G is 0.1-far from $\Pi_{h,w,b}$). This completes the proof of the theorem. ◀

References

1 N. Alon, E. Fischer, M. Krivelevich, and M. Szegedy. Efficient testing of large graphs. *Combinatorica*, 20:451–476, 2000.
2 F.R.K Chung, R.L Graham, and R.M. Wilson. Quasi-random graphs. *Combinatorica*, 9:345–362, 1989.
3 O. Goldreich. *Introduction to Property Testing*. Cambridge University Press, 2017.
4 O. Goldreich, S. Goldwasser, and D. Ron. Property testing and its connection to learning and approximation. *J. ACM*, 45:653–750, 1998.
5 O. Goldreich and D. Ron. On proximity oblivious testing. *SIAM J. Comput*, 40:534–566, 2011.
6 O. Goldreich and I. Shinkar. Two-sided error proximity oblivious testing. *Random Structures Algorithms*, 48:341–383, 2016.
7 O. Goldreich and L. Trevisan. Three theorems regarding testing graph properties. *Random Structures Algorithms*, 23:23–57, 2003.
8 O. Goldreich and L. Trevisan. Errata to Three theorems regarding testing graph properties. available from `http://www.wisdom.weizmann.ac.il/~oded/p_ttt.html`, 2005.
9 L. Lovász. *Large networks and graph limits*. Providence: American Mathematical Society, 2012.
10 R. Rubinfeld and M. Sudan. Robust characterizations of polynomials with applications to program testing. *SIAM J. Comput.*, 25:252–271, 1996.
11 A. Sidorenko. A correlation inequality for bipartite graphs. *Graphs and Combinatorics*, 9:201–204, 1993.

Top-Down Induction of Decision Trees: Rigorous Guarantees and Inherent Limitations

Guy Blanc
Stanford University, CA, USA

Jane Lange
Stanford University, CA, USA

Li-Yang Tan
Stanford University, CA, USA

Abstract

Consider the following heuristic for building a decision tree for a function $f : \{0,1\}^n \to \{\pm 1\}$. Place the *most influential variable* x_i of f at the root, and recurse on the subfunctions $f_{x_i=0}$ and $f_{x_i=1}$ on the left and right subtrees respectively; terminate once the tree is an ε-approximation of f. We analyze the quality of this heuristic, obtaining near-matching upper and lower bounds:

- *Upper bound:* For every f with decision tree size s and every $\varepsilon \in (0, \frac{1}{2})$, this heuristic builds a decision tree of size at most $s^{O(\log(s/\varepsilon)\log(1/\varepsilon))}$.
- *Lower bound:* For every $\varepsilon \in (0, \frac{1}{2})$ and $s \leq 2^{\tilde{O}(\sqrt{n})}$, there is an f with decision tree size s such that this heuristic builds a decision tree of size $s^{\tilde{\Omega}(\log s)}$.

We also obtain upper and lower bounds for monotone functions: $s^{O(\sqrt{\log s}/\varepsilon)}$ and $s^{\tilde{\Omega}(\sqrt[4]{\log s})}$ respectively. The lower bound disproves conjectures of Fiat and Pechyony (2004) and Lee (2009).

Our upper bounds yield new algorithms for properly learning decision trees under the uniform distribution. We show that these algorithms – which are motivated by widely employed and empirically successful top-down decision tree learning heuristics such as ID3, C4.5, and CART – achieve provable guarantees that compare favorably with those of the current fastest algorithm (Ehrenfeucht and Haussler, 1989), and even have certain qualitative advantages. Our lower bounds shed new light on the limitations of these heuristics.

Finally, we revisit the classic work of Ehrenfeucht and Haussler. We extend it to give the first uniform-distribution proper learning algorithm that achieves polynomial sample and memory complexity, while matching its state-of-the-art quasipolynomial runtime.

2012 ACM Subject Classification Theory of computation → Oracles and decision trees; Theory of computation → Boolean function learning

Keywords and phrases Decision trees, Influence of variables, Analysis of boolean functions, Learning theory, Top-down decision tree heuristics

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.44

Funding *Li-Yang Tan*: Supported by NSF grant CCF-1921795.

Acknowledgements We thank Clément Canonne, Adam Klivans, Charlotte Peale, Toniann Pitassi, Omer Reingold, and Rocco Servedio for helpful conversations and suggestions. We also thank the anonymous reviewers of ITCS 2020 for their feedback.

1 Introduction

Consider the problem of constructing a *decision tree* representation of a function $f : \{0,1\}^n \to \{\pm 1\}$, where the goal is to build a decision tree for f that is as small as possible, ideally of size close to the optimal decision tree size of f. Perhaps the simplest and most natural approach is to proceed in a *top-down*, greedy fashion:

1. Choose a "good" variable x_i to query as the root of the decision tree;
2. Build the left and right subtrees by recursing on the subfunctions $f_{x_i=0}$ and $f_{x_i=1}$ respectively.

This reduces the task of building a decision tree to that of choosing the root variable – i.e. determining the *splitting criterion* of this top-down heuristic. Intuitively, a good root variable should be one that is very "relevant" and "important" in terms of determining the value of f; it is reasonable to expect that querying such a variable first would reduce the number of subsequent queries necessary. Our focus in this paper will be on a specific splitting criterion: *influence*.

▶ **Definition 1** (Influence). *The influence of the variable x_i on a function $f : \{0,1\}^n \to \{\pm 1\}$ is defined to be*

$$\mathrm{Inf}_i(f) \coloneqq \Pr_{\boldsymbol{x} \sim \{0,1\}^n}[f(\boldsymbol{x}) \neq f(\boldsymbol{x}^{\oplus i})],$$

where $\boldsymbol{x}$ is drawn uniformly at random, and $\boldsymbol{x}^{\oplus i}$ denotes $\boldsymbol{x}$ with its i-th coordinate flipped.

Influence is a fundamental and well-studied notion in the analysis of boolean functions [51]. It is the key quantity of interest in many landmark results (e.g. the KKL inequality [35], Friedgut's junta theorem [24], the Invariance Principle [50]) and open problems (e.g. the Gotsman–Linial conjecture [29], the Aaronson–Ambainis conjecture [1], the Fourier Entropy-Influence conjecture [25]) of the field. Beyond the analysis of boolean functions, this notion has been widely employed across both algorithms and complexity theory, where it has indeed proven to be a useful quantitative measure of the relevance and importance of a variable. Most relevant to the algorithmic applications in this paper, influence has been a key enabling ingredient in a large number of results in learning theory [15, 9, 45, 63, 53, 54, 31, 19, 36, 37, 5].

1.1 Influence as a splitting criterion

We now give a formal description of the heuristic for constructing decision trees that we study. We define a *bare tree* to be a decision tree with unlabeled leaves, and write T° to denote such trees. We refer to any decision tree T obtained from T° by a labelling of its leaves as a *completion* of T°. Given a bare tree T° and a function f, there is a canonical completion of T° that minimizes the approximation error with respect to f:

▶ **Definition 2** (f-completion of a bare tree). *Let T° be a bare tree and $f : \{0,1\}^n \to \{\pm 1\}$. Consider the following completion of T°: for every leaf ℓ in T°, label it* $\mathrm{sign}(\mathbb{E}[f_\ell(\boldsymbol{x})])$*, where f_ℓ is the restriction of f by the path leading to ℓ and $\boldsymbol{x} \sim \{0,1\}^n$ is uniform random. This completion minimizes the approximation error* $\Pr[T(\boldsymbol{x}) \neq f(\boldsymbol{x})]$*, and we refer to it as the* f-completion of T°.

In addition to the function f, our heuristic will also take in an error parameter ε, allowing us to construct both *exact* ($\varepsilon = 0$) and *approximate* ($\varepsilon \in (0, \frac{1}{2})$) decision tree representations of f.

BuildTopDownDT(f, ε):

Initialize T° to be the empty tree.

while (f-completion of T° is not an ε-approximation of f) {

1. (Score) For every leaf ℓ in T°, let $x_{i(\ell)}$ denote the most influential variable of the subfunction f_ℓ:

$$\mathrm{Inf}_{i(\ell)}(f_\ell) \geq \mathrm{Inf}_j(f_\ell) \quad \text{for all } j \in [n].$$

   Assign ℓ the score:

$$\mathrm{score}(\ell) \coloneqq \Pr_{\boldsymbol{x} \sim \{0,1\}^n}[\boldsymbol{x} \text{ reaches } \ell] \cdot \mathrm{Inf}_{i(\ell)}(f_\ell) = 2^{-|\ell|} \cdot \mathrm{Inf}_{i(\ell)}(f_\ell),$$

   where $|\ell|$ denotes the depth of ℓ in T°.
2. (Split) Let $\ell^\star$ be the leaf with the highest score. Grow T° by replacing $\ell^\star$ with a query to $x_{i(\ell^\star)}$.

}

Figure 1 Top-down heuristic for building an ε-approximate decision tree representation of f, with influence as the splitting criterion.

In words, BuildTopDownDT builds a bare tree T° in a top-down fashion, starting from the empty tree. In each iteration, we first check if the f-completion of T° is an ε-approximation of f, and if so, we output the completion. Otherwise, we split the leaf $\ell^\star$ with the highest score by querying the most influential variable of $f_{\ell^\star}$, where the score of a leaf ℓ is the influence of the most influential variable of f_ℓ normalized by the depth of ℓ within T°.[1]

1.2 This work

By design, the decision tree returned by BuildTopDownDT(f, ε) is an ε-approximation of f. We write TopDownDTSize(f, ε) to denote the size of this tree, and when $\varepsilon = 0$, we simply write TopDownDTSize(f). The question that motivates our work is:

What guarantees can we make on TopDownDTSize(f, ε) *as a function of the optimal decision tree size of* f *and* ε*?*

That is, we would like to understand the quality of BuildTopDownDT as a heuristic for constructing exact and approximate decision tree representations. In addition to being a natural structural question concerning decision trees, this question also has implications in learning theory. Indeed, BuildTopDownDT is motivated by top-down decision tree learning heuristics such as ID3, C4.5, and CART that are widely employed and empirically successful in machine learning practice. We discuss the learning-theoretic context and applications of our structural results in Section 2.1, and the connection to practical machine learning heuristics in Section 3.1.

[1] There are two possibilities for ties in BuildTopDownDT: two variables may have the same influence within a subfunction f_ℓ, and two leaves may have the same score. Our upper bounds hold regardless of how ties are broken, and our lower bounds hold even if ties are broken in the most favorable way.

To our knowledge, the question above has not been studied in such generality. The most directly relevant prior work is that of Fiat and Pechyony [23], who considered the case when f is either a linear threshold function or a read-once DNF formula, and the setting of exact representation ($\varepsilon = 0$). For such functions, they proved that the heuristic builds an exact decision tree representation of optimal size. We give an overview of other related work in Section 3.2.

2 Our results

As our first contribution, we give near-matching upper and lower bounds that provide a fairly complete answer to the question above. Our upper bound is as follows:

▶ **Theorem 3** (Upper bound for approximate representation). *For every $\varepsilon \in (0, \frac{1}{2})$ and every size-s decision tree f, we have* $\textsc{TopDownDTSize}(f, \varepsilon) \leq s^{O(\log(s/\varepsilon)\log(1/\varepsilon))}$.

We complement Theorem 3 with lower bounds showing that (a) for exact representation ($\varepsilon = 0$), no non-trivial upper bound can be obtained; and (b) for approximate representation ($\varepsilon \in (0, \frac{1}{2})$), the dependence on s in Theorem 3 is essentially optimal:

▶ **Theorem 4** (Lower bounds for exact and approximate representations).

(a) Exact representation: *There is an $f : \{0,1\}^n \to \{\pm 1\}$ with decision tree size $s = \Theta(n)$ such that* $\textsc{TopDownDTSize}(f) \geq 2^{\Omega(s)}$.

(b) Approximate representation: *For every $\varepsilon \in (0, \frac{1}{2})$ and function $s(n) \leq 2^{\tilde{O}(\sqrt{n})}$, there is an $f : \{0,1\}^n \to \{\pm 1\}$ with decision tree size s such that* $\textsc{TopDownDTSize}(f, \varepsilon) \geq s^{\tilde{\Omega}(\log s)}$.

Prior to our work, it was not known whether an upper bound of $\textsc{TopDownDTSize}(f, \varepsilon) \leq \mathrm{poly}(s, 1/\varepsilon)$ holds for all size-s decision trees f and $\varepsilon \in (0, \frac{1}{2})$; Theorem 4(b) provides a strong negative answer. Indeed, such an upper bound had been conjectured to hold for the class of *monotone* functions [44]. We now discuss our results on monotone functions, which disprove this conjecture, along with a stronger variant of it for exact representation [23].

Monotone functions

A monotone boolean function $f : \{0,1\}^n \to \{\pm 1\}$ is one that satisfies $f(x) \leq f(y)$ for all $x \preceq y$ (where $x \preceq y$ iff $x_i \leq y_i$ for all $i \in [n]$). An elementary and useful fact about monotone functions is that the influence of a variable on a monotone function f is equivalent to its *correlation* with f:

▶ **Fact 5** (Influence ≡ correlation for monotone functions). *For all monotone functions $f : \{0,1\}^n \to \{\pm 1\}$ and $i \in [n]$, we have* $\mathrm{Inf}_i[f] = 2\,\mathbb{E}[f(\boldsymbol{x})\boldsymbol{x}_i] - \mathbb{E}[f(\boldsymbol{x})]$.[2]

Therefore, for monotone functions, splitting on the most influential variable of a subfunction is equivalent to splitting on the variable that has the *highest correlation* with the subfunction.[3]

---

[2] The equivalence between influence and correlation for monotone functions is more transparent if one works with $\{\pm 1\}^n$ instead of $\{0,1\}^n$ as the domain: for monotone functions $f : \{\pm 1\}^n \to \{\pm 1\}$, we have $\mathrm{Inf}_i[f] = \mathbb{E}[f(\boldsymbol{x})\boldsymbol{x}_i]$.

[3] We observe that for general *non-monotone* functions, correlation can in general be a very poor splitting criterion, in the sense of building a decision tree that is much larger than the optimal decision tree. Consider $f : \{0,1\}^n \to \{\pm 1\}$ where $f(x) = x_j \oplus x_k$, the parity of two variables. The optimal decision tree size of f is 4, but since $\mathbb{E}[f(\boldsymbol{x})\boldsymbol{x}_i] = 0$ for all $i \in [n]$, the top-down heuristic using correlation as its splitting criterion may build a tree of size $\Omega(2^n)$ before achieving any non-trivial accuracy $\varepsilon < \frac{1}{2}$. (On the other hand, the top-down heuristic using influence as its splitting criterion would build the optimal tree of size 4.) We revisit this observation in Section 3.1.

Our proof of Theorem 3 extends in a straightforward manner to give a different upper bound under the assumption of monotonicity, where the dependence on s is significantly better. We refer to a size-s decision tree computing a monotone function as a size-s monotone decision tree.

▶ **Theorem 6** (Upper bound for approximate representation of monotone functions.). *For every $\varepsilon \in (0, \frac{1}{2})$ and every size-s monotone decision tree f, we have* $\textsc{TopDownDTSize}(f, \varepsilon) \leq s^{O(\sqrt{\log s}/\varepsilon)}$.

In analogy with Theorem 4, we also obtain lower bounds for exact and approximate representations of monotone functions:

▶ **Theorem 7** (Lower bounds for exact and approximate representations of monotone functions).

(a) Exact representation: *There is a monotone $f : \{0,1\}^n \to \{\pm 1\}$ with decision tree size $s = \Theta(n)$ such that* $\textsc{TopDownDTSize}(f) \geq 2^{\Omega(s)}$.

(b) Approximate representation: *For every $\varepsilon \in (0, \frac{1}{2})$ and function $s(n) \leq 2^{\tilde{O}(n^{4/5})}$, there is an $f : \{0,1\}^n \to \{\pm 1\}$ with decision tree size s such that* $\textsc{TopDownDTSize}(f, \varepsilon) \geq s^{\tilde{\Omega}(\sqrt[4]{\log s})}$.

Although we have stated Theorem 7 in terms of the specific heuristic BuildTopDownDT that we study, the actual lower bounds that we establish are significantly stronger: they apply to all "impurity-based top-down heuristics". This is a broad class that captures a wide variety of decision tree learning heuristics used in machine learning practice, including ID3, C4.5, and CART; see Section 3.1 for details.

▶ **Theorem 8** (Stengthening of Theorem 7(b)). *For every $\varepsilon \in (0, \frac{1}{2})$ and function $s(n) \leq 2^{\tilde{O}(n^{4/5})}$, there is a size-$s$ monotone decision tree f such that the ε-approximator built by any impurity-based top-down heuristic must have size $s^{\tilde{\Omega}(\sqrt[4]{\log s})}$.*

Disproving conjectures of Fiat–Pechyony and Lee

Motivated by applications in learning theory (discussed next in Section 2.1), Fiat and Pechyony [23] and Lee [44] also considered the quality of BuildTopDownDT as a heuristic for building decision trees for monotone functions.

[23] conjectured that for all monotone functions f, even in the case of exact representation ($\varepsilon = 0$), BuildTopDownDT returns a tree of minimal depth and size "not far from minimal". Theorem 7(a) provides a counterexample to the conjectured bound on size, and the function in Theorem 7(b) disproves the conjecture about depth; see Remark 40.[4]

Stated in the notation of our paper, [44] raised the possibility that $\textsc{TopDownDTSize}(f, \varepsilon) \leq \text{poly}(s, 1/\varepsilon)$ for all size-s monotone decision trees f and $\varepsilon \in (0, \frac{1}{2})$. The author further remarked that "showing $\textsc{TopDownDTSize}(f, \varepsilon) \leq \text{poly}(s)$, even only for constant accuracy ε,[5] would be a huge advance". Theorem 7(b) rules this out.

[4] For clarity of exposition, throughout this overview we discuss our results with decision tree *size* as the complexity measure. There are analogues of all of our results, both upper and lower bounds, for decision tree *depth* as the complexity measure.

[5] That is, a bound of the form $\textsc{TopDownDTSize}(f, \varepsilon) \leq s^{O_\varepsilon(1)}$.

2.1 Algorithmic applications: Properly learning decision trees

Learning decision trees has been a touchstone problem in uniform-distribution PAC learning for more than thirty years. It sits right at the boundary of our understanding of efficient learnability, and continues to be the subject of intensive research. The seminal work of Ehrenfeucht and Haussler [21] gave a $\mathrm{poly}(n^{\log s}, 1/\varepsilon)$-time algorithm for learning decision trees using random examples (see also [8] for an alternative proof based on Rivest's algorithm for learning decision lists [59]);[6] subsequently, Linial, Mansour, and Nisan [45] gave an algorithm that also runs in quasipolynomial time, but achieves polynomial sample complexity; Kusilevitz and Mansour [43], leveraging a novel connection to cryptography [29], gave a polynomial-time algorithm using membership queries; Gopalan, Kalai, and Klivans [30] obtained an *agnostic* analogue of [43]'s algorithm, extending it to tolerate adversarial noise; O'Donnell and Servedio [53] gave a polynomial-time algorithm for learning *monotone* decision trees from random examples; recent work of Hazan, Klivans, and Yuan [33] gives an algorithm agnostically learning decision trees with *polynomial sample complexity*; even more recent work of Chen and Moitra [17] gives an algorithm for learning *stochastic* decision trees.

Properly learning decision trees

When learning decision trees, it is natural to seek a hypothesis that is itself a decision tree. Indeed, it may be natural to seek a decision tree hypothesis even when learning other concept classes. The simple structure of decision trees makes them desirable both in terms of interpretability and explanatory power, which is why they are ubiquitous in empirical machine learning. A further advantage of decision tree hypotheses is that they are very fast to evaluate: evaluating a depth-d decision tree on a given input takes time $O(d)$,[7] whereas evaluating say a degree-d polynomial – another canonical and ubiquitous representation class in learning theory – can take time $\Theta(n^d)$, the number of monomials in the polynomial.

In learning theory, algorithms that return a hypothesis belonging to the concept class are known as *proper*. Understanding the complexity of proper learning (vis-à-vis improper learning) is an important research direction in learning theory [22]; proper learning also has deep connections to proof complexity [2] and property testing [28].

2.1.1 New proper learning algorithms

Among the decision tree learning algorithms discussed at the beginning of this subsection, the only one that is proper is the one of Ehrenfeucht and Haussler [21]. Our upper bounds on TopDownDTSize yield new algorithms for properly learning decision trees under the uniform distribution:

▶ **Theorem 9** (Algorithmic consequence of Theorem 3)**.** *Size-s decision trees can be properly learned under the uniform distribution in time* $\mathrm{poly}(n, s^{\log(s/\varepsilon)\log(1/\varepsilon)})$ *using membership queries.*[8]

---

[6] In fact, the algorithm of [21] learns decision trees in the more challenging setting of *distribution-free* PAC learning. All other results in this discussed in this section, including ours, are specific to uniform-distribution learning, and we focus our exposition on this setting.

[7] Every size-s decision tree is well-approximated by a decision tree of depth $O(\log s)$.

[8] We remark that our algorithm only requires fairly "mild" use of membership queries. Our algorithm only requires *random edge samples* (Definition 43), and hence falls within both the random walk model of Bshouty et al. [16] and the local membership queries model of Awasthi et al. [3]. These (incomparable) models are natural relaxations of the standard model of learning from random examples, and do not allow the learning algorithm unrestricted membership query access to the target function.

Analogously, Theorem 6 yields a new algorithm for learning monotone decision trees using only random examples. The learnability of monotone functions with respect to various complexity measures has been the subject of intensive study in uniform-distribution learning [32, 41, 39, 14, 15, 9, 66, 60, 63, 53, 62, 18, 44, 34, 54].

▶ **Theorem 10** (Algorithmic consequence of Theorems 3 and 6). *Size-s monotone decision trees can be properly learned under the uniform distribution in time*

$$\mathrm{poly}(n, \min(s^{O(\log(s/\varepsilon)\log(1/\varepsilon))}, s^{O(\sqrt{\log s}/\varepsilon)}))$$

using only random examples.

We now compare our results with the prior state of the art for properly learning decision trees.

- **Polynomial-time algorithms for superlogarithmic size.** Theorems 9 and 10 give the first polynomial-time algorithms for properly learning decision trees of size $\omega(\log n)$ to constant accuracy. To see this, we first note that [21]'s runtime of $\mathrm{poly}(n^{\log s}, 1/\varepsilon)$ is superpolynomial time for any $s = \omega(1)$. Alternatively, functions depending on $k \ll n$ variables ("k-juntas") can be properly learned in time $\mathrm{poly}(n, 2^k)$, using random examples for monotone juntas, and membership queries otherwise [10, 49]. Since every size-s decision tree certainly depends on at most $k \leq s$ variables, this runtime is polynomial for decision trees of size $s = O(\log n)$, but becomes superpolynomial once $s = \omega(\log n)$. In contrast, the runtimes of our algorithms in Theorems 9 and 10 remain polynomial for $s = 2^{\Omega(\sqrt{\log n})}$ and $s = 2^{\Omega((\log n)^{2/3})}$ respectively.
- **Dimension-independent hypothesis size.** Related to the above, the sizes of the hypotheses returned by the algorithms of Theorems 9 and 10 are $s^{O(\log(s/\varepsilon)\log(1/\varepsilon))}$ and $s^{O(\sqrt{\log s}/\varepsilon)}$ respectively, independent of n, whereas the size of the hypotheses returned by [21]'s algorithm can be as large as $n^{\Omega(\log s)}$. This is gap can be exponential or even larger for small values of s.
- **Average depth as the complexity measure.** Our algorithms and analyses extend easily to accommodate *average depth* as the complexity measure. The average depth of a decision tree, $\triangle(T)$, is the number of queries T makes on a uniform random input. Average depth is a stronger complexity measure than size since $\triangle(T) \leq \log(\mathrm{size}(T))$.[9]

  ▶ **Theorem 11** (Learning trees with small average depth). *Decision trees of average depth $\triangle$ can be properly learned under the uniform distribution in time* $\mathrm{poly}(n, 2^{\triangle^2/\varepsilon})$ *using membership queries, and monotone decision trees of average depth $\triangle$ can be properly learned in time* $\mathrm{poly}(n, 2^{\triangle^{3/2}/\varepsilon})$ *using random examples.*

  To our knowledge, these represent the first polynomial-time algorithms for properly learning decision trees of superconstant average depth, $\triangle = \omega(1)$. Prior to our work, the fastest algorithm ran in time $\mathrm{poly}(n^{\triangle/\varepsilon})$; this algorithm, which uses random examples, follows implicitly from the results of Mehta and Raghavan [46].

[9] Furthermore, it is easy to construct examples of decision trees T with the largest possible gap between these measures: $\triangle(T) = O(1)$ and $\log(\mathrm{size}(T)) = \Omega(n)$.

2.2 Proper learning with polynomial sample and memory complexity

For our final contribution, we revisit the classic algorithm of Ehrenfeucht and Haussler [21]. As discussed above, this remains the fastest algorithm for properly learning decision trees. We extend it to give the first uniform-distribution proper algorithm that achieves polynomial sample and memory complexity, while matching its state-of-the-art quasipolynomial runtime (Theorem 54).

Table 1 Algorithms for learning size-s decision trees from random examples under the uniform distribution.

Reference	Running time	Sample complexity	Memory complexity	Proper?
[21]	$\mathrm{poly}(n^{\log s}, 1/\varepsilon)$	$\mathrm{poly}(n^{\log s}, 1/\varepsilon)$	$\mathrm{poly}(n^{\log s}, 1/\varepsilon)$	✓
[45]	$\mathrm{poly}(n^{\log(s/\varepsilon)})$	$\mathrm{poly}(s, 1/\varepsilon) \cdot \log n$	$\mathrm{poly}(n, s, 1/\varepsilon)$	×
[46]	$\mathrm{poly}(n^{\log(s/\varepsilon)})$	$\mathrm{poly}(s, 1/\varepsilon) \cdot \log n$	$\mathrm{poly}(n^{\log(s/\varepsilon)})$	✓
This work	$\mathrm{poly}(n^{\log s}, 1/\varepsilon)$	$\mathrm{poly}(s, 1/\varepsilon) \cdot \log n$	$\mathrm{poly}(n, s, 1/\varepsilon)$	✓

Ehrenfeucht and Haussler had posed (as the first open problem of their paper) the question of achieving polynomial sample complexity. Such algorithms were subsequently obtained by Linial, Mansour, and Nisan [45] and Mehta and Raghavan [46]. Interestingly, these two algorithms are very different from each other and from [21]: the algorithm of [45], being Fourier-based, is non-proper, whereas the algorithm of [46], which uses dynamic programming, has a large memory footprint. Furthermore, both algorithms have a quasipolynomial dependence on $1/\varepsilon$ in their runtimes, rather than [21]'s polynomial dependence.

This state of affairs raises the question of whether there is a *single* algorithm that achieves "the best of [21], [45], and [46]" in each of the four metrics discussed above; see Table 1. We give such an algorithm in this work (Theorem 54). Our algorithm is a surprisingly simple modification of [21]'s algorithm, but our analysis is more involved. At a high level, the idea is to terminate [21]'s algorithm early to achieve our improved sample and memory complexity. However, incorporating this plan with the inherently bottom-up nature of [21]'s algorithm necessitates a delicate error analysis. (In particular, [21]'s algorithm is an Occam algorithm, whereas ours is not.)[10][11]

We remark that there is an ongoing flurry of research activity on the memory complexity of learning basic concept classes under the uniform distribution, with a specific focus on tradeoffs between memory and sample complexity [64, 65, 57, 42, 47, 58, 48, 4, 26, 27].

[10] Although our algorithm, like the others in Table 1, only uses random examples, to our knowledge there are no known membership query algorithms that achieves our guarantees.

[11] We note that it is possible to combine the ideas in [21] and [46] to give an algorithm that runs in $\mathrm{poly}(n^{\log(s/\varepsilon)})$ time and has sample and memory complexity $\mathrm{poly}(s, 1/\varepsilon) \cdot \log n$ and $\mathrm{poly}(n, s, 1/\varepsilon)$ respectively. We do not provide the details in this paper since our main result (Theorem 54) achieves strictly better guarantees.

3 Discussion and related work

3.1 Relationship to practical machine learning heuristics

Our work is motivated in part by the tremendous popularity and empirical success of top-down decision tree learning heuristics in machine learning practice, such as ID3 [55], its successor C4.5 [56], and CART [11]. The data mining textbook [67] describes C4.5 as "a landmark decision tree program that is probably the machine learning workhorse most widely used in practice to date". In a similar vein, quoting Kearns and Mansour [40], "In experimental and applied machine learning work, it is hard to exaggerate the influence of top-down heuristics for building a decision tree from labeled sample data [...] Dozens of papers describing experiments and applications involving top-down decision tree learning algorithms appear in the machine learning literature each year".

We give a high-level description of how these heuristics work, using the framework of uniform-distribution learning. As we will soon see, they serve as motivation for the heuristic that we study, BuildTopDownDT (Figure 1). These heuristics grow a bare tree T° for a function $f : \{0,1\}^n \to \{0,1\}$ as follows. Consider the progress measure

$$\mathcal{H}(T^\circ) \coloneqq \sum_{\ell \in \text{leaves}(T^\circ)} \Pr_{\boldsymbol{x} \sim \{0,1\}^n}[\,\boldsymbol{x} \text{ reaches } \ell\,] \cdot \mathscr{G}(\mathbb{E}[f_\ell]),$$

where $\mathscr{G} : [0,1] \to [0,1]$ is known as the *impurity function*, and encapsulates the splitting criterion of the heuristic. This carefully chosen function is restricted to be concave, symmetric around $\frac{1}{2}$, and to satisfy $\mathscr{G}(0) = \mathscr{G}(1) = 0$ and $\mathscr{G}(\frac{1}{2}) = 1$. For example, $\mathscr{G}$ is the binary entropy function in ID3 and C4.5; CART uses $\mathscr{G}(p) = 4p(1-p)$, known as the *Gini criterion*; [40] studies the variant $\mathscr{G}(p) = 2\sqrt{p(1-p)}$.[12] Writing $T^\circ_{\ell,i}$ to denote T° with its leaf ℓ replaced with a query to the variable x_i, these heuristics, in a single iteration, grow T° to $T^\circ_{\ell^\star,i^\star}$, where

$$(\ell^\star, i^\star) \text{ is the leaf-variable pair that maximizes } \mathcal{H}(T^\circ) - \mathcal{H}(T^\circ_{\ell^\star,i^\star}). \tag{1}$$

We refer to any such top-down heuristic as an *impurity-based* heuristic, and the progress measure $\mathcal{H}(T^\circ) - \mathcal{H}(T^\circ_{\ell^\star,i^\star})$ as the *purity gain*.

Inherent limitations of impurity-based heuristics

It is easy to see (and has been well known [38]) that impurity-based heuristics can, in general, fare very badly, in the sense of building a decision tree that is much larger than the optimal decision tree. For example, consider $f(x) = x_j \oplus x_k$ for $j, k \in [n]$, the parity of two variables. For such a target function, *regardless of the choice of the impurity function* $\mathscr{G}$, splitting on any of the n variables results in zero purity gain. This is because $\mathbb{E}[f] = \mathbb{E}[f_{x_i=b}]$ for all $i \in [n]$ and $b \in \{0,1\}$. Therefore, *any* impurity-based heuristic may build a tree of size $\Omega(2^n)$ before achieving any non-trivial error $\varepsilon < \frac{1}{2}$, whereas the size of the optimal tree of f is only 4.

One could exclude such "parity-like" examples by considering only *monotone* functions. Monotonicity is a ubiquitous condition in machine learning since many data sets are naturally monotone in their attributes. In the case of monotone functions, it can be shown that for

[12] The work of Dietterich, Kearns, and Mansour [20] gives a detailed experimental comparison of various impurity functions.

any impurity function $\mathscr{G}$, the variable split that results in the most progress in the sense of (1), i.e. the variable x_i that maximizes the purity gain

$$\mathscr{G}(\mathbb{E}[f]) - \tfrac{1}{2}(\mathscr{G}(\mathbb{E}[f_{x_i=0}]) + \mathscr{G}(\mathbb{E}[f_{x_i=1}])),$$

is precisely the most influential variable of f (we prove this in Section 7; see Proposition 41). In other words, in the case of monotone functions, BuildTopDownDT closely models impurity-based heuristics. The works of Fiat and Pechyony [23] and Lee [44] (recall our discussion following Theorem 8) were explicitly motivated by this observation, as are our results on monotone functions (Theorems 6 to 8 and 10).

As we will show, our monotone lower bounds for BuildTopDownDT actually apply to all impurity-based heuristics (Theorem 8), regardless of the choice of the impurity function $\mathscr{G}$ (hence including ID3, C4.5, and CART).[13] Since one could argue that real-world data sets are unlikely to be "parity-like", we view our monotone lower bounds as providing more robust (albeit still only theoretical) evidence of the limitations and potential shortcomings of the impurity-based top-down heuristics used in practice.

Top-down versus bottom-up: from practice to theory?

We find it especially intriguing that the algorithm of Ehrenfeucht and Haussler [21] – which as discussed, remains the fastest algorithm for properly learning decision trees with provable runtime guarantees – builds its hypothesis tree *bottom up*, in exactly the *opposite* order from the top-down heuristics used in practice. It is natural to ask if top-down heuristics can serve as inspiration for the design and analyses of fundamentally different algorithms for properly learning decision trees.

Our algorithmic upper bounds for BuildTopDownDT (Theorems 9 and 10) provide affirmative answers, and as discussed above, these new algorithms even have certain qualitative advantages over [21]. Our lower bounds (Theorems 4 and 7), on the other hand, establish their inherent limitations. They imply that BuildTopDownDT is provably *not* a polynomial-time algorithm for properly learning decision trees using membership queries, or a polynomial-time algorithm for properly learning monotone decision trees using random examples. Either of these results would constitute a major advance in learning theory, and BuildTopDownDT – and other impurity-based variants of it – had been a natural candidate for obtaining them. Indeed, the results of [44] were explicitly motivated by the goal of showing that BuildTopDownDT *is* a polynomial-time algorithm for properly learning monotone decision trees. This is now ruled out by Theorems 7 and 8.[14]

3.2 Related work

Fiat and Pechyony [23] considered linear threshold functions and read-once DNF formulas, and showed that BuildTopDownDT, when run on such functions, returns a decision tree of optimal size computing them exactly. (Stated in the notation of Theorem 3, TopDownDTSize$(f) = s$ for such functions.)

[13] Different impurity functions $\mathscr{G}$ lead to different *orderings* of leaves to split, and hence result in different trees.

[14] Blum et al. [7] gave an information-theoretic lower bound showing that no "statistical query" algorithm can learn decision trees in polynomial time. However, this lower bound does not apply when membership queries are allowed or when the function is assumed to be monotone.

Kearns and Mansour [40] (see also [38, 20]) showed that impurity-based heuristics are *boosting algorithms*, where one views the functions labeling internal nodes of the tree (single variables in our case) as weak learners. At a high level, the proofs of our upper bounds (Theorems 3 and 6) are similar in spirit to their analysis, in the sense that they are all incremental in nature, showing that each split contributes to the accuracy of the decision tree hypothesis. However, our results and analyses are incomparable – for example, [40] does not relate the size of the resulting hypothesis to the size of the optimal decision tree; [40]'s analysis assumes the existence of weak learners for all filtered-and-rebalanced versions of the target distribution, whereas we carry out the entirety of our analyses with respect to the uniform distribution.[15]

Recent work of Brutzkus, Daniely, and Malach [13] studies a variant of ID3 proposed by [40], focusing on learning conjunctions and read-once DNF formulas under product distributions. They provide theoretical and empirical evidence showing that for such functions, the size-t tree grown by [40]'s variant of ID3 achieves optimal or near-optimal error among all trees of size t. Concurrent work by the same authors [12] shows that ID3 efficiently learns $(\log n)$-juntas in the setting of smoothed analysis.

4 Preliminaries

Throughout this paper, we use bold font (e.g. $\boldsymbol{x}$ and $\boldsymbol{S}$) to denote random variables; all probabilities and expectations are with respect to the uniform distribution unless otherwise stated.

For any decision tree T, we say the *size* of T is the number of leaves in T, and the *depth* of T is length of the longest path between the root and a leaf. If a tree has size 1, then it contains a single leaf, computes either the constant $+1$ or constant -1 function, and has depth 0. For a function $f : \{0,1\}^n \to \{\pm 1\}$, the *optimal decision tree size* of f is the smallest s for which there exists a decision tree of size s that exactly computes f, and we write $\mathrm{size}(f)$ to denote this quantity. If T is a decision tree that computes f, then we will often use T interchangeably with f.

Choose any $f, g : \{0,1\}^n \to \{\pm 1\}$. Then, the *error* is defined as

$$\mathrm{error}(f,g) = \Pr_{\boldsymbol{x} \sim \{0,1\}^n}[f(\boldsymbol{x}) \neq g(\boldsymbol{x})].$$

We say that f is an ε-approximation of g if $\mathrm{error}(f,g) \leq \varepsilon$. If T° is a bare tree, then $\mathrm{error}(T^\circ, f)$ is shorthand for $\mathrm{error}(T, f)$ where T is the f-completion of T°. We also use the following shorthand.

$$\mathrm{error}(f, \pm 1) = \min(\mathrm{error}(f,-1), \mathrm{error}(f,1)).$$

The *variance* of $f : \{0,1\}^n \to \{\pm 1\}$, denoted $\mathrm{Var}(f)$, is

$$\mathrm{Var}(f) = 4 \cdot \Pr[f(\boldsymbol{x}) = -1] \cdot \Pr[f(\boldsymbol{x}) = 1].$$

The *total influence* of f, denoted $\mathrm{Inf}(f)$, is

$$\mathrm{Inf}(f) = \sum_{i=1}^{n} \mathrm{Inf}_i(f).$$

[15] Indeed, [40]'s results concern impurity-based heuristics, and as discussed above, statements like Theorem 3 that apply to all functions cannot hold for such heuristics because of parity-like functions.

It is easy to see that for any decision tree $T : \{0,1\}^n \to \{\pm 1\}$,

$$\text{error}(T, \pm 1) \leq \text{Inf}(T)$$

and

$$\frac{\text{Var}(T)}{2} \leq \text{error}(T, \pm 1) \leq \text{Var}(T)$$

always hold.

5 Upper bounds on TopDownDTsize: Proofs of Theorems 3 and 6

Recall that BuildTopDownDT(f, ε) continually grows a bare tree, T°, until the f-completion of T° is an ε-approximation of f. At a high level, the proofs of our upper bounds on TopDownDTSize proceed as follows.

Section 5.1 We define a progress metric, the "cost" of T°, which upper bounds the error of the f-completion of T° with respect to f. Hence, when the "cost" drops below ε, BuildTopDownDT can terminate. We show that whenever BuildTopDownDT grows T°, the "cost" of T° decreases by exactly the score of the leaf selected.

Section 5.2 We lower bound the score of the leaf that BuildTopDownDT selects.

Section 5.3 We put the above together to prove upper bounds on TopDownDTSize. At each step, the "cost" of T° must decrease by at least the lower bounds in Section 5.2, which allows us to upper bound the number of steps until the "cost" falls below ε. This is sufficient since the size of the tree that BuildTopDownDT produces is exactly one more than the number of steps it takes.

5.1 Definition and properties of "Cost"

▶ **Definition 12** (Cost of a bare tree). *Let $f : \{0,1\}^n \to \{\pm 1\}$ be a function and T° be a bare tree. Then the* cost of T° relative to f *is defined as*

$$\text{cost}_f(T^\circ) = \sum_{\text{leaf } \ell \in T^\circ} 2^{-|\ell|} \cdot \text{Inf}(f_\ell).$$

This cost function is useful to track because it naturally decreases during BuildTopDownDT and upper bounds the error of the completion.

▶ **Lemma 13** (Properties of cost of a bare tree). *For any $f : \{0,1\}^n \to \{\pm 1\}$ and bare tree T°, the following hold:*

1. $\text{error}(T^\circ, f) \leq \text{cost}_f(T^\circ)$.
2. *Choose any leaf ℓ of T° and variable x_i. Let $(T^\circ)'$ be the bare tree that results from replacing ℓ in T° with a query to x_i. Then,*

$$\text{cost}_f((T^\circ)') = \text{cost}_f(T^\circ) - 2^{-|\ell|} \cdot \text{Inf}_i(f_\ell).$$

At each step, BuildTopDownDT splits the leaf with the largest score, resulting in the cost decreasing by exactly the score selected. Once the cost decreases to below ε, we know the completion of T° is an ε-approximation of f, meaning BuildTopDownDT can terminate.

Proof. The proof of (1) is a simple application of the fact that $\text{error}(g, \pm 1) \leq \text{Inf}(g)$ for any boolean function g:

$$\begin{aligned}\text{error}(T^\circ, f) &= \Pr_{\boldsymbol{x}\sim\{0,1\}^n}[(\text{Completion of } T^\circ)(\boldsymbol{x}) \neq f(\boldsymbol{x})] \\ &= \sum_{\text{leaf } \ell\in T^\circ} \Pr_{\boldsymbol{x}\sim\{0,1\}^n}[\boldsymbol{x} \text{ reaches } \ell] \cdot \text{error}(f_\ell, , \pm 1) \\ &\leq \sum_{\text{leaf } \ell\in T^\circ} 2^{-|\ell|} \cdot \text{Inf}_i(f_\ell) = \text{cost}_f(T^\circ).\end{aligned}$$

The proof of (2) follows from the fact that if T is a tree with x_i at the root, T_0 as its 0-subtree, and T_1 as its 1-subtree, then $\text{Inf}(T) - \text{Inf}_i(T) = \frac{1}{2}(\text{Inf}(T_0) + \text{Inf}(T_1))$. This fact is true because

$$\begin{aligned}\text{Inf}(T) - \text{Inf}_i(T) &= \sum_{j\neq i} \text{Inf}_j(T) \\ &= \sum_{j\neq i} \frac{1}{2}\text{Inf}_j(T_0) + \frac{1}{2}\text{Inf}_j(T_1) \\ &= \frac{1}{2}\left(\sum_{j=1}^{n} \text{Inf}_j(T_0) + \sum_{j=1}^{n} \text{Inf}_j(T_1)\right) \\ &= \frac{1}{2}(\text{Inf}(T_0) + \text{Inf}(T_1)).\end{aligned}$$

◀

5.2 Lower bounds on the score of the leaf BuildTopDownDT selects

We give two different lower bounds. These lower bounds are incomparable, so when proving Theorems 3 and 6, we use whichever is better. Both of these lower bounds rely on a powerful inequality from the analysis of boolean functions due to O'Donnell, Saks, Schramm, and Servedio [52], which we restate in the form most convenient for us.

▶ **Theorem 14** (Corollary of Theorem 1.1 from [52]). *Let f be a size-s decision tree. Then,*

$$\max_i \left(\text{Inf}_i(f)\right) \geq \frac{\text{Var}(f)}{\log s}.$$

We prove our first lower bound on the score of the leaf selected.

▶ **Lemma 15.** *Let f be a size s decision tree. At step j,* BuildTopDownDT(f, ε) *selects a leaf, ℓ^* with score at least*

$$\text{score}(\ell^*) \geq \frac{\varepsilon}{(j+1)\log(s)}.$$

Proof. If BuildTopDownDT has not terminated at step j, then, the completion of T° is not an ε-approximation of f. Equivalently,

$$\sum_{\text{leaf } \ell\in T^\circ} 2^{-|\ell|} \cdot \text{error}(f_\ell, \pm 1) > \varepsilon$$

At step j, there are exactly $j+1$ leaves in T°, so there must be at least one leaf, ℓ, where

$$2^{-|\ell|} \cdot \text{error}(f_\ell, \pm 1) > \frac{\varepsilon}{j+1}.$$

Since $\mathrm{Var}(f_\ell) \geq \mathrm{error}(f_\ell, \pm 1)$, we also know

$$2^{-|\ell|} \cdot \mathrm{Var}(f_\ell) > \frac{\varepsilon}{j+1}.$$

By Theorem 14, we know, for some variable x_i, $\mathrm{Inf}_i(f_\ell) \geq \mathrm{Var}(f_\ell)/\log(\mathrm{size}(f_\ell))$. The optimal size of any restriction of f is certainly at most the optimal size of f itself, so

$$2^{-|\ell|} \cdot \mathrm{Inf}_i(f_\ell) > \frac{\varepsilon}{(j+1)\log(s)}.$$

Since BuildTopDownDT picks a leaf with maximum score, and ℓ has a score at least $\varepsilon/(j+1)\log(s)$, it must pick a leaf with at least that score. ◀

A standard fact from the analysis of boolean functions gives a $\log s$ upper bound on the total influence of a size-s decision tree (see e.g. [53]). In order to prove a second lower bound on the score of the leaf that BuildTopDownDT selects, we will need a refinement of this bound that takes into account the variance of the function. The following lemma is a slight variant of a related (though incomparable) result in [6], which upper bounds the total influence of an s-term DNF formula by $2\mu\log(s/\mu)$, where $\mu \coloneqq \Pr[f(\boldsymbol{x}) = 1]$.

▶ **Lemma 16** (Total influence of size-s DTs). *Let $f : \{0,1\}^n \to \{\pm 1\}$ be computed by a size-s decision tree T. Then*

$$\mathrm{Inf}(f) \leq \mathrm{Var}(f)\log(4s/\mathrm{Var}(f)).$$

Proof. We may assume without loss of generality that $\mu \coloneqq \Pr[f(\boldsymbol{x}) = 1] \leq \frac{1}{2}$, since $\mathrm{Inf}(f) = \mathrm{Inf}(\neg f)$ and if f is a size-s decision tree then so is its negation $\neg f$. Since

$$\begin{aligned}
\mathrm{Inf}(f) &= \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\{0,1\}^n}[\mathrm{sens}_f(\boldsymbol{x})] && (\text{where } \mathrm{sens}_f(x) \coloneqq |\{i \in [n] \colon f(x) \neq f(x^{\oplus i})\}|)\\
&= 2 \cdot \mathbb{E}\left[\mathrm{sens}_f(\boldsymbol{x})\mathbb{1}[f(\boldsymbol{x}) = 1]\right]\\
&\leq 2 \sum_{\text{1-leaves } \ell \in T} 2^{-|\ell|} \cdot |\ell| && (\mathrm{sens}_f(x) \leq |\ell| \text{ for every } x \text{ that reaches } \ell)\\
&\leq 2\mu\log(s/\mu) && (\text{Concavity of } t \mapsto t\log(1/t), \text{ and } \mathrm{size}(T) \leq s)\\
&\leq \mathrm{Var}(f)\log(4s/\mathrm{Var}(f)), && (\mathrm{Var}(f) = 4\mu(1-\mu), \text{ and our assumption that } \mu \leq \tfrac{1}{2})
\end{aligned}$$

the lemma follows. ◀

We now provide a second lower bound on the score of the leaf BuildTopDownDT selects. The lower bound provided below in Lemma 17 is better than the bound provided by Lemma 15 when $\mathrm{cost}_f(T^\circ)$ is large.

▶ **Lemma 17.** *Let f be a size s decision tree. Suppose that* BuildTopDownDT(f, ε) *has already constructed the bare tree T° at step j and that $\mathrm{cost}_f(T^\circ) \geq \varepsilon\log(4s/\varepsilon)$. Then, the next leaf, ℓ^*, that* BuildTopDownDT *picks has score at least*

$$\mathrm{score}(\ell^*) \geq \frac{\mathrm{cost}_f(T^\circ)}{(j+1)\log(4s/\varepsilon)\log(s)}.$$

Proof. We will show that when $\mathrm{cost}_f(T^\circ)$ is large, there is some leaf with high total influence, which means it must have high variance, and finally a variable with high influence.

We define:

$$h_s : [0,1] \to \mathbb{R} \text{ where } h_s(t) = t\log\left(\frac{4s}{t}\right) \text{ and } h_s(0) = 0.$$

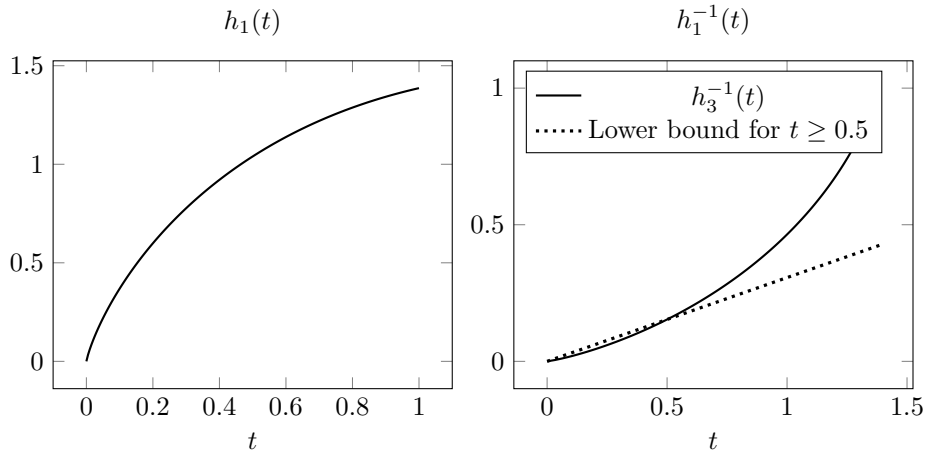


Figure 2 Graphs of the function $h_1(t) = t \cdot \log(\frac{4}{t})$ on the left, and of its inverse, $h_1^{-1}(t)$ on the right. Since the inverse is convex, we can use a linear lower bound as the dotted line in the right plot shows.

Then, for any tree T of size at most s, we have that

$$\mathrm{Inf}(T) \leq h_s(\mathrm{Var}(T)).$$

As long as $s \geq 1$, h_s is an increasing concave function. This means it has a convex inverse, h_s^{-1}, and that for any tree T of size at most s, the following lower bounds the variance.

$$\mathrm{Var}(T) \geq h_s^{-1}(\mathrm{Inf}(T)). \tag{2}$$

Since h_s^{-1} is convex and $h_s^{-1}(0) = 0$, we can lower bound it as follows. Choose arbitrary $a \in \mathbb{R}$. Then, for $t \geq a$ we have that $h_s^{-1}(t) \geq t \cdot \frac{h_s^{-1}(a)}{a}$. Choosing $a = \varepsilon \log(4s/\varepsilon)$, we have that,

$$h_s^{-1}(t) \geq \frac{t}{\log(4s/\varepsilon)} \text{ for all } t \geq \varepsilon \log\left(\frac{4s}{\varepsilon}\right).$$

Consider the bare tree, T°, at step j. By definition, it has cost

$$\sum_{\text{leaf } \ell \in T^\circ} 2^{-|\ell|} \cdot \mathrm{Inf}(f_\ell) = \mathrm{cost}_f(T^\circ).$$

We next apply Jensen's inequality.

$$\sum_{\text{leaf } \ell \in T^\circ} 2^{-|\ell|} \cdot h_s^{-1}(\mathrm{Inf}(f_\ell)) \geq h_s^{-1}(\mathrm{cost}_f(T^\circ)).$$

Since, at step j, there are $j+1$ leaves in T°, for at least one of the leaves, ℓ,

$$2^{-|\ell|} \cdot h_s^{-1}(\mathrm{Inf}(f_\ell)) \geq \frac{h_s^{-1}(\mathrm{cost}_f(T^\circ))}{j+1} \geq \frac{\mathrm{cost}_f(T^\circ)}{(j+1)\log(\frac{4s}{\varepsilon})}.$$

By Equation (2), we can lower bound the variance of f_ℓ:

$$2^{-|\ell^*|} \cdot \mathrm{Var}(f_\ell) \geq 2^{-|\ell|} \cdot h_s^{-1}(\mathrm{Inf}(f_\ell)) \geq \frac{\mathrm{cost}_f(T^\circ)}{(j+1)\log(\frac{4s}{\varepsilon})}.$$

Then, using Theorem 14 and the fact that if f is exactly computed by a size s tree, then f_ℓ is exactly computed by a tree of size at most s.

$$2^{-|\ell|} \cdot \max_i \left(\mathrm{Inf}_i[f_\ell]\right) \geq \frac{\mathrm{cost}_f(T^\circ)}{(j+1)\log(\frac{4s}{\varepsilon})\log(s)}.$$

Recall that BuildTopDownDT picks the leaf with largest score, so it will pick a leaf with score at least $\frac{\mathrm{cost}_f(T^\circ)}{(j+1)\log(4s/\varepsilon)\log(s)}$. ◀

5.3 Proofs of Theorems 3 and 6

Armed with the above Lemmas, we are now ready to prove our upper bounds on the size of the tree that BuildTopDownDT produces.

▶ **Theorem 3** (Upper bound for approximate representation). *For every $\varepsilon \in (0, \frac{1}{2})$ and every size-s decision tree f, we have* $\textsc{TopDownDTSize}(f, \varepsilon) \leq s^{O(\log(s/\varepsilon)\log(1/\varepsilon))}$.

Proof. We use C_j to refer to $\mathrm{cost}_f(T^\circ)$ after j steps of BuildTopDownDT. The size of the tree returned is one more than the number of steps BuildTopDownDT takes. Furthermore, if $C_j \leq \varepsilon$, then T° has error at most ε at step j, so BuildTopDownDT will return a tree of size at most $j+1$.

Our analysis proceeds in two phases:

Phase 1: We will show that the larger C_j is, the faster it must decrease at each step. This multiplicative reduction of C_j will allow us to conclude that after at most $k = s^{\log(4s/\varepsilon)\log(1/\varepsilon)}$ steps, that $C_k \leq \varepsilon\log(\frac{4s}{\varepsilon})$.

Phase 2: We will argue that C_j makes additive progress towards 0 once it is less than $\varepsilon\log(\frac{4s}{\varepsilon})$, showing that after $m = s^{2\log(4s/\varepsilon)\log(1/\varepsilon)}$ steps, that $C_m \leq \varepsilon$.

Once $C_m \leq \varepsilon$, the algorithm must terminate.

Phase 1: Based on Lemma 17, we know that during phase 1, BuildTopDownDT will select a leaf with influence at least $\frac{\mathrm{cost}_f(T^\circ)}{(j+1)\log(4s/\varepsilon)\log s}$ at each step j. From Lemma 13, we know that:

$$\begin{aligned} C_j &\leq C_{j-1} - \frac{C_j}{j\log(4s/\varepsilon)\log s} \\ &= C_{j-1} \cdot \left(1 - \frac{1}{j\log(4s/\varepsilon)\log s}\right). \end{aligned}$$

We can use this to bound C_k, the cost after some (k) number of steps, in terms of C_0.

$$\begin{aligned} C_k &\leq C_0 \prod_{j=1}^{k} \left(1 - \frac{1}{j\log(4s/\varepsilon)\log s}\right) \\ &= C_0 \exp\left(\sum_{j=1}^{k} \log\left(1 - \frac{1}{j\log(4s/\varepsilon)\log s}\right)\right). \end{aligned}$$

Using the fact that $\log(1+t) < t$,

$$\begin{aligned} C_k &\leq C_0 \exp\left(-\sum_{j=1}^{k} \frac{1}{j\log(4s/\varepsilon)\log s}\right) \\ &\leq C_0 \exp\left(-\frac{\log k}{\log(4s/\varepsilon)\log s}\right). \end{aligned}$$

We know that $C_0 \leq \log s$ because a size-s decision tree has total influence at most $\log s$ (see e.g. [53]). Choosing

$$k = \exp\big(\log(4s/\varepsilon)\log(s)\log(1/\varepsilon)\big) = s^{\log(4s/\varepsilon)\log(1/\varepsilon)}$$

it must be true that $C_k \leq \varepsilon \log(4s/\varepsilon)$.

Phase 2: This phase combines Lemmas 13 and 15, which together imply that

$$C_{j+1} \leq C_j - \frac{\varepsilon}{(j+1)\log s}.$$

This means that, for $m > k$,

$$C_k - C_m \geq \sum_{j=k+1}^{m} \frac{\varepsilon}{(j+1)\log s} \geq \frac{\varepsilon}{\log s}(\log m - \log k).$$

We are guaranteed to terminate at the first j such that $C_j \leq \varepsilon$, or earlier. Choosing

$$\log m = \frac{\log s}{\varepsilon} C_k + \log k$$

ensures that $C_m \leq 0$, which means BuildTopDownDT must terminate before step m. Plugging in $C_k \leq \varepsilon \log(4s/\varepsilon)$ and $k = s^{\log(4s/\varepsilon)\log(1/\varepsilon)}$ gives that

$$m \leq s^{2\log(4s/\varepsilon)\log(1/\varepsilon)}.$$

Since BuildTopDownDT terminates after at most m steps, it returns a tree of size at most $m + 1$. ◀

The proof of Theorem 6 is mostly the same as Phase 2 from the proof of Theorem 3, except we have a better guarantee on the starting cost. We will use the following upper bound on the total influence of monotone decision trees, due to O'Donnell and Servedio [53]:

▶ **Theorem 18** ([53]). *Let f be a size-s monotone decision tree. Then* $\mathrm{Inf}(f) \leq \sqrt{\log s}$.

▶ **Theorem 6** (Upper bound for approximate representation of monotone functions.). *For every $\varepsilon \in (0, \frac{1}{2})$ and every size-s monotone decision tree f, we have* $\textsc{TopDownDTSize}(f, \varepsilon) \leq s^{O(\sqrt{\log s}/\varepsilon)}$.

Proof. We use C_j to refer to $\mathrm{cost}_f(T^\circ)$ after j steps of BuildTopDownDT. By combining Lemma 15 and Lemma 13, we know that

$$C_{j+1} \leq C_j - \frac{\varepsilon}{(j+1)\log s}.$$

At any step k,

$$C_0 - C_k \geq \sum_{j=0}^{k-1} \frac{\varepsilon}{(j+1)\log s} \geq \frac{\varepsilon \cdot \log k}{\log s}.$$

Since f is a monotone decision tree of size s, it has total influence at most $\sqrt{\log s}$ (Theorem 18). This means that $C_0 \leq \sqrt{\log s}$. We choose

$$k = \exp(\log(s)^{1.5}/\varepsilon) = s^{\sqrt{\log s}/\varepsilon}$$

at which point, $C_k \leq 0 \leq \varepsilon$, so BuildTopDownDTreturns a tree of size $k + 1$. ◀

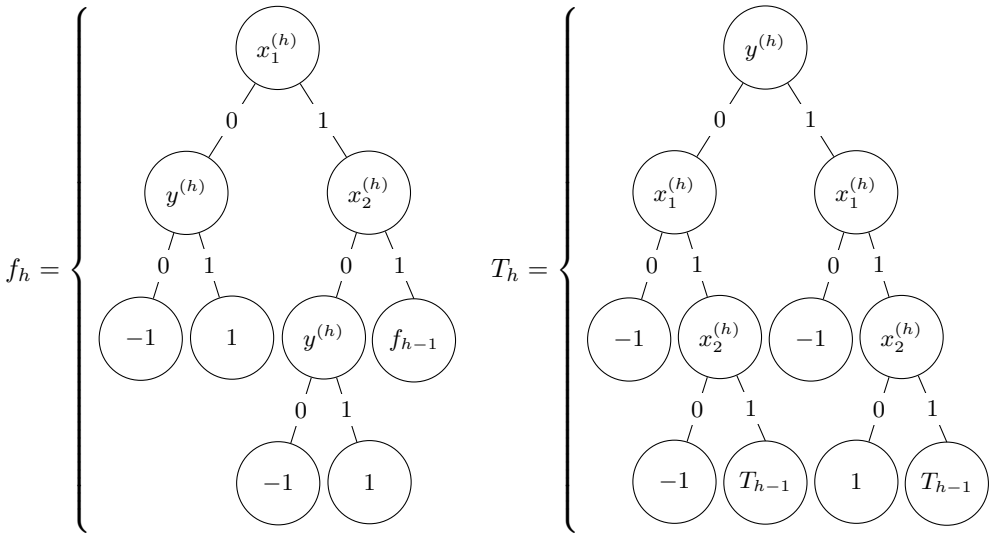


Figure 3 Diagrams exhibiting a function with exponential difference between the optimal decision tree size and TopDownDTSize. The left diagram shows how to compute f_h with a decision tree of size $O(h)$. The right diagram shows T_h, the tree BuildTopDownDT builds, which has size $2^{\Omega(h)}$.

6 Lower bounds on TopDownDTsize for general functions: Proof of Theorem 4

6.1 Size separation for exact representation: Proof of Theorem 4(a)

We begin with a simple family of functions $\{f_h\}_{h\in\mathbb{N}}$ whose BuildTopDownDT tree has exponential size compared to the optimal tree. Each f_h is a function over $3h+1$ boolean variables $x_1^{(1)}, x_2^{(1)}, \ldots, x_1^{(h)}, x_2^{(h)}, y^{(1)}, \ldots, y^{(h)}, z$, and is defined inductively as follows:

$$f_0(z) = z,$$

and for $h \geq 1$,

$$f_h(x,y,z) = \begin{cases} y^{(h)} & \text{if } x_1^{(h)} \vee x_2^{(h)} \\ f_{h-1}(x,y,z) & \text{otherwise.} \end{cases}$$

The structure of BuildTopDownDT(f_h)

We see that $y^{(h)}$ has influence $\frac{3}{4}$, both $x_1^{(h)}$ and $x_2^{(h)}$ have influence $\frac{1}{4}$, and each variable in f_{h-1} has influence $< \frac{1}{4}$. BuildTopDownDT(f_h) therefore queries y_k at the root. In the restrictions of f_h obtained by setting $y^{(h)}$ to a constant, $x_1^{(h)}$ and $x_2^{(h)}$ have equal influence of $\frac{1}{4}$ and each variable in f_{h-1} has influence $< \frac{1}{4}$. By setting either $x_1^{(h)}$ or $x_2^{(h)}$ to a constant, we get a subfunction where the other $x^{(h)}$-variable has influence $\frac{1}{2}$ and each node in f_{h-1} has influence $< \frac{1}{2}$. Thus, BuildTopDownDT(f_h) builds the tree T_h depicted in Figure 3.

We see that each T_h contains two copies of T_{h-1}. It follows that the optimal size of f_h is $O(h)$, whereas the size of T_h is $2^{\Omega(h)}$: a size separation of TopDownDTSize$(f_h) = 2^{\Omega(s)}$ where s denotes the optimal size of f_h.

6.2 Size separation for approximate representation: Proof of Theorem 4(b)

Warmup/intuition: An s versus $s^{\Omega(\log(1/\varepsilon))}$ separation

Before proving Theorem 4(b), we first give a brief, informal description of how a simple modification to the family of functions $\{f_h\}_{h\in\mathbb{N}}$ in Theorem 4(a) above yields a separation of $\textsc{TopDownDTSize}(f,\varepsilon) = s^{\Omega(\log(1/\varepsilon))}$ for approximate representation. Theorem 4(b) – which improves this to a superpolynomial separation even for constant ε – builds on these ideas, but the family of functions and the proof of the lower bound are significantly more involved.

Consider replacing each $y^{(h)}$ variable in the definition of f_h with the parity of k variables $y_1^{(h)} \oplus \cdots \oplus y_k^{(h)}$, i.e. consider the following variant $\tilde{f}_h$ of f_h:

$$\tilde{f}_h(x,y,z) = \begin{cases} y_1^{(h)} \oplus \cdots \oplus y_k^{(h)} & \text{if } x_1^{(h)} \vee x_2^{(h)} \\ \tilde{f}_{h-1}(x,y,z) & \text{otherwise.} \end{cases}$$

Just like the single $y^{(h)}$ variable in f_h, we see that the k many $y_i^{(h)}$ variables are the most influential in $\tilde{f}_h$ (each having influence $\frac{3}{4}$). Furthermore, each $y_i^{(h)}$ variable remains the most influential even under *any* restriction to *any* number of the other $y_j^{(h)}$ variables. Therefore the tree $\tilde{T}_h$ that BuildTopDownDT builds for $\tilde{f}_h$ first queries all k many $y^{(h)}$ variables. At each of the 2^k resulting leaves, $x_1^{(h)}$ and $x_2^{(h)}$ are then queried, followed by a copy of $\tilde{T}_{h-1}$, the tree that BuildTopDownDT recursively constructs for $\tilde{f}_h$, in the branch corresponding to $x_1^{(h)} = x_2^{(h)} = 1$. The fact that there are 2^k copies of $\tilde{T}_{h-1}$ within $\tilde{T}_h$ should be contrasted with the fact that the tree T_h in Theorem 4(a) contains just two copies of T_{h-1}; recall Figure 3.

It is straightforward to see that there is a tree of size $O(h \cdot 2^k)$ that computes $\tilde{f}_h$. This tree is built by first querying the $x^{(h)}$ variables before the $y^{(h)}$ variables, and recursing on just one of the $\Omega(2^k)$ many resulting leaves. On the other hand, by first querying the $y^{(h)}$ variables followed by the $x^{(h)}$ variables, BuildTopDownDT recurses on $\Omega(2^k)$ many branches while only correctly classifying a $\frac{3}{4}$ fraction of inputs. Choosing $h = \Theta(\log(1/\varepsilon))$, we get a separation of $O(h \cdot 2^k)$ versus $2^{\Omega(kh)}$, or equivalently, s versus $s^{\Omega(\log(1/\varepsilon))}$.

6.2.1 Proof of Theorem 4(b)

Before defining the family of functions witnessing the separation, we define a couple of basic boolean functions and state a few of their properties that will be useful for our analyses:

▶ **Definition 19** (Tribes). *For any input length r, let w be the largest integer such that $(1-2^{-w})^{r/w} \leq \frac{1}{2}$. The $\textsc{Tribes}_r : \{0,1\}^r \to \{\pm 1\}$ function is defined to be the function computed by the read-once DNF with $\lfloor \frac{r}{w} \rfloor$ terms (over disjoint sets of variables) of width exactly w:*

$$\textsc{Tribes}_r(z) = (z_{1,1} \wedge \cdots \wedge z_{1,w}) \vee \cdots \vee (z_{t,1} \wedge \cdots \wedge z_{t,w}) \quad \text{where } t := \lfloor \tfrac{r}{w} \rfloor,$$

and where we adopt the convention that -1 represents logical False *and 1 represents logical* True.

The following facts about the Tribes function are standard (see Chapter §4.2 of [51]) and can be easily verified:

▶ **Fact 20** (Properties of $\textsc{Tribes}_r$).

- $\Pr[\textsc{Tribes}_r(\boldsymbol{z}) = 1] = \frac{1}{2} - O\big(\frac{\log r}{r}\big)$.
- $\mathrm{Inf}(\textsc{Tribes}_r) = (1 \pm o(1)) \cdot \ln r$ *and consequently,* $\mathrm{Inf}_i(\textsc{Tribes}_r) = (1 \pm o(1)) \cdot \frac{\ln r}{r}$ *for all* $i \in [n]$.
- $w = \log r - \log \ln r \pm O(1)$.
- $\mathrm{size}(\textsc{Tribes}_r) \leq w^{O(r/w)} = 2^{O(r \log \log r / \log r)}$.

▶ **Definition 21** (Threshold). *For any input length* ℓ *and* $t \in \{0, \ldots, \ell\}$*, the* $\textsc{Threshold}_{\ell,t} : \{0,1\}^\ell \to \{\pm 1\}$ *function is defined to be*

$$\textsc{Threshold}_{\ell,t}(x) = 1 \iff \sum_{i=1}^{\ell} x_i \leq t.$$

Defining the family of functions witnessing the separation

Consider the following family of functions $\{f_h\}_{h \in \mathbb{N}}$. Each f_h is a function over $h(\ell + k) + r$ boolean variables $x^{(1)}, x^{(2)}, \ldots, x^{(h)} \in \{0,1\}^\ell$,$y^{(1)}, \ldots, y^{(h)} \in \{0,1\}^k$, and $z \in \{0,1\}^r$, and is defined inductively as follows:

$$f_0(z) = \textsc{Tribes}_r(z),$$

and for $h \geq 1$,

$$f_h(x, y, z) = \begin{cases} \textsc{Parity}_k(y^{(h)}) & \text{if } \textsc{Threshold}_{\ell,1}(x^{(h)}) = 1 \\ f_{h-1}(x, y, z) & \text{otherwise.} \end{cases}$$

▷ Claim 22 (Optimal decision tree size of f_h).

$$\begin{aligned} \mathrm{size}(f_h) &\leq \ell^{O(h)} \cdot (\mathrm{size}(\textsc{Parity}_k) + \mathrm{size}(\textsc{Tribes}_r)) \\ &\leq \ell^{O(h)} \cdot (2^k + 2^{O(r \log \log r / \log r)}). \end{aligned}$$

Proof. Please refer to figure Figure 4. We first build a tree of size $O(\ell^2)$ that evaluates $\textsc{Threshold}_{\ell,1}(x^{(h)})$. Of these leaves, $\ell + 1$ descend into a tree computing $\textsc{Parity}_k(y^{(h)})$, which has size 2^k. The others descend into a tree computing f_{h-1}. This yields the recurrence

$$\begin{aligned} \mathrm{size}(f_h) &\leq O(\ell) \cdot \mathrm{size}(\textsc{Parity}_k) + O(\ell^2) \cdot \mathrm{size}(f_{h-1}) \\ &\leq O(\ell \cdot 2^k) + O(\ell^2) \cdot \mathrm{size}(f_{h-1}) \\ \mathrm{size}(f_0) &= \mathrm{size}(\textsc{Tribes}_r) \leq 2^{O(r \log \log r / \log r)}, \qquad \text{(Recall Fact 20)} \end{aligned}$$

and the claim follows. ◁

In the remainder of this section, we will prove a lower bound on $\textsc{TopDownDTSize}(f, \varepsilon)$. Figure 5 should be contrasted with Figure 4.

The structure of BuildTopDownDT(f_h)

The following helper lemma will be useful in determining the structure of the tree BuildTopDownDT produces.

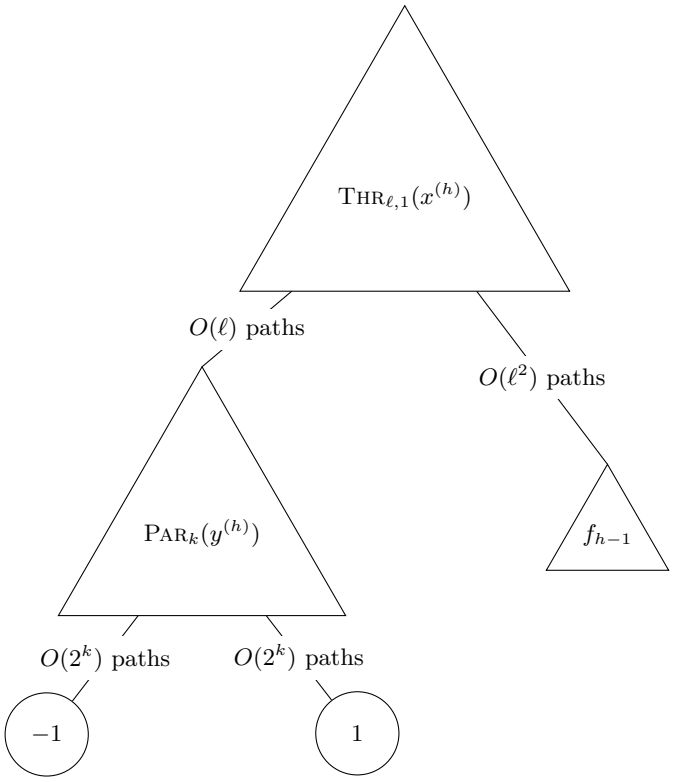


Figure 4 A small decision tree for f_h.

▶ **Lemma 23** (Preservation of influence order). *Let* $f : \{0,1\}^{\overline{S}} \times \{0,1\}^S \to \{\pm 1\}$ *and* $\tilde{f} : \{0,1\}^S \to \{\pm 1\}$ *be two functions satisfying the following: there is a function* $g : \{0,1\}^{\overline{S}} \times \{\pm 1\} \to \{\pm 1\}$ *such that:*

$$f(a,b) = g(a, \tilde{f}(b)) \quad \text{for all } a \in \{0,1\}^{\overline{S}},\ b \in \{0,1\}^S.$$

Then for all variables $v_1, v_2 \in S$,

$$\text{Inf}_{v_1}(\tilde{f}) \geq \text{Inf}_{v_2}(\tilde{f}) \quad \text{if and only if} \quad \text{Inf}_{v_1}(f) \geq \text{Inf}_{v_2}(f).$$

Proof. This holds by noting that for $v \in \{v_1, v_2\}$,

$$\begin{aligned}
\text{Inf}_v(f) &= \Pr_{\boldsymbol{a},\boldsymbol{b}}[f(\boldsymbol{a},\boldsymbol{b}) \neq f(\boldsymbol{a},\boldsymbol{b}^{\oplus v})] \\
&= \Pr_{\boldsymbol{a},\boldsymbol{b}}[g(\boldsymbol{a}, \tilde{f}(\boldsymbol{b})) \neq g(\boldsymbol{a}, \tilde{f}(\boldsymbol{b}^{\oplus v}))] \\
&= \Pr_{\boldsymbol{b}}[\tilde{f}(\boldsymbol{b}) \neq \tilde{f}(\boldsymbol{b}^{\oplus v})] \cdot \Pr_{\boldsymbol{a}}[g(\boldsymbol{a},-1) \neq g(\boldsymbol{a},1)] \\
&= \text{Inf}_v(\tilde{f}) \cdot \Pr_{\boldsymbol{a}}[g(\boldsymbol{a},-1) \neq g(\boldsymbol{a},1)].
\end{aligned}$$

The lemma follows since $\Pr_{\boldsymbol{a}}[g(\boldsymbol{a},-1) \neq g(\boldsymbol{a},1)]$ does not depend on v (and hence is the same regardless of whether $v = v_1$ or $v = v_2$). ◀

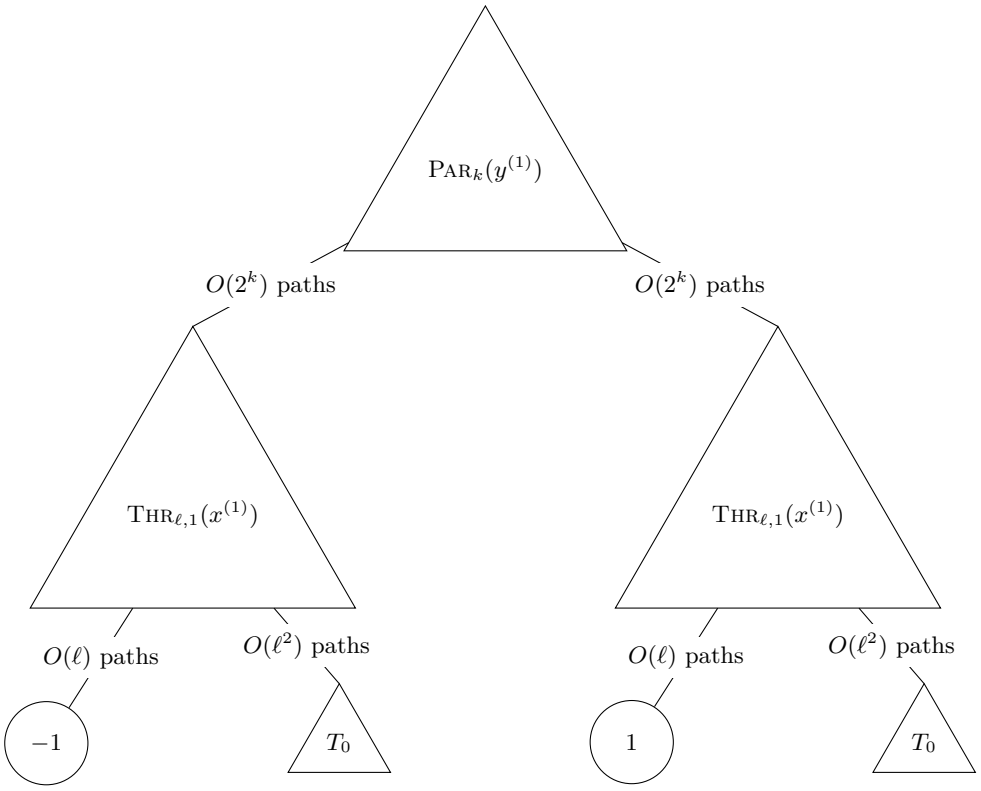


Figure 5 The tree T_1 that BuildTopDownDT builds for f_1. Since $y^{(1)}$ has all the most influential variables, BuildTopDownDT puts them all at the root. As a result, it ends up building a significantly larger tree than optimal (cf. Figure 4). Notice that the size of T_1 is $\Omega(2^k)$ times as large as T_0. This leads to exponential growth of the tree size as a function of h.

Lemma 23 is especially well-suited for our inductively-defined family of functions $\{f_h\}_{h\in\mathbb{N}}$. For each $i \in \{0, 1, \ldots, h\}$, we let S_i denote the relevant variables of f_i. Therefore

$$S_0 = \{z_1, \ldots, z_r\}$$
$$S_{i+1} = S_i \sqcup \{x_1^{(i)}, \ldots, x_\ell^{(i)}, y_1^{(i)}, \ldots, y_k^{(i)}\}$$

▶ **Observation 24.** *For all $i \in \{0, 1, \ldots, h\}$, there exists g_i such that*

$$f_h(a, b) = g_i(a, h_i(b)) \quad \text{for all } a \in \{0,1\}^{S_h \setminus S_i} \text{ and } b \in \{0,1\}^{S_i}. \tag{3}$$

Consequently, we may apply Lemma 23 to get that for all $v_1, v_2 \in S_i$, we have that

$$\mathrm{Inf}_{v_1}(f_i) \geq \mathrm{Inf}_{v_2}(f_i) \quad \text{if and only if} \quad \mathrm{Inf}_{v_1}(f_h) \geq \mathrm{Inf}_{v_2}(f_h).$$

We note the following corollary, which is a straightforward consequence of the observation that the property (3) is preserved under restrictions:

▶ **Corollary 25** (Preservation of influence order under restrictions)**.** *Let π be any restriction. For all $i \in \{0, 1, \ldots, h\}$, we have that*

$$\mathrm{Inf}_{v_1}((f_i)_\pi) \geq \mathrm{Inf}_{v_2}((f_i)_\pi) \quad \text{if and only if} \quad \mathrm{Inf}_{v_1}((f_h)_\pi) \geq \mathrm{Inf}_{v_2}((f_h)_\pi).$$

Lower bounding the size of BuildTopDownDT(f, ε)

Let T_{exact} denote the tree returned by BuildTopDownDT(f_h) and T_{approx} that returned by BuildTopDownDT(f_h, ε). (So T_{exact} computes f_h, and T_{approx} is an ε-approximation of f_h.) Our goal is to lower bound the size of T_{approx}. We will in fact establish something stronger: our lower bound holds for *any pruning* of T_{exact} that is an ε-approximation of T_{exact}, where a pruning of a tree T is any tree obtained by iteratively removing leaves from T in a bottom-up fashion. Since BuildTopDownDT(f, ε) is simply BuildTopDownDT(f_h) terminated early, we have that T_{approx} is indeed a pruning of T_{exact}.

Let V_{exact} be defined as follows:

$$V_{\text{exact}} \coloneqq \{v \colon v \text{ is the first node in a path of } T_{\text{exact}} \text{ that queries a } z\text{-variable}\}.$$

We define $V_{\text{approx}} \subseteq V_{\text{exact}}$ analogously. At a very high level, our proof of Theorem 4(b) will proceed by showing that V_{exact} has large size, and V_{approx} has to contain many nodes in V_{exact}. For the remainder of this proof, we will need that r and ℓ are chosen to satisfy:

$$\frac{2 \ln r}{r} < 2^{-\ell}. \tag{4}$$

▶ **Lemma 26** (All nodes in V_{exact} occur deep within T_{exact})**.** *Fix $v \in V_{\text{exact}}$ and let π denote the path in T_{exact} that leads to v. Then $|\pi| \geq kh$.*

Proof. Suppose without loss of generality that v is a query to z_1. We claim that $y_j^{(i)} \in \pi$ for all $i \in [h]$ and $j \in [k]$, from which the lemma follows. Fix $i \in [h]$. We will prove there are at least k queries to variables in S_i within π, and that the first k of these queries have to be $y_j^{(i)}$ for $j \in [k]$. We prove both these claims simultaneously by induction on k.

- (Base case.) Seeking a contradiction, suppose π does not contain any queries to variables in S_i, in which case $(f_i)_\pi \equiv f_i$. Since z_1 is the variable queried at the root of $(f_h)_\pi$, it is the most influential variable within $(f_h)_\pi$. By Corollary 25, it follows that z is the most influential variable within $(f_i)_\pi \equiv f_i$. This contradicts Equation (4) since

$$\mathrm{Inf}_{y_1^{(i)}}(f_i) = \frac{\ell + 1}{2^\ell} \quad \text{and} \quad \mathrm{Inf}_{z_1}(f_i) < \mathrm{Inf}_{z_1}(\textsc{Tribes}_r) = (1 \pm o(1)) \cdot \frac{\ln r}{r}.$$

  Therefore π has to contain at least one variable in S_i. Let $u \in \pi$ be the first query to a variable in S_i, which we claim must be $y_j^{(i)}$ for some j. Let $\pi_u \subset \pi$ be the path in T_{exact} that leads to u. Again, we have that u must be the most influential variable within $(f_h)_{\pi_u}$, and hence, by Corollary 25, it is the most influential within $(f_i)_{\pi_u}$. Since π_u does not contain any queries to variables in S_i, we have that $(f_i)_{\pi_u} \equiv f_i$, and hence u must be $y_j^{(i)}$ for some j since these are the most influential variables within f_i.
- (Inductive step.) Fix $k' < k$, and suppose we have established that there are at least k' queries to variables in S_i within π, the first k' of which are to $y^{(i)}$-variables. We first claim that there is at least one more query to variable in S_i within π. Suppose not. It follows that z_1 must be the most influential variable within $(f_h)_\pi$, and hence, by Corollary 25, it is the most influential variable within $(f_i)_\pi$. This is a contradiction, since z_1 is less influential than any of the $k - k'$ many $y_j^{(i)}$ variables that are not queried by π.
  Therefore π has to contain at least one more query to a variable in S_i. Let $u \in \pi$ be the $(k+1)^{st}$ query to a variable in S_i, which we claim must be $y_j^{(i)}$ for some j. Let $\pi_u \subset \pi$ be the path in T_{exact} that leads to u. Again, we have that u must be the most

influential variable within $(f_h)_{\pi_u}$, and hence, by Corollary 25, it is the most influential within $(f_i)_{\pi_u}$. Since π_u contains exactly k' queries to variables S_i, and all these queries are to $y^{(i)}$ variables, we have that u must be $y_j^{(i)}$ for one of the remaining $k - k'$ many $y^{(i)}$-variables since these are the most influential variables within $(f_i)_{\pi_u}$.

This completes the inductive proof of Lemma 26. ◀

▶ **Lemma 27.** *Fix $v \in V_{\text{exact}}$ and let π denote that path in T_{exact} that leads to v. Then $(f_h)_\pi \equiv \textsc{Tribes}_r$.*

Proof. Suppose $(f_h)_\pi \not\equiv \textsc{Tribes}_r$. Our proof of Lemma 26 shows that π contains every y-variable, so it must be the case that some x-variable remains relevant (i.e. has nonzero influence) in $(f_h)_\pi$. Let $i^* \geq 1$ be the highest value of i for which there is a relevant $x^{(i)}$-variable in $(f_h)_\pi$. Assume without loss of generality that $x_1^{(i)}$ remains relevant, and that z_1 is that z-variable that is queried at v.

Since z_1 is queried at the root of $(f_h)_\pi$, we have that it must be maximally influential in $(f_h)_\pi$, and in particular,

$$\mathrm{Inf}_{z_1}((f_h)_\pi) \geq \mathrm{Inf}_{x_1^{(i^*)}}((f_h)_\pi).$$

Applying Corollary 25, we infer that

$$\mathrm{Inf}_{z_1}((f_{i^*})_\pi) \geq \mathrm{Inf}_{x_1^{(i^*)}}((f_{i^*})_\pi). \tag{5}$$

Let us say that an input (x, y, z) to f_{i^*} is *z-dependent* if

$$\textsc{Threshold}_{\ell,1}(x^{(i)}) = 0 \quad \text{for all } 1 \leq i \leq i^*.$$

Note that the output of f_{i^*} on any z-dependent input is $\textsc{Tribes}_r(z)$. Since π contains every y-variable, it fixes $\textsc{Parity}_k(y^{(i^*)})$ to either -1 or 1; we assume without loss of generality that $\textsc{Parity}_k(y^{(i^*)})_\pi \equiv 1$. We have that

$$\begin{aligned}
\mathrm{Inf}_{x_1^{(i^*)}}((f_{i^*})_\pi) &\geq \Pr_{(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})}[(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}) \text{ is } z\text{-dependent}] \cdot \Pr_{\boldsymbol{z}}[\textsc{Tribes}_r(\boldsymbol{z}) \neq 1] \\
&\qquad \times \mathrm{Inf}_{x_1^{(i^*)}}\big(\textsc{Threshold}_{\ell,1}(x^{(i^*)})_\pi\big) \\
&\geq \Pr_{(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})}[(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}) \text{ is } z\text{-dependent}] \cdot \tfrac{1}{2} \cdot 2^{-(\ell-1)} \\
&= \Pr_{(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})}[(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}) \text{ is } z\text{-dependent}] \cdot 2^{-\ell}.
\end{aligned}$$

The second inequality uses the fact that $\mathrm{Inf}_{x_1^{(i^*)}}(\textsc{Threshold}_{\ell,1}(x^{(i^*)})_\pi) \geq 2^{-(\ell-1)}$, which holds with equality when exactly one other $x^{(i^*)}$-variable is in π and that variable is set to 1.[16]

On the other hand, we have that

$$\begin{aligned}
\mathrm{Inf}_{z_1}((f_{i^*})_\pi) &\leq \Pr_{(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})}[(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}) \text{ is } z\text{-dependent}] \cdot \mathrm{Inf}_{z_1}[\textsc{Tribes}_r(z)] \\
&< \Pr_{(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})}[(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}) \text{ is } z\text{-dependent}] \cdot \frac{2\ln r}{r}. \qquad \text{(Fact 20)}
\end{aligned}$$

[16] In this derivation, we have assumed that $\textsc{Tribes}_r$ is *perfectly* balanced, i.e. that $\Pr[\textsc{Tribes}_r(\boldsymbol{z}) = 1] = \frac{1}{2}$, when in fact $\Pr[\textsc{Tribes}_r(\boldsymbol{z}) = 1] = \frac{1}{2} \pm o(1)$ (recall Fact 20). The same proof goes through if one carries around the additive $o(1)$ factor.

These bounds on influences, along with Equation (5), imply that $2^{-\ell} < \frac{2\ln r}{r}$. This contradicts our assumption on the relationship between ℓ and r (Equation (4)), and the proof is complete. ◀

We are now ready to lower bound the size of T_{approx}.

▷ Claim 28 (Lower bound on the size of T_{approx}). Fix $\varepsilon \in (0, \frac{1}{2})$ and let $c = (\frac{1}{2} - \varepsilon)/2$. If

$$\left(1 - \frac{\ell+1}{2^\ell}\right)^h \geq (2+c)\varepsilon, \tag{6}$$

then $|V_{\text{approx}}| \geq \Omega(\varepsilon \cdot 2^{kh})$. Consequently, the size of T_{approx} is also at least $\Omega(\varepsilon \cdot 2^{kh})$.

Proof. An input to f_h reaches *some* node in V_{exact} if and only if $\textsc{Threshold}_{\ell,1}(x^{(i)}) = 0$ for all $1 \leq i \leq h$. The fraction of inputs that satisfies this is exactly $(1 - \frac{\ell+1}{2^\ell})^h$, which is at least $(2+c)\varepsilon$ by our choice of parameters given by Equation (6).

Fix $v \in V_{\text{exact}}$. If $v \notin V_{\text{approx}}$, then T_{approx} assigns all inputs reaching v the same -1 or $+1$ value, whereas f_h labels half of them -1 and half of them $+1$ (Lemma 27). Therefore, T_{approx} errors on half of the inputs that each v. On the other hand, if $v \in V_{\text{approx}}$, we have by Lemma 26 that at most a 2^{-kh} fraction of inputs reach this specific v. Combining all of the above observations, it follows that

$$\text{error}(T_{\text{exact}}, T_{\text{approx}}) \geq \tfrac{1}{2}\left((2+c)\varepsilon - |V_{\text{approx}}| \cdot 2^{-kh}\right).$$

Since $\text{error}(T_{\text{exact}}, T_{\text{approx}}) \leq \varepsilon$, it follows that

$$\varepsilon \geq \tfrac{1}{2}\left((2+c)\varepsilon - |V_{\text{approx}}| \cdot 2^{-kh}\right),$$

and the claim follows by rearranging. ◁

Theorem 4(b) now follows from Claim 22 and Claim 28 by setting parameters appropriately:

Proof of Theorem 4(b). Choosing

$$\begin{aligned} h &= \Theta\left(\frac{2^\ell}{\ell} \cdot \log(1/\varepsilon)\right) && \text{(to satisfy Equation (6))} \\ r &= \Theta(\ell 2^\ell) && \text{(to satisfy Equation (4))} \\ k &= \Theta(h \log \ell), \end{aligned}$$

we may apply Claim 22 and Claim 28 to get that

$$\text{size}(f_h) \leq 2^{O(k \log k)} \quad \text{whereas} \quad \textsc{TopDownDTSize}(f, \varepsilon) \geq 2^{\Omega_\varepsilon(k^2/\log\log k)}.$$

This is a separation of s versus $s^{\tilde{\Omega}(\log s)}$. ◀

▶ Remark 29. For our choice of parameters above, we have that $s(n) = \text{size}(f_h) = 2^{\tilde{\Theta}(\sqrt{n})}$, where $n = h(\ell + k) + r$ is the number of variables of f_h. A standard padding argument yields the same s versus $s^{\tilde{\Omega}(\log s)}$ separation for any function $s(n) \leq 2^{\tilde{O}(\sqrt{n})}$.

7 Lower bounds on TopDownDTsize for monotone functions: Proof of Theorem 7

7.1 Size separation for exact representation: Proof of Theorem 7(a)

We will give a family of monotone functions, $\{f_h\}_{h\in\mathbb{N}}$ whose BuildTopDownDT tree has exponential size compared to the optimal tree. First, we define a few terms which will be useful for our monotone constructions.

▶ **Definition 30** (Comparing vectors and upper/lower shadows). *For any $x, y \in \{0,1\}^n$, we use $x \preceq y$ to represent*

$$x \preceq y \iff x_i \leq y_i \text{ for all } i \in [n]$$

and $\succeq$ is defined similarly. For any vector x, the upper shadow *of x is the set of all vectors y such that $x \preceq y$. Similarly, the* lower shadow *of x is the set of all vectors y such that $x \succeq y$.*

Defining the family of functions witnessing the separation

Each f_h in $\{f_h\}_{h\in\mathbb{N}}$ is a function over $5h+1$ boolean variables $x^{(1)}, x^{(2)}, \ldots, x^{(h)} \in \{0,1\}^4$, $y^{(1)}, \ldots, y^{(h)} \in \{0,1\}$, and $z \in \{0,1\}$, and is defined inductively as follows:

$$f_0(z) = z,$$

and for $h \geq 1$, we fix $x^* \coloneqq (0,0,1,1)$ and define

$$f_h(x,y,z) = \begin{cases} f_{h-1}(x,y,z) & \text{if } x^{(h)} = x^* \\ +1 & \text{if } x^{(h)} \succeq x^* \text{ and } x^{(h)} \neq x^* \\ -1 & \text{if } x^{(h)} \preceq x^* \text{ and } x^{(h)} \neq x^* \\ y & \text{otherwise.} \end{cases}$$

It is straightforward to verify that f_h is indeed monotone. We will show that f_h can be computed by a tree of size $O(h)$, but that BuildTopDownDT produces a tree of size $2^{\Omega(h)}$. For the first claim, we construct a decision tree for f_h directly from its definition. We start with a complete tree on the $x^{(h)}$ variables – this complete tree has size 2^4, a constant. At one of the branches, we recursively build a tree for f_{h-1}; at all the other branches, we build a tree of size 1 or 2 computing one of $-1, 1$, or $y^{(h)}$. The result is a tree of size $O(h)$.

On the other hand, we claim that BuildTopDownDT will build a tree of size $2^{\Omega(h)}$, as depicted in Figure 6. In f_h, $y^{(h)}$ has influence $\frac{9}{16}$ and all the other variables have influence at most $\frac{1}{2}$. Hence, $y^{(h)}$ will be placed at root. Then, BuildTopDownDT will query enough of $x^{(h)}$ to determine whether the output should be $-1, +1$, or f_{h-1}. If the output should be f_{h-1}, which will occur once for each choice of y, then the entire tree T_{h-1} will be placed. Hence, the size of T_h is more than double the size of T_{h-1}, and BuildTopDownDT builds a tree of size $2^{\Omega(h)}$.

7.2 Size separation for approximate representation: Theorem 7(b)

For any ε, we will prove there exists a function f with optimal tree size s but for which the tree BuildTopDownDT(f, ε) builds has size $s^{\tilde{\Omega}(\sqrt[4]{\log s})}$. The following function, a biased version of the Tribes function defined in Definition 19, will be used as a building block in our monotone construction.

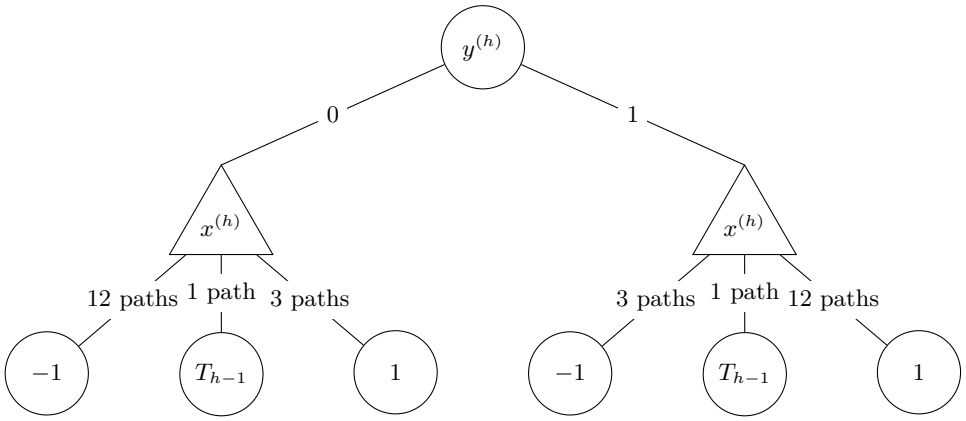


Figure 6 The tree that BuildTopDownDT builds for f_h. It will first query $y^{(h)}$, followed by the variables of $x^{(h)}$. For most choices of $y^{(h)}$ and $x^{(h)}$, the function is determined, and BuildTopDownDT will place a constant leaf equal to ± 1. However, the paths with $y = 0, x^{(h)} = x^*$ and $y = 1, x^{(h)} = x^*$ each include a copy of the tree for T_{h-1}.

▶ **Definition 31** (Biased Tribes). *Fix any input length ℓ and $\delta \in (0,1)$. We define* $\textsc{Tribes}_{\ell,\delta} : \{0,1\}^\ell \to \{\pm 1\}$ *to be the read-once DNF with $\lfloor \frac{\ell}{w} \rfloor$ terms of width exactly w over disjoint sets of variables (with some variables possibly left unused), where $w = w(\ell,\delta) \approx \log(\ell) \pm \log\log(1/\delta)$ is chosen such that $\Pr[\textsc{Tribes}_{\ell,\delta}(\boldsymbol{x}) = 1]$ is as close to δ as possible.*[17]

▶ **Fact 32** (Variable influences in biased Tribes). *All variables in $\textsc{Tribes}_{\ell,\delta}$ and $\textsc{Tribes}_{\ell,1-\delta}$ have influence at most*

$$(2 + o(1)) \cdot \delta \log(1/\delta) \cdot \frac{\log \ell}{\ell}.$$

Proof. We prove the lemma for the case of $\textsc{Tribes}_{\ell,1-\delta}$. (The calculations for $\textsc{Tribes}_{\ell,\delta}$ are very similar, and both claims are special cases of more general facts about variable influences in DNF formulas [61].) Suppose

$$\textsc{Tribes}_{\ell,1-\delta}(x) = T_1(x) \vee \cdots T_{\frac{\ell}{w}}(x),$$

where the T_i's are disjoint terms of width exactly w. We first observe that since

$$\begin{aligned}\delta = \Pr_{\boldsymbol{x}\sim\{0,1\}^n}[\textsc{Tribes}_{\ell,1-\delta}(\boldsymbol{x}) = 1] &= \Pr[\text{all } T_i(\boldsymbol{x}) \text{ are falsified by } \boldsymbol{x}] \\ &= (1 - 2^{-w})^{\ell/w} \approx e^{-\ell/w 2^w},\end{aligned}$$

we have that $w = (1 \pm o(1))(\log \ell - \log\log \ell - \log\log(1/\delta))$. The influence of any variable $i \in [n]$ on $\textsc{Tribes}_{\ell,1-\delta}$ is the probability, over a uniform $\boldsymbol{x}$ that each other variable j in i's term has $\boldsymbol{x}_j = 1$ and all other clauses evaluate to 0 under $\boldsymbol{x}$:

$$\begin{aligned}\mathrm{Inf}_i(\textsc{Tribes}_{\ell,1-\delta}) &= 2^{-(w-1)} \cdot (1 - 2^{-w})^{(\ell/w)-1} \\ &\le 2\delta \cdot 2^{-w} \\ &= (1 \pm o(1)) \cdot 2\delta \log(1/\delta) \cdot \frac{\log \ell}{\ell}.\end{aligned}$$

◀

[17] Although the acceptance probability of $\textsc{Tribes}_{\ell,\delta}$ cannot be made *exactly* δ due to granularity issues, it will be the case that $\textsc{Tribes}_{\ell,\delta} = \delta \pm o(1)$. For clarity, we will assume for the rest of this paper that the acceptance probability of $\textsc{Tribes}_\delta$ is exactly δ, noting that all of our proofs still go through if one carries around the $o(1)$ factor.

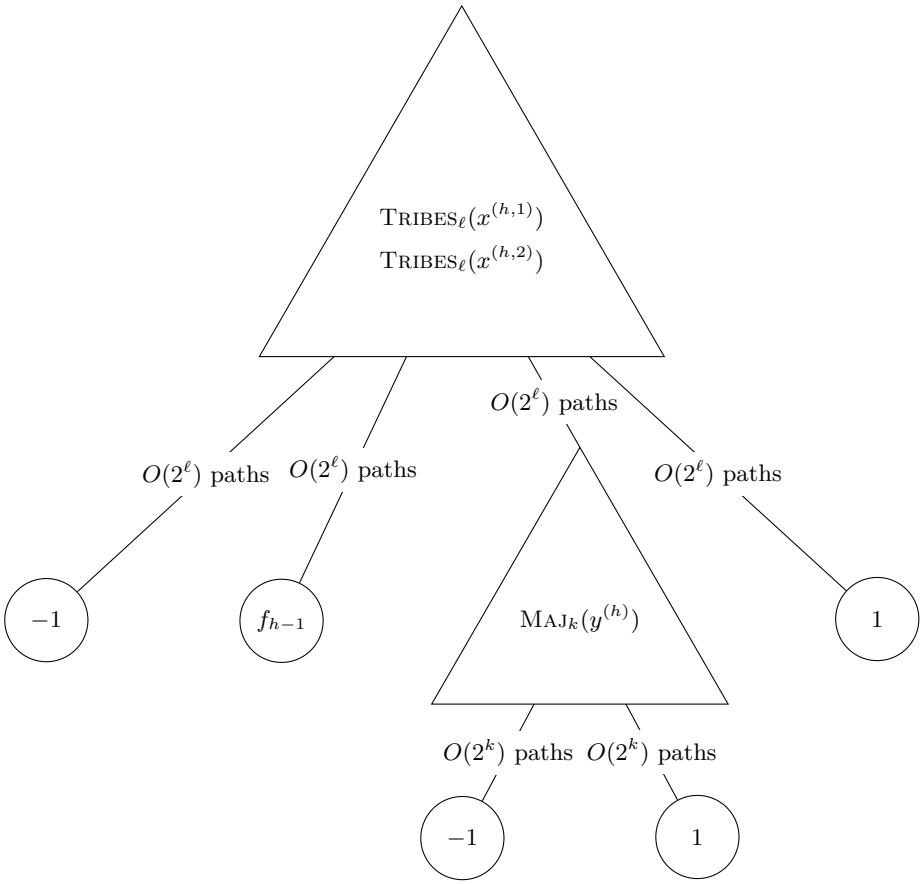


Figure 7 A small decision tree that computes f_h.

Defining the family of functions witnessing the separation

Each f_h in the family $\{f_h\}_{h\in\mathbb{N}}$ is a function over $h(2\ell+k)+r$ boolean variables, $x^{(1,1)}, x^{(1,2)}, \ldots, x^{(h,1)}, x^{(h,2)} \in \{0,1\}^\ell$, $y^{(1)}, \ldots, y^{(h)} \in \{0,1\}^k$, and $z \in \{0,1\}^r$, and is defined inductively as follows:

$$f_0(z) = \textsc{Tribes}_r(z),$$

and for $h \geq 1$,

$$f_h(x,y,z) = \begin{cases} -1 & \text{if } \textsc{Tribes}_{\ell,\delta}(x^{(h,1)}) = \textsc{Tribes}_{\ell,1-\delta}(x^{(h,2)}) = 0 \\ f_{h-1}(x,y,z) & \text{if } \textsc{Tribes}_{\ell,\delta}(x^{(h,1)}) = 0 \text{ and } \textsc{Tribes}_{\ell,1-\delta}(x^{(h,2)}) = 1 \\ \textsc{Maj}_k(y^{(h)}) & \text{if } \textsc{Tribes}_{\ell,\delta}(x^{(h,1)}) = 1 \text{ and } \textsc{Tribes}_{\ell,1-\delta}(x^{(h,2)}) = 0 \\ +1 & \text{otherwise.} \end{cases}$$

Clearly f_h is monotone in $x^{(h,1)}$ and $x^{(h,2)}$. Furthermore, since each of the functions $-1, +1$, and $\textsc{Maj}_k(y^{(h)})$, are monotone, if f_{h-1} is monotone then so is f_h.

▷ Claim 33 (Optimal size of f_h). Choose any integers $\ell, h, r, k > 0$ and let Then, $f_{h,\ell,k,r}$ has optimal decision tree size

$$\begin{aligned}\mathrm{size}(f_h) &\leq (\mathrm{size}(\textsc{Tribes}_{\ell,\delta}) \cdot \mathrm{size}(\textsc{Tribes}_{\ell,1-\delta}))^{O(h)} \cdot (\mathrm{size}(\textsc{Maj}_k) + \mathrm{size}(\textsc{Tribes}_r)) \\ &\leq 2^{O(h\cdot\ell \log\log \ell/\log \ell)} \cdot (2^k + 2^{O(r\log\log r/\log r)}). \qquad \text{(Fact 20)}\end{aligned}$$

Proof. As in the proofs of the previous separations, this upper bound is witnessed by the natural decision tree that one builds by following the definition of f_h. This tree first evaluates $\textsc{Tribes}_{\ell,\delta}(x^{(h,1)})$ followed by $\textsc{Tribes}_{\ell,1-\delta}(x^{(h,2)})$, resulting in a tree of size $(\mathrm{size}(\textsc{Tribes}_{\ell,\delta}) \cdot \mathrm{size}(\textsc{Tribes}_{\ell,1-\delta}))$. At the end of each branch, we either recursively build a tree for f_{h-1}, or a tree for $\textsc{Maj}_k(y^{(h)})$, or place constants $\{\pm 1\}$ as leaves. Please refer to Figure 7. ◁

The remainder of this section is devoted to lower bounding $\textsc{TopDownDTSize}(f_h, \varepsilon)$, the size of the tree T_{approx} that $\textsc{BuildTopDownDT}$ constructs to ε-approximate f_h.

7.2.1 "Mostly precedes"

By choosing parameters appropriately, we will ensure that when T_{approx} begins by querying the variables of $\textsc{Maj}_k(y^{(h)})$. The first technical challenge that arises is the following: unlike $\textsc{Parity}_k(y^{(h)})$ in our proof of Theorem 4(b), the influence of variables in $\textsc{Maj}_k(y^{(h)})$ changes as variables are queried. For example, the influence of the remaining variables of $\textsc{Maj}_k(y^{(h)})$ after $\frac{k}{2}$ variables have been queried is 0 if all of the queried variables are 1 and is $\Theta(\frac{1}{\sqrt{k}})$ if half of the variables queries are 0 and half are 1. Hence, in T_{approx}, the number of nodes from $y^{(h)}$ queried before some non-$y^{(h)}$-variable is queried will vary by path. (In other words, the analogue of Lemma 26 in the proof of Theorem 4(b) is somewhat trickier to establish.)

To handle this, we define the following notion, which will allow us to show that *most* $y^{(h)}$-variables are before other variables in *most* paths of the tree (Corollary 37).

▶ **Definition 34** (Mostly precedes). *Let S be a subset of the relevant variables of f_h. We say that $y^{(i)}$-*variables mostly precede S in T_{approx} *if for every path π in T_{approx} leading to a first query to a variable in S, and every $j \in [k]$,*

$$\mathrm{Inf}_j\big(\textsc{Maj}_k(y^{(i)})_\pi\big) \leq \frac{1}{100\sqrt{k}}.$$

(For some intuition behind Definition 34, we note that *pre-restriction*, the influences of variables in $\textsc{Maj}_k$ are given by:

$$\mathrm{Inf}_j(\textsc{Maj}_k) = \frac{1}{k} \cdot \binom{k}{\frac{k}{2}} \sim \frac{\sqrt{2/\pi}}{\sqrt{k}} \quad \text{for all } j \in [k],$$

which is significantly larger than the $\frac{1}{100\sqrt{k}}$ of Definition 34.)

With Definition 34 in hand, we now begin to formalize the structure of T_{approx} as depicted in Figure 8. For each $i \in [h]$, we define

$$R_i = \{x^{(i,1)}, x^{(i,2)}, \text{ and } z \text{ variables}\}.$$

▶ **Lemma 35.** *There is a universal constant c such that the following holds. Suppose*

$$\frac{c}{\sqrt{k}} \geq \frac{1}{\delta^2} \cdot \max\left\{\frac{\delta \log(1/\delta)\log \ell}{\ell}, \frac{\log r}{r}\right\}. \tag{7}$$

Then for all $i \in [h]$, we have that $y^{(i)}$-variables mostly precede R_i in T_{approx}.

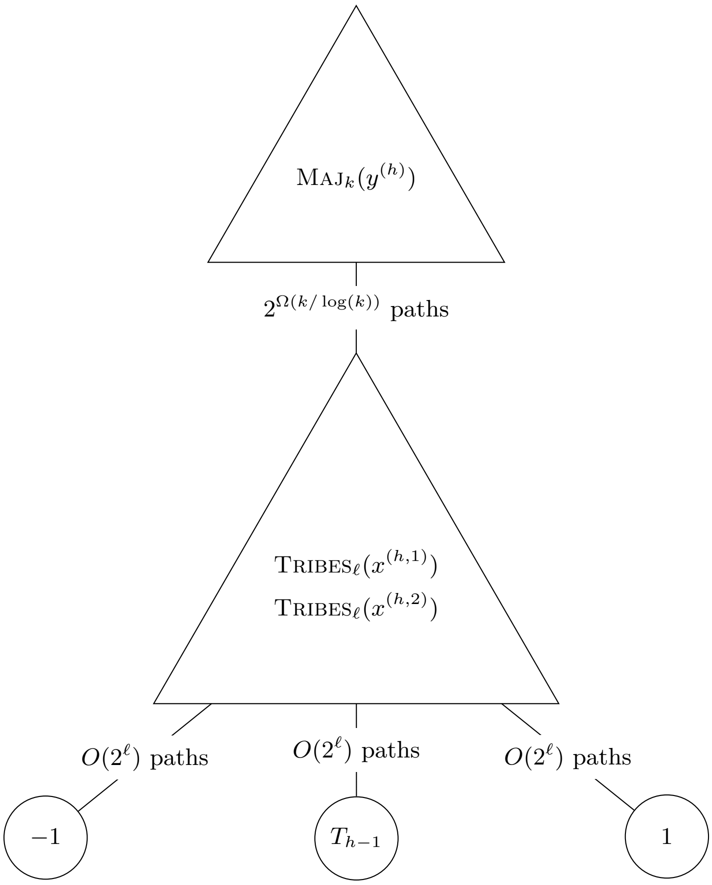


Figure 8 With appropriately chosen parameters, the $y^{(h)}$-variables are the most influential in f_h, so the tree built by BUILDTOPDOWNDT will query them first. Our analysis shows that this leads to a tree of size $2^{\Omega(kh/\log k)}$ (cf. Figure 7).

Proof. Fix $i \in [h]$. Let π be a path in T_{approx} that leads to a first query to a variable in $v \in R_i$. Since v is maximally influential in $(f_h)_\pi$, we may apply Corollary 25 to infer that v is also maximally influential in $(f_i)_\pi$ (and in particular, v is more influential than any $y^{(i)}$ variable). We have that

$$\begin{aligned}\mathrm{Inf}_v((f_i)_\pi) &\le \max\left\{\mathrm{Inf}_j(\mathrm{TRIBES}_{\ell,\delta}(x^{(i,1)})), \mathrm{Inf}_j(\mathrm{TRIBES}_{\ell,1-\delta}(x^{(i,2)})), \mathrm{Inf}_j(\mathrm{TRIBES}_r(z))\right\}\\ &\le \max\left\{(2+o(1))\cdot\delta\log(1/\delta)\cdot\frac{\log\ell}{\ell}, (1+o(1))\cdot\frac{\ln r}{r}\right\}.\end{aligned}$$

(Fact 20 and Fact 32)

On the other hand, for any $j \in [k]$,

$$\begin{aligned}\mathrm{Inf}_{y_j^{(i)}}((f_i)_\pi) &= \Pr\left[\mathrm{TRIBES}_{\ell,\delta}(\boldsymbol{x}^{(h,1)}) = 1, \mathrm{TRIBES}_{\ell,1-\delta}(\boldsymbol{x}^{(h,2)}) = 0\right]\cdot\mathrm{Inf}_j\left(\mathrm{MAJ}_k(y^{(i)})_\pi\right)\\ &= \delta^2\cdot\mathrm{Inf}_j\left(\mathrm{MAJ}_k(y^{(i)})_\pi\right).\end{aligned}$$

Since $\mathrm{Inf}_{y_j^{(i)}}((f_i)_\pi) \le \mathrm{Inf}_v((f_i)_\pi)$, the bounds above imply that

$$\mathrm{Inf}_j\big(\textsc{Maj}_k(y^{(i)})_\pi\big) \le \frac{1}{\delta^2} \cdot \max\left\{(2+o(1))\cdot\delta\log(1/\delta)\cdot\frac{\log\ell}{\ell}, (1+o(1))\cdot\frac{\ln r}{r}\right\}.$$

The lemma follows: by choosing c to be a sufficiently small constant in Equation (7), we can ensure that $\mathrm{Inf}_j\big(\textsc{Maj}_k(y^{(i)})_\pi\big) \le \frac{1}{100\sqrt{k}}$. ◀

▶ **Lemma 36.** *There is a universal constant c such that the following holds. Fix $i \in [h]$ and consider a uniform random $\boldsymbol{y}^{(i)} \in \{0,1\}^k$. The probability there is an input u to f_h consistent with $\boldsymbol{y}^{(i)}$ such that T_{approx}, on input u, queries an R_i-variable before a querying at least $ck/\log k$ many $\boldsymbol{y}^{(i)}$-variables is $O(k^{-2})$.*

Proof. Fix an outcome $y^{(i)}$ of $\boldsymbol{y}^{(i)}$. Suppose that there is an input u consistent with $y^{(i)}$ such that T_{approx}, on input u, queries an R_i-variable after querying only $< ck/\log k$ many $y^{(i)}$-variables. Call such a $y^{(i)}$ outcome *bad*, and let π denote the corresponding path in T_{approx} that leads to the first query to an R_i-variable. Since $y^{(i)}$-variables mostly precede R_i in T_{approx}, we have that

$$\mathrm{Inf}_j\big(\textsc{Maj}_k(y^{(i)})_\pi\big) \le \frac{1}{100\sqrt{k}} \quad \text{for all } j \in [k].$$

For this to hold, it must be the case that among the $t < ck/\log k$ many $y^{(i)}$-variables that occur in π, the discrepancy between the number of 0's and 1's is $\Omega(\sqrt{k})$. We can therefore bound

$$\begin{aligned}
\Pr_{\boldsymbol{y}^{(i)}\in\{0,1\}^k}[\boldsymbol{y}^{(i)} \text{ is bad}] &\le \sum_{t=1}^{ck/\log k} \Pr_{\boldsymbol{b}\sim\mathrm{Bin}(t,\frac{1}{2})}\left[|\boldsymbol{b}-\tfrac{t}{2}| \ge \Omega(\sqrt{k})\right] \\
&\le \sum_{t=1}^{ck/\log k} e^{-\Theta(k/t)} && \text{(Hoeffding's inequality)} \\
&\le \frac{ck}{\log k}\cdot e^{-\Theta((\log k)/c)} \ll \frac{1}{k^2},
\end{aligned}$$

where the final inequality holds by choosing c to be a sufficiently small constant. ◀

By a union bound over $i \in [h]$, we have the following corollary of Lemma 36 (which can be thought of as being roughly analogous to Lemma 26 in the proof of Theorem 4(b)):

▶ **Corollary 37** (Most queries to z-variables are deep within T_{approx})**.** *Let $\boldsymbol{y} = (\boldsymbol{y}^{(1)},\dots,\boldsymbol{y}^{(k)})$ be uniform random. The probability that there is an input u to f_h consistent with $\boldsymbol{y}$ such that T_{approx}, on input u, queries a z-variable before querying at least $(ck/\log k)\cdot h$ many $\boldsymbol{y}$-variables is $O(h/k^2)$.*

We are finally ready to lower bound the size of T_{approx}:

▷ Claim 38 (Lower bound on the size of T_{approx}). Fix $\varepsilon \in (0,\frac{1}{2})$ and let $c = (\frac{1}{2}-\varepsilon)/2$. If

$$(1-\delta)^{2h} \ge (2+c)\varepsilon \tag{8}$$

$$h \le k, \tag{9}$$

then the size of T_{approx} is at least $2^{\Omega(hk/\log k)}$.

Proof. We will call an input to f_h *z-dependent* if it satisfies:

$$\textsc{Tribes}_{\ell,\delta}(x^{(i,1)}) = 0 \text{ and } \textsc{Tribes}_{\ell,1-\delta}(x^{(i,2)}) = 1 \text{ for all } i \in [h].$$

Note that the output of f_h on any z-dependent input is $\textsc{Tribes}_r(z)$. Let us define $\zeta(y^{(1)}, \ldots, y^{(h)})$ to be the $\{0,1\}$-valued indicator of whether there is an input u consistent with $y^{(1)}, \ldots, y^{(h)}$ such that T_{approx} on input u queries a z-variable. For any fixed $y = (y^{(1)}, \ldots, y^{(h)})$,

- If $\zeta(y) = 0$, then T_{approx} must assign the same -1 or $+1$ value to all z-dependent inputs consistent with y;
- The fraction of z-dependent inputs that are consistent with y is

$$\Pr\left[\textsc{Tribes}_{\ell,\delta}(\boldsymbol{x}^{(i,1)}) = 0 \text{ and } \textsc{Tribes}_{\ell,1-\delta}(\boldsymbol{x}^{(i,2)}) = 1 \text{ for all } i \in [h]\right] = (1-\delta)^{2h},$$

  which is at least $(2+c)\varepsilon$ by Equation (8). Furthermore, since output of f_h on any z-dependent input is $\textsc{Tribes}_r(z)$, among the z-dependent inputs that are consistent with y, we have that f_h labels half of them -1 and half of them $+1$.

Therefore,

$$\text{error}(T_{\text{approx}}, f_h) \geq \tfrac{1}{2} \cdot (2+c)\varepsilon \cdot \Pr[\zeta(\boldsymbol{y}) = 0].$$

Since $\text{error}(T_{\text{approx}}, f_h) \leq \varepsilon$, it follows that $\Pr[\zeta(\boldsymbol{y}) = 1] \geq \Omega(1)$. Next, applying Corollary 37 along with our assumption that $h \leq k$ (Equation (9)), we further have that

$$\Pr_{\boldsymbol{y}}\left[\exists\, \boldsymbol{y}\text{-consistent } u \text{ s.t. } T_{\text{approx}}(u) \text{ queries } z\text{-variable after} \geq \tfrac{ck}{\log k} \cdot h \text{ many } \boldsymbol{y}\text{-variables}\right] \geq \Omega(1)$$

On the other hand, for any fixed path π in T_{approx} that queries $\geq \Omega(kh/\log k)$ many y-variables, at most a $2^{-\Omega(kh/\log k)}$ fraction of $\boldsymbol{y}$'s can be consistent with this specific π. We conclude that the size of T_{approx} must be at least $2^{\Omega(kh/\log k)}$, and the proof is complete. ◁

Theorem 7(b) now follows from Claim 33 and Claim 38 by setting parameters appropriately:

Proof of Theorem 7(b). By choosing

$$\begin{aligned}
\delta &= \Theta\left(\sqrt[3]{(\log \ell)^4 \log(1/\varepsilon)/\ell}\right) \\
k &= \Theta\left(\sqrt[3]{\ell^4 \log(1/\varepsilon)^2/(\log \ell)^4}\right) \\
r &= \Theta(k) \\
h &= \Theta\left(\frac{1}{\delta} \cdot \log(1/\varepsilon)\right),
\end{aligned}$$

we satisfy Equations (7) to (9). We may therefore apply Claim 33 to get that the optimal size of f_h is upper bounded by:

$$\text{size}(f_h) \leq \exp\left(O(\sqrt[3]{\ell^4 \log(1/\varepsilon)^2/(\log \ell)^4})\right).$$

On the other hand, by Claim 38, we have that

$$\textsc{TopDownDTSize}(f_h, \varepsilon) \geq 2^{\Omega(kh/\log k)} = \exp\left(\Omega(\sqrt[3]{\ell^5 \log(1/\varepsilon)^4/(\log \ell)^{11}})\right).$$

This is a separation of s versus $s^{\tilde{\Omega}(\sqrt[4]{\log(s)})}$. ◀

▶ Remark 39. For our choice of parameters above, we have that $s(n) = \text{size}(f_h) = 2^{\tilde{\Theta}(n^{4/5})}$, where $n = h(2\ell + k) + r$ is the number of variables of f_h. A standard padding argument yields the same s versus $s^{\tilde{\Omega}(\sqrt[4]{\log s})}$ separation for any function $s(n) \leq 2^{\tilde{O}(n^{4/5})}$.

▶ Remark 40 (Depth separation). The same proof witnesses a separation of d versus $\tilde{\Omega}_\varepsilon(d^{5/4})$ for the optimal *depth* of f_h versus the depth of the tree that BuildTopDownDT(f_h, ε) builds. This disproves the conjecture of Fiat and Pechyony [23] discussed in Section 2, which states that BuildTopDownDT builds a tree of optimal depth for all monotone functions, even in the case of exact representation ($\varepsilon = 0$).

7.3 Lower bounds for all impurity-based heuristics

▶ **Proposition 41** (Splitting on the most influential variable of a monotone function maximizes purity gain). *Let $f : \{\pm 1\}^n \to \{\pm 1\}$ be a monotone boolean function.*[18] *Let $\mathscr{G} : [-1, 1] \to [0, 1]$ be a concave function that is symmetric around 0, and satisfies $\mathscr{G}(-1) = \mathscr{G}(1) = 0$ and $\mathscr{G}(0) = 1$. Suppose $i \in [n]$ maximizes:*

$$\mathscr{G}(\mathbb{E}[f]) - \tfrac{1}{2}(\mathscr{G}(\mathbb{E}[f_{x_i=-1}]) + \mathscr{G}(\mathbb{E}[f_{x_i=1}])), \tag{10}$$

Then $\mathbb{E}[f(\boldsymbol{x})\boldsymbol{x}_i] \geq \mathbb{E}[f(\boldsymbol{x})\boldsymbol{x}_j]$ for all $j \in [n]$.

Proof. For all functions f, not necessarily monotone, $\mathbb{E}[f]$ is precisely the average of $\mathbb{E}[f_{x_i=0}]$ and $\mathbb{E}[f_{x_i=1}]$. Because $\mathscr{G}$ is concave everywhere on its domain, Jensen's inequality ensures that $\mathscr{G}(\mathbb{E}[f])$ is greater than $\frac{1}{2}(\mathscr{G}(\mathbb{E}[f_{x_i=0}]) + \mathscr{G}(\mathbb{E}[f_{x_i=1}])$. Furthermore, again by concavity, we have that this difference increases with the difference between $\mathbb{E}[f_{x_i=1}]$ and $\mathbb{E}[f_{x_i=-1}]$. Therefore the variable $i \in [n]$ that maximizes purity gain (10) also maximizes $|\mathbb{E}[f_{x_i=1}] - \mathbb{E}[f_{x_i=-1}]|$.

For a monotone function $f : \{\pm 1\}^n \to \{\pm 1\}$, we have the following identity for all variables $j \in [n]$:

$$\begin{aligned}\text{Inf}_j(f) &= \Pr[f(\boldsymbol{x}) \neq f(\boldsymbol{x}^{\oplus j})] \\ &= \mathbb{E}[f(\boldsymbol{x})\boldsymbol{x}_j] \\ &= \mathbb{E}[f_{x_j=1}] - \mathbb{E}[f_{x_j=-1}].\end{aligned}$$

Thus, the variable that maximizes purity gain (10) is also the most influential variable of f. ◀

Recall that in Theorem 8, we claimed that our lower bound on TopDownDTSize(f_h, ε) holds not just for the specific algorithm BuildTopDownDT, but in fact *any* impurity-based top-down heuristic. To see this, note that in our proof of Theorem 7(b) described in Section 7.2, we never used any information about *which* leaf BuildTopDownDT chooses to split on at each stage, only that when a leaf *is* split, it is replaced by the most influential variable of the corresponding subfunction. In other words, just like our proof of Theorem 4(b), our proof of Theorem 7(b) applies not just to the specific tree build by BuildTopDownDT(f_h, ε); it in fact lower bounds the size of *any* pruning T_{approx} of $T_{\text{exact}} =$ BuildTopDownDT$(f, \varepsilon = 0)$ that is an ε-approximator to f_h.

By Proposition 41, any tree build by a impurity-based top-down heuristic is a pruning of T_{exact}, and hence our proof of Theorem 7(b) extends to establish Theorem 8.

[18] For this proof, for notational reasons it will be slightly more convenient for us to work with $\{\pm 1\}^n$ instead of $\{0, 1\}^n$ as the domain of f.

8 New proper learning algorithms: Proofs of Theorems 9 and 10

Recall that BuildTopDownDT builds an approximation to f iteratively. It starts with an empty bare tree T° and repeatedly replaces the leaf with the highest score with a query to that leaf's most influential variable. In section Section 5.2, we proved lower bounds on the score of the leaf that BuildTopDownDT selects. Using those lower bounds, in section Section 5.3, we are able to prove upper bounds on the size of the tree BuildTopDownDT needs to produce to guarantee at most ε error. If, instead, we only guaranteed that we would pick a leaf with score a fourth of that guaranteed by the lower bounds in Section 5.2, all of our upper bounds would still hold, up to constant factors in the exponent. In this section, we will show that it is possible to accurately enough estimate influences to guarantee we pick a leaf with score at least a fourth the maximum score. First, we provide a definition of score that takes into account both the leaf and the variable selected.

▶ **Definition 42** (score). *Given any function f, we define the score of a leaf ℓ and variable i as follows.*

$$\operatorname{score}(\ell, i) \coloneqq \Pr_{\boldsymbol{x}\sim\{0,1\}^n}[\boldsymbol{x} \text{ reaches } \ell] \cdot \operatorname{Inf}_i(f_\ell) = 2^{-|\ell|} \cdot \operatorname{Inf}_i(f_\ell).$$

We first show that it is possible to estimate scores sufficiently accurately for monotone functions just from random samples of a function, which proves Theorem 10. Let $\boldsymbol{S}$ be a random sample from a monotone function f, and recall Fact 5. We can estimate the score of a particular leaf and variable as follows.

$$\operatorname{score}(\ell, i, \boldsymbol{S}) = \mathop{\mathbb{E}}_{\boldsymbol{x},\boldsymbol{y}\in\boldsymbol{S}}\left[\mathbb{1}[\boldsymbol{x} \text{ reaches } \ell] \cdot f(\boldsymbol{x})(2\boldsymbol{x}_i - 1)\right].$$

Note that $\mathbb{E}_{\boldsymbol{S}}[\operatorname{score}(\ell, i, \boldsymbol{S}))] = \operatorname{score}(\ell, i)$. Let t be any score threshold and m be the number of examples in $\boldsymbol{S}$. By Chernoff bounds, for any particular leaf ℓ and variable x_i,

$$\Pr_{\boldsymbol{S}}\left[\operatorname{score}(\ell, i, \boldsymbol{S}) \le \tfrac{t}{2}\right] \le e^{-\frac{1}{8}t\cdot m} \quad \text{if } \operatorname{score}(\ell, i) \ge t$$
$$\Pr_{\boldsymbol{S}}\left[\operatorname{score}(\ell, i, \boldsymbol{S}) \ge \tfrac{t}{2}\right] \le e^{-\frac{1}{12}t\cdot m} \quad \text{if } \operatorname{score}(\ell, i) \le t/4.$$

At step j in BuildTopDownDT, there are $j+1$ leaves in T°. If t is the maximum score possible at that step, with probability at least $1-(j+1)e^{-\frac{1}{12}t\cdot m}$, the leaf and variable with maximum empirical score will have true score at least $\frac{t}{4}$. By Lemma 15, BuildTopDownDT(f, ε), at step j, there will always be a leaf with score at least $\frac{\varepsilon}{(j+1)\log(s)}$, where s is the decision tree size of f. Hence, the maximum empirical score will have true score at least $\frac{1}{4}$ the optimal value with probability at least $1-(j+1)e^{-\frac{\varepsilon\cdot m}{12(j+1)\log(s)}}$.

The probability that selecting the maximum empirical score is always within $\frac{1}{4}$ of the optimal value for every step from $j=0$ to $j=k-1$ is at least $1-k^2e^{-\frac{\varepsilon\cdot m}{12k\log(s)}}$. By setting the sample size to

$$m = O\left(\frac{k\log s}{\varepsilon}(\log k + \log(1/\delta))\right) \tag{11}$$

with probability at least $1-\delta$, we choose a sufficiently good leaf for k steps of BuildTopDownDT. Recall that, for monotone functions, BuildTopDownDT builds a decision tree of size at most

$$k = \min(s^{O(\log(s/\varepsilon)\log(1/\varepsilon))}, s^{O(\sqrt{\log s}/\varepsilon)}).$$

Hence, with probability at least $1-\delta$, taking $\min(s^{O(\log(s/\varepsilon)\log(1/\varepsilon))}, s^{O(\sqrt{\log s}/\varepsilon)})\log(1/\delta)$ is enough to learn to accuracy ε. This proves Theorem 10.

If f is not monotone, we cannot accurately estimate influences from just random samples. However, we can estimate influences if given access to *random edges* from f.

▶ **Definition 43** (Random edges). *For any function $f:\{0,1\}^n \to \{\pm 1\}$, a* random edge *is two points of the form $((\boldsymbol{x}, f(\boldsymbol{x})), (\boldsymbol{x}^{\oplus \boldsymbol{i}}, f(\boldsymbol{x}^{\oplus \boldsymbol{i}})))$, where $\boldsymbol{x} \in \{0,1\}^n$ and $\boldsymbol{i} \in [n]$ are both picked uniformly at random. A* random edge sample $\boldsymbol{E}$ *is a collection of random edges. Given random edge sample $\boldsymbol{E}$, we will use $\boldsymbol{E}_i$ to refer to all those edges in $\boldsymbol{E}$ in which the i^{th} bit of x is flipped.*

Given a random edge sample $\boldsymbol{E}$ of a function f, we will be able to accurately estimate influences of the variables in f, and learn f using BuildTopDownDT. We use the following estimate of score:

$$\mathrm{score}(\ell, i, \boldsymbol{E}) = \mathop{\mathbb{E}}_{((\boldsymbol{x}_1,\boldsymbol{y}_1),(\boldsymbol{x}_2,\boldsymbol{y}_2))\in \boldsymbol{E}_i}\left[\mathbb{1}[\boldsymbol{x}_1 \text{ and } \boldsymbol{x}_2 \text{ reach } \ell]\cdot \mathbb{1}[\boldsymbol{y}_1 \neq \boldsymbol{y}_2]\right]$$

If we desire there to be m samples in each $\boldsymbol{E}_i$ with probability at least $1-\delta$, then having $\boldsymbol{E}$ by size $O(n\cdot(m+\log(\frac{1}{\delta}))$ is sufficient, where m is as defined in Equation (11). Since one can certainly general a random edge sample $\boldsymbol{E}$ if given membership query access to f, this proves Theorem 9.

Learning trees with small average depth: Theorem 11

Let f be a monotone function computed by a decision tree T of average depth $\triangle$.

1. We first observe that the total influence of f is at most $\triangle$. To see this, first recall that $\mathrm{Inf}(f) = \mathbb{E}[\mathrm{sens}_f(\boldsymbol{x})]$, where $\mathrm{sens}_f(x) = |\{i\in[n]\colon f(x)\neq f(x^{\oplus i})\}|$, i.e. that total influence is equivalent to average sensitivity. For any x, the sensitivity of f at x is at most the depth of the path that x follows in T, and hence the average sensitivity of f is at most the average depth $\triangle$ of T.
2. Recall Theorem 18, which says that monotone functions with decision tree size s have total influence at most $\sqrt{\log s}$. In fact, [53] proves a stronger statement: if f is monotone, then it has total influence at most $\sqrt{\triangle}$. (This is indeed a stronger statement because $\triangle \leq \log s$.)
3. Similarly, [52] also establishes a stronger version of Theorem 14, showing that f has a variable of influence at least $\mathrm{Var}(f)/\triangle$ (rather than just $\mathrm{Var}(f)/\log s$). Hence an equivalent statement to Lemma 15 holds, where BuildTopDownDT selects a leaf with score at least $\frac{\varepsilon}{(j+1)\triangle}$.

Combining these observations, with the same proof as for Theorem 6, we conclude that BuildTopDownDT produces a tree of size $2^{O(\triangle^2/\varepsilon)}$, and if T is monotone, size only $2^{O(\triangle^{3/2}/\varepsilon)}$. Then, for the same reasons as Theorems 9 and 10 hold, Theorem 11 holds.

9 Proper learning with polynomial sample and memory complexity

In this section we give a quasipolynomial-time algorithm for properly learning decision trees under the uniform distribution, where sample and memory of our algorithm are both polynomial. To our knowledge, this is the first algorithm for properly learning decision trees that achieves polynomial memory complexity. (Recall Table 1.)

Background: Ehrehfeucht–Haussler and Mehta–Raghavan

At the core of most learning algorithms is an algorithm that achieves low error on a set of samples. We will use the following notation in this section:

Notation

A sample, S, is a set of examples of the form (x, y) where $x \in \{0,1\}^n$ and $y \in \{-1, 1\}$. The error of a decision tree, T, with respect to the samples is

$$\text{error}_S(T) = \Pr_{\boldsymbol{x},\boldsymbol{y} \in S}[T(\boldsymbol{x}) \neq \boldsymbol{y}].$$

We say that a set of samples is *exactly fit* by a tree of size s if there exists a zero-error tree with size at most s. Furthermore, we use S_0^v and S_1^v to refer to all the points in the sample S where the variable corresponding to v is 0 and 1 respectively. Lastly, all learning statements in this section are with respect to the uniform distribution.

Ehrenfeucht and Haussler's algorithm makes the following guarantee:

▶ **Theorem 44** (Algorithmic core of [21]). *There is an algorithm that takes in a set of samples, S, over n variables that can be exactly fit by a decision tree of size s and returns a tree of size at most $n^{\log(s)}$ that exactly fits S. Furthermore, that algorithm runs in time $|S| \cdot n^{O(\log s)}$.*

One downside of their algorithm is that it leads to a large hypothesis class – the class of all decision trees of size $n^{\log s}$ – so in order to generalize with high probability, they require $\text{poly}(n^{\log s}, \frac{1}{\varepsilon})$ samples.

Mehta and Raghavan observe that if a function is computable by a tree of size s, then it is also ε-approximated by a tree of depth at most $\log(\frac{s}{\varepsilon})$. They combine this observation with a new algorithm that makes the following guarantee:

▶ **Theorem 45** (Algorithmic core of [46]). *There is an algorithm that takes a sample, S, over n variables as well as budgets for size s and depth d and returns the decision tree of size at most s and depth at most d with minimal error on S.*[19] *Furthermore, the algorithm runs in time $n^{O(d)} \cdot (s^2 + |S|)$.*

Importantly, there are only $2 \cdot (4n)^s$ decision trees of size at most s, a much smaller hypothesis class than for Ehrenfeucht and Haussler's algorithm. As a result, they only need $\text{poly}(s, \frac{1}{\varepsilon}) \cdot \log n$ samples to generalize with high probability. A downside of their work, relative to Ehrenfeucht and Haussler's, is that they need to set $d = O(\log(\frac{s}{\varepsilon}))$, so their algorithm has a runtime of approximately $n^{O(\log(s/\varepsilon))}$ instead of $n^{O(\log s)}$.

Neither [21] nor [46] are able to learn decision trees with only $\text{poly}(n, s, \frac{1}{\varepsilon})$ memory. [21] uses a sample of size approximately $n^{O(\log s)}$ to guarantee generalization, and their algorithm must store all of the samples, so it needs at least that much memory. [46] use a dynamic programming algorithm that stores computation for each restriction of the n variables of size at most $d = O(\log(\frac{s}{\varepsilon}))$. There are $\binom{n}{d} \cdot 2^d = \Theta\binom{n}{\log(s/\varepsilon)}$ such restrictions, resulting in superpolynomial memory complexity.

[19] They also guarantee that if their are multiple trees with minimal error, they return a tree with minimal size among those with minimal error.

Algorithm 1 Our variant of Ehrenfeucht and Haussler's FIND algorithm.

FIND(S, s, d):

Input: A sample S that can be exactly fit by a tree of size s and depth budget d.

Output: A decision tree T with depth at most d that approximately fits S. If S cannot be fit by a tree of size s, may return "None".

1. If all samples in S have the same label, return the single-node tree computing that label.
2. If $s \leq 1$ return "None".
3. If $d = 0$ return the single-node tree computing the majority label of S.
4. For each relevant[20] variable v:
   a. Let $T_0^v = \text{FIND}(S_0^v, \frac{s}{2}, d-1)$ and $T_1^v = \text{FIND}(S_1^v, \frac{s}{2}, d-1)$.
   b. If both T_0^v and T_1^v are not "None", return the tree with root labeled v, 0-subtree T_0^v and 1-subtree T_1^v.
   c. If one of T_0^v and T_1^v is "None" and the other is not:
      i. Reexecute the recursive call for the side that was "None" with size $s-1$ instead of size $\frac{s}{2}$. For example, if T_1^v was "None", set $T_1^v = \text{FIND}(S_1^v, s-1, d-1)$
      ii. If the reexecuted call still returns "None", return "None".
      iii. Return the tree with root labeled v, 0-subtree T_0^v and 1-subtree T_1^v.
5. Return "None".

Our algorithm: proper learning with polynomial sample and memory complexity

We introduce a new algorithm that makes more assumptions about its input than either [21]'s or [46]'s algorithms. It requires the samples it receives to be *well-distributed*, a property we will later define (Definition 48). In exchange, it only uses polynomial memory. The following should be contrasted with Theorems 44 and 45:

▶ **Theorem 46** (Core of our algorithm). *There is an algorithm (Algorithm 1) that when given a depth budget d and a well-distributed sample S that can be exactly fit by a tree of size s returns a tree with depth at most d and error at most $(\frac{3}{4} + o(1))^d \cdot s$ on the samples. Furthermore, the algorithm runs in time $|S| \cdot n^{O(\log s)}$ and uses $\text{poly}(2^d, \log n, |S|)$ memory.*

Note that if the goal is to learn the sample to accuracy ε, we can set $d = O(\log(\frac{s}{\varepsilon}))$. The result is an algorithm that runs in time $|S| \cdot n^{O(\log s)}$ and uses memory $\text{poly}(n, s, 1/\varepsilon, |S|)$. Furthermore, the well-distributed requirement turns out to be true for nearly all uniformly random samples that are sufficiently large. The result is, to the best of our knowledge, the first polynomial memory proper learning algorithm for decision trees.

Our algorithm (Algorithm 1) is a surprisingly simple modification of [21]'s algorithm, but our analysis is quite a bit more involved. A key difference is that [21]'s algorithm is an Occam algorithm, whereas ours is not. The original [21] algorithm breaks down when in cannot fully fit the sample; the analysis showing that our algorithm *is* able to handle a sample it cannot fully fit is subtle.

▶ **Lemma 47** (Correctness of FIND). *If S can be exactly fit by a tree of size s, then* FIND(S, s, d) *will not return "None".*

Proof. By induction. If $s = 1$ and S can be fit by a tree of size s, then all samples in S will have the same label. Hence, FIND will return a tree on line 1, and not return "None".

For $s \geq 2$, there are only two spots where FIND could return "None":

Line 4.c.ii FIND returns "None" on line 4.c.ii only if a call of the form $\text{FIND}(S_a^v, s-1, d-1)$ returns "None" where $a = \pm 1$ and v is relevant. Let T_S be a minimal size tree that fits S, which by assumption, has size at most s. Since v is relevant, a node labeled with it must appear somewhere in that tree. This means that there is a size $s-1$ tree that will exactly fit S_a^v. By induction, this means that $\text{FIND}(S_a^v, s-1, d-1)$ will not return "None".

Line 5. FIND returns "None" on line 5 only if there was not a single relevant variable v for which either of the calls $\text{FIND}(S_0^v, \frac{s}{2}, d-1)$ or $\text{FIND}(S_1^v, \frac{s}{2}, d-1)$ on line 4.a succeeded (i.e. did not return "None"). Once again, let T_S be a minimal size tree that fits S. Then, every node in T_S must be relevant for S. Furthermore, T_S has some root variable v^* and subtrees $(T_S)_0$ and $(T_S)_1$. At least one of $(T_S)_0$ or $(T_S)_1$ must have size at most $\frac{s}{2}$. If $(T_S)_0$ has size at most $\frac{s}{2}$, then by the inductive hypothesis, $\text{FIND}(S_0^v, \frac{s}{2}, d-1)$ does not return "None". Otherwise $\text{FIND}(S_1^v, \frac{s}{2}, d-1)$ does not return "None". Hence, FIND won't return "None" on line 5. ◀

We hope to prove that FIND will produce low error trees, but it turns out to be difficult to guarantee this for arbitrary samples. One particular sample we can guarantee this for is the sample containing all possible points. If S contains all 2^n possible points, then FIND will return a tree with error at most $\frac{1}{4} \cdot (\frac{3}{4})^d$, which we will prove in Lemma 49. The following property allows us to quantify how close S is to the full sample.

▶ **Definition 48** (Well-distributed samples). *We say that a sample of points S is c-well-distributed to depth d if, for any restriction α where $|\alpha| \leq d$,*

$$||(S_\alpha)| - \mu| \leq c\mu$$

where $\mu = 2^{-|\alpha|} \cdot |\boldsymbol{S}|$ is the expected size of $\boldsymbol{S}_\alpha$ if $\boldsymbol{S}$ is chosen uniformly at random.

For example, if S contains all possible 2^n points, then S is 0-*well-distributed* to any depth.

▶ **Lemma 49** (Error of FIND on well-distributed samples). *Let S be c-well-distributed to depth d. If $\text{FIND}(S, s, d)$ does not return "None", it returns a tree with error at most $\frac{1}{4}(\frac{3}{4} + \frac{c}{4})^d \cdot s$ with respect to S.*

Proof. By induction on the d; If $d = 0$ and $s \geq 2$, then this lemma requires the error to be less than $\frac{1}{2}$, which FIND satisfies since it places the majority node. If $s = 1$ and FIND doesn't return "None", it must have returned a zero-error tree on Line 1, satisfying the desired error bound.

Next, consider $d \geq 1$. If all samples have the same label, FIND returns a 0 error tree. Otherwise, it returns a tree, T, with 0-subtree T_0^v and 1-subtree T_1^v for some variable v. Let ℓ_0 and ℓ_1 be the number of points in S_0^v and S_1^v respectively. Then, we can relate the errors of the trees as follows:

$$\text{error}(T) = \frac{\ell_0}{\ell_0 + \ell_1} \cdot \text{error}(T_0^v) + \frac{\ell_1}{\ell_0 + \ell_1} \cdot \text{error}(T_1^v)$$

At least one of T_0^v and T_1^v was generated using a recursive call to FIND with size parameter $\frac{s}{2}$. Without loss of generality, let that tree be T_0^v. The other tree, T_1^v was generated by a recursive call with size at most s. By the inductive hypothesis,

$$\text{error}(T) \leq \frac{\ell_0}{\ell_0 + \ell_1} \cdot \frac{1}{4}\left(\frac{3}{4} + \frac{c}{4}\right)^d \cdot \frac{s}{2} + \frac{\ell_1}{\ell_0 + \ell_1} \cdot \frac{1}{4}\left(\frac{3}{4} + \frac{c}{4}\right)^d \cdot s \tag{12}$$

Since S is c-well-distributed to depth d, $(1-c)\mu \leq \ell_0, \ell_1 \leq (1+c)\mu$, where $\mu = \frac{|S|}{2}$. Choosing $\ell_0 = (1-c)\mu$ and $\ell_1 = (1+c)\mu$ maximizes equation Equation (12) and so results in a valid upper bound.

$$\begin{aligned}\text{error}(T) &\leq \frac{\mu(1-c)}{2\mu} \cdot \frac{1}{4}\left(\frac{3}{4}+\frac{c}{4}\right)^{d-1} \cdot \frac{s}{2} + \frac{\mu(1+c)}{2\mu} \cdot \frac{1}{4}\left(\frac{3}{4}+\frac{c}{4}\right)^{d-1} \cdot s \\ &= \frac{1}{4} \cdot \left(\frac{3}{4}+\frac{c}{4}\right)^{d-1} \cdot \left(\frac{3}{4}+\frac{c}{4}\right) \cdot s \\ &= \frac{1}{4}\left(\frac{3}{4}+\frac{c}{4}\right)^{d} \cdot s\end{aligned}$$

◀

The above Lemma shows that if a sample is sufficiently well-distributed, FIND will return a tree with low error. We next show that, with high probability, sufficiently large samples will be well-distributed.

▶ **Lemma 50** (Well-distributed samples are common). *Choose any* $0 < c < 1.0, \delta > 0$. *Then for*

$$m = O\left(\frac{2^d}{c^2} \cdot (d\log(n) + \log(1/\delta))\right),$$

a sample of size m *chosen uniformly at random from* $\{0,1\}^n$ *is* c*-well-distributed to depth* d *with probability at least* $1-\delta$

Proof. Consider an arbitrary restriction α of length $h \leq d$. By Chernoff bounds,

$$\Pr[||\boldsymbol{S}_\alpha| - \mu| \geq c\mu] \leq 2e^{-\mu c^2/3}$$

where $\mu = \mathbb{E}[|\boldsymbol{S}_\alpha|] = m \cdot 2^{-h}$. Since $h \leq d$, we can upper bound the probability as follows.

$$\Pr[||\boldsymbol{S}_\alpha| - \mu| \geq c\mu] \leq 2e^{-m\cdot 2^{-d}c^2/3}$$

$\boldsymbol{S}$ is c-well-distributed if $||\boldsymbol{S}_\alpha| - \mu| \leq c\mu$ for all possible restrictions α of size at most d. There are $\sum_{i=0}^{d}\binom{n}{i}2^i = n^{O(d)}$ such restrictions. Thus, by a union bound:

$$\Pr[\boldsymbol{S} \text{ is } c\text{-well-distributed}] \geq 1 - n^{O(d)} \cdot e^{-m\cdot 2^{-d}c^2/3}.$$

We set the right-hand side of the above equation to be at least $1-\delta$ and solve for m, which proves this Lemma. ◀

Our analysis of the time complexity of FIND is very similar to Ehrenfeucht and Haussler's:

▶ **Lemma 51** (Time complexity of FIND). FIND(S,s,d) *takes time* $|S| \cdot (n+1)^{2\log(s)}$.

Proof. Fix a total number of variables n and sample size m. Let $T(i,s)$ be the maximum time needed by FIND(S,s,d) where S has size at most m, i is the number of relevant variables in S, and d is arbitrary.

If $i = 0$ or $s = 1$, then FIND must return on Line 1 or 2, using only $O(m)$ time. Otherwise, the FIND makes at most $2i$ recursive calls on line 4.(a) each of which takes time at most $T(i-1, \frac{s}{2})$ It also makes zero or one recursive call on line 4.(c).i which takes time up

to $T(i-1, s-1) \leq T(i-1, s)$. In addition to these recursive calls, all of the auxiliary computations can be done in $O(mn)$ time. Hence, we have the following recurrence relation:

$$T(i, s) \leq 2i \cdot T\left(i-1, \tfrac{s}{2}\right) + T(i-1, s) + O(mn).$$

If we substitute $r = \log(s)$, then equivalently, we have the relation:

$$\tilde{T}(i, r) \leq 2i \cdot \tilde{T}(i-1, r-1) + \tilde{T}(i-1, r) + O(mn).$$

The above relation is shown to be upper bounded by $\tilde{T}(i, r) = O(m \cdot (n+1)^{2r})$ in [21]. Substituting back $r = \log(s)$ gives that $T(i, s) \leq O(m \cdot (n+1)^{2\log(s)})$. ◀

▶ **Lemma 52** (Memory complexity of FIND). FIND(S, s, d) *takes memory* $O(2^d(|S| + \log n))$.

Proof. Each call to FIND with depth d will only ever need simultaneously need to run up to 2 calls to FIND, each with depth $d-1$. Hence, there are at most 2^d copies of FIND that need to be stored in memory at any one time. At worst, each copy stores the sample as well as pointers to it and the n different variables. This means each copy uses $O(|S| + \log n)$ memory, for a total of $O(2^d(|S| + \log n))$ memory. ◀

The last step in this analysis is a standard generalization argument relying on Chernoff bounds.

▶ **Lemma 53** (Generalization). *Choose any $\delta, \varepsilon \geq 0$. Suppose that $\boldsymbol{S}$ is a uniformly random sample from a function, f, that can be computed by a decision tree of size at most s and depth at most d. If the number of samples in $\boldsymbol{S}$ is at least*

$$m = O\left(\frac{2^d \log(n) + \log(\frac{1}{\delta})}{\varepsilon}\right)$$

and FIND$(\boldsymbol{S}, s, d)$ *returns a decision tree that fits $\boldsymbol{S}$ with error at most $\frac{\varepsilon}{4}$. Then, with probability at least $1-\delta$, the decision tree returned by* FIND *has error at most ε on f.*

Proof. We will upper bound the number of different decision trees FIND could return when given depth limit d. There up to 2^d spots where a decision tree of depth $\leq d$ could have a node. In each of these spots, the decision tree could either have one of n variables, a leaf that is either $+1$ or -1, or nothing. Thus, the number of decision trees of depth at most d is at most $(n+3)^{2^d}$.

We call a decision tree, T, "bad", if T has error at least ε on f. For any particular bad tree T, the probability it will have error less than $\frac{\varepsilon}{4}$ on $\boldsymbol{S}$ can be upper bounded using a Chernoff Bound:

$$\Pr_{\boldsymbol{S}}\left[\text{error}_{\boldsymbol{S}}(T) \leq \tfrac{\varepsilon}{4}\right] \leq \exp(-\tfrac{9}{32}\varepsilon m).$$

Since there are most $(n+3)^{2^d}$ bad trees, the probability that any bad tree has error at most $\frac{\varepsilon}{4}$ is at most $(n+3)^{2^d} \cdot \exp(-\frac{9}{32}\varepsilon m)$. Setting this equal to δ and solving for m completes the proof of this lemma. ◀

Finally, we are able to put all these pieces together to prove our main theorem of this section, and show how FIND is used to learn decision trees:

▶ **Theorem 54** (Proper learning with polynomial sample and memory complexity)**.** *Let f be any function over n variables computable by a size s decision tree. Choose any $\varepsilon, \delta > 0$. There is an algorithm that*

- *runs in time* $\text{poly}(n^{\log s}, \frac{1}{\varepsilon}, \log(\frac{1}{\delta}))$
- *requires memory* $\text{poly}(s, \log n, \frac{1}{\varepsilon}, \log(\frac{1}{\delta}))$
- *uses* $\text{poly}(s, \log n, \frac{1}{\varepsilon}, \log(\frac{1}{\delta}))$ *random samples from f*

and with probability at least $1 - \delta$ returns a decision tree that is an ε-approximation of f.

Proof. Choose any constant $0 < c < 1$ and set $d = \log(\frac{s}{\varepsilon})/(-\log(\frac{3+c}{4}))$. Then, by taking a uniformly random sample, $\boldsymbol{S}$, of size

$$m = O\left(\frac{2^d}{\varepsilon c^2} \cdot (d \log(n) + \log(1/\delta))\right)$$

we have the following holds:

1. $\boldsymbol{S}$ is c-well-distributed with probability at least $1 - \frac{\delta}{2}$.
2. If $\boldsymbol{S}$ is c-well-distributed, then $\textsc{Find}(\boldsymbol{S}, s, d)$ returns a tree, T, with error at most $\frac{1}{4}(\frac{3+c}{4})^d \cdot s = \frac{\varepsilon}{4}$ on $\boldsymbol{S}$.
3. If T has error less than $\frac{\varepsilon}{4}$, then with probability at least $1 - \frac{\delta}{2}$, T is a ε-approximation for f.

Furthermore, this procedure meets the time constraints by Lemma 51 and memory constraints by Lemma 52. ◀

References

1. Scott Aaronson and Andris Ambainis. The Need for Structure in Quantum Speedups. *Theory of Computing*, 10(6):133–166, 2014.
2. Misha Alekhnovich, Mark Braverman, Vitaly Feldman, Adam Klivans, and Toniann Pitassi. The complexity of properly learning simple concept classes. *Journal of Computer & System Sciences*, 74(1):16–34, 2009.
3. Pranjal Awasthi, Vitaly Feldman, and Varun Kanade. Learning Using Local Membership Queries. In *Proceedings of the 26th Annual Conference on Learning Theory (COLT)*, pages 398–431, 2013.
4. Paul Beame, Shayan Oveis Gharan, and Xin Yang. Time-Space Tradeoffs for Learning Finite Functions from Random Evaluations, with Applications to Polynomials. In *Proceedings of the 31st Conference On Learning Theory (COLT)*, volume 75, pages 843–856, 2018.
5. Eric Blais, Clément Canonne, Igor Oliveira, Rocco Servedio, and Li-Yang Tan. Learning circuits with few negations. In *Proceedings of the 18th International Workshop on Randomization and Computation (RANDOM)*, pages 512–527, 2015.
6. Eric Blais and Li-Yang Tan. Approximating Boolean functions with depth-2 circuits. *SIAM J. Comput.*, 44(6):1583–1600, 2015.
7. Avirm Blum, Merrick Furst, Jeffrey Jackson, Michael Kearns, Yishay Mansour, and Steven Rudich. Weakly learning DNF and characterizing statistical query learning using Fourier analysis. In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing (STOC)*, pages 253–262, 1994.
8. Avrim Blum. Rank-r decision trees are a subclass of r-decision lists. *Inform. Process. Lett.*, 42(4):183–185, 1992. doi:10.1016/0020-0190(92)90237-P.
9. Avrim Blum, Carl Burch, and John Langford. On Learning Monotone Boolean Functions. In *Proceedings of the 39th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 408–415, 1998.
10. Avrim Blum and Pat Langley. Selection of Relevant Features and Examples in Machine Learning. *Artificial Intelligence*, 97(1-2):245–271, 1997.

11 Leo Breiman. *Classification and regression trees*. Routledge, 2017.

12 Alon Brutzkus, Amit Daniely, and Eran Malach. ID3 Learns Juntas for Smoothed Product Distributions. *ArXiv*, abs/1906.08654, 2019. arXiv:1906.08654.

13 Alon Brutzkus, Amit Daniely, and Eran Malach. On the Optimality of Trees Generated by ID3. *ArXiv*, abs/1907.05444, 2019. arXiv:1907.05444.

14 Nader Bshouty. Exact learning via the monotone theory. *Information and Computation*, 123(1):146–153, 1995.

15 Nader Bshouty and Christino Tamon. On the Fourier spectrum of monotone functions. *Journal of the ACM*, 43(4):747–770, 1996.

16 Nader H. Bshouty, Elchanan Mossel, Ryan O'Donnell, and Rocco A. Servedio. Learning DNF from random walks. *J. Comput. System Sci.*, 71(3):250–265, 2005.

17 Sitan Chen and Ankur Moitra. Beyond the low-degree algorithm: mixtures of subcubes and their applications. In *Proceedings of the 51st Annual ACM Symposium on Theory of Computing (STOC)*, pages 869–880, 2019.

18 Dana Dachman-Soled, Homin K. Lee, Tal Malkin, Rocco A. Servedio, Andrew Wan, and Hoeteck Wee. Optimal Cryptographic Hardness of Learning Monotone Functions. *Theory of Computing*, 5(13):257–282, 2009. doi:10.4086/toc.2009.v005a013.

19 Ilias Diakonikolas, Prahladh Harsha, Adam Klivans, Raghu Meka, Prasad Raghavendra, Rocco Servedio, and Li-Yang Tan. Bounding the average sensitivity and noise sensitivity of polynomial threshold functions. In *Proceedings of the 42nd Annual Symposium on Theory of Computing (STOC)*, pages 533–542, 2010.

20 Tom Dietterich, Michael Kearns, and Yishay Mansour. Applying the weak learning framework to understand and improve C4.5. In *Proceedings of the 13th International Conference on Machine Learning (ICML)*, pages 96–104, 1996.

21 Andrzej Ehrenfeucht and David Haussler. Learning decision trees from random examples. *Information and Computation*, 82(3):231–246, 1989.

22 Vitaly Feldman. Hardness of Proper Learning. In *Encyclopedia of Algorithms*, 2016.

23 Amos Fiat and Dmitry Pechyony. Decision trees: More theoretical justification for practical algorithms. In *Proceedings of the 15th International Conference on Algorithmic Learning Theory (ALT)*, pages 156–170, 2004.

24 Ehud Friedgut. Boolean functions with low average sensitivity depend on few coordinates. *Combinatorica*, 18(1):474–483, 1998.

25 Ehud Friedgut and Gil Kalai. Every Monotone Graph Property has a Sharp Threshold. *Proceedings of the American Mathematical Society*, 124:2993–3002, 1996.

26 Sumegha Garg, Ran Raz, and Avishay Tal. Extractor-based Time-space Lower Bounds for Learning. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 990–1002, 2018.

27 Sumegha Garg, Ran Raz, and Avishay Tal. Time-Space Lower Bounds for Two-Pass Learning. In *Proceedings of the 34th Computational Complexity Conference (CCC)*, pages 22:1–22:39, 2019.

28 Oded Goldreich, Shafi Goldwasser, and Dana Ron. Property testing and its connection to learning and approximation. *Journal of the ACM*, 45:653–750, 1998.

29 Oded Goldreich and Leonid Levin. A hard-core predicate for all one-way functions. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing (STOC)*, pages 25–32, 1989.

30 Parikshit Gopalan, Adam Kalai, and Adam Klivans. Agnostically learning decision trees. In *Proceedings of the 40th ACM Symposium on Theory of Computing (STOC)*, pages 527–536, 2008.

31 Parikshit Gopalan and Rocco Servedio. Learning and lower bounds for AC^0 with threshold gates. In *Proceedings of the 14th International Workshop on Randomization and Computation (RANDOM)*, pages 588–601, 2010.

32 Thomas Hancock and Yishay Mansour. Learning monotone k-μ DNF formulas on product distributions. In *Proceedings of the 4th Annual Conference on Computational Learning Theory (COLT)*, pages 179–193, 1991.

33 Elad Hazan, Adam Klivans, and Yang Yuan. Hyperparameter Optimization: A Spectral Approach. *Proceedings of the 6th International Conference on Learning Representations (ICLR)*, 2018.

34 Jeffrey Jackson, Homin Lee, Rocco Servedio, and Andrew Wan. Learning Random Monotone DNF. *Discrete Applied Mathematics*, 159(5):259–271, 2011.

35 Jeff Kahn, Gil Kalai, and Nathan Linial. The influence of variables on boolean functions. In *Proceedings of the 29th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 68–80, 1988.

36 Daniel Kane. The average sensitivity of an intersection of halfspaces. In *Proceedings of the 42nd ACM Symposium on Theory of Computing (STOC)*, pages 437–440, 2014.

37 Daniel Kane. The correct exponent for the Gotsman–Linial conjecture. *Computational Complexity*, 23(2):151–175, 2014.

38 Michael Kearns. Boosting theory towards practice: recent developments in decision tree induction and the weak learning framework (invited talk). In *Proceedings of the 13th National Conference on Artificial intelligence (AAAI)*, pages 1337–1339, 1996.

39 Michael Kearns, Ming Li, and Leslie Valiant. Learning Boolean formulas. *Journal of the ACM*, 41(6):1298–1328, 1994.

40 Michael Kearns and Yishay Mansour. On the boosting ability of top-down decision tree learning algorithms. *Journal of Computer and System Sciences*, 58(1):109–128, 1999.

41 Michael Kearns and Leslie Valiant. Cryptographic limitations on learning Boolean formulae and finite automata. *Journal of the ACM*, 41(1):67–95, 1994.

42 Gillat Kol, Ran Raz, and Avishay Tal. Time-space Hardness of Learning Sparse Parities. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 1067–1080, 2017.

43 Eyal Kushilevitz and Yishay Mansour. Learning Decision Trees Using the Fourier Spectrum. *SIAM Journal on Computing*, 22(6):1331–1348, December 1993.

44 Homin Lee. *On the learnability of monotone functions*. PhD thesis, Columbia University, 2009.

45 Nathan Linial, Yishay Mansour, and Noam Nisan. Constant depth circuits, Fourier transform and learnability. *Journal of the ACM*, 40(3):607–620, 1993.

46 Dinesh Mehta and Vijay Raghavan. Decision tree approximations of Boolean functions. *Theoretical Computer Science*, 270(1-2):609–623, 2002.

47 Dana Moshkovitz and Michal Moshkovitz. Mixing Implies Lower Bounds for Space Bounded Learning. In *Proceedings of the 30th Conference on Learning Theory (COLT)*, pages 1516–1566, 2017.

48 Dana Moshkovitz and Michal Moshkovitz. Entropy Samplers and Strong Generic Lower Bounds For Space Bounded Learning. In *Proceedings of the 9th Innovations in Theoretical Computer Science Conference (ITCS)*, pages 28:1–28:20, 2018.

49 Elchanan Mossel, Ryan O'Donnell, and Rocco A. Servedio. Learning functions of k relevant variables. *Journal of Computer and System Sciences*, 69(3):421–434, 2004.

50 Elchannan Mossel, Ryan O'Donnell, and Krzysztof Oleszkiewicz. Noise stability of functions with low influences: Invariance and optimality. *Annals of Mathematics*, 171:295–341, 2010.

51 Ryan O'Donnell. *Analysis of Boolean Functions*. Cambridge University Press, 2014. Available at `http://analysisofbooleanfunctions.net/`.

52 Ryan O'Donnell, Michael Saks, Oded Schramm, and Rocco Servedio. Every Decision Tree Has an Influential Variable. In *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 31–39, 2005.

53 Ryan O'Donnell and Rocco Servedio. Learning monotone decision trees in polynomial time. *SIAM Journal on Computing*, 37(3):827–844, 2007.

54 Ryan O'Donnell and Karl Wimmer. KKL, Kruskal–Katona, and Monotone Nets. *SIAM Journal on Computing*, 42(6):2375–2399, 2013.

55 Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

56 Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

57 Ran Raz. A Time-Space Lower Bound for a Large Class of Learning Problems. In *Proceedings of the 58th IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 732–742, 2017.

58 Ran Raz. Fast Learning Requires Good Memory: A Time-Space Lower Bound for Parity Learning. *Journal of the ACM*, 66(1):3:1–3:18, December 2018.

59 Ronald Rivest. Learning decision lists. *Machine learning*, 2(3):229–246, 1987.

60 Yoshifumi Sakai and Akira Maruoka. Learning monotone log-term DNF formulas under the uniform distribution. *Theory of Computing Systems*, 33:17–33, 2000.

61 Dominik Scheder and Li-Yang Tan. On the Average Sensitivity and Density of k-CNF Formulas. In *Proceedings of the 17th International Workshop on Randomization and Computation (RANDOM)*, pages 683–698, 2013.

62 Linda Sellie. Learning random monotone DNF under the uniform distribution. In *Proceedings of the 21st Annual Conference on Learning Theory (COLT)*, pages 181–192, 2008.

63 Rocco Servedio. On learning monotone DNF under product distributions. *Information and Computation*, 193(1):57–74, 2004.

64 Ohad Shamir. Fundamental limits of online and distributed algorithms for statistical learning and estimation. In *Proceedings of the 28th Conference on Neural Information Processing Systems*, pages 163–171, 2014.

65 Jacob Steinhardt, Gregory Valiant, and Stefan Wager. Memory, communication, and statistical queries. In *Proceedings of the 29th Annual Conference on Learning Theory (COLT)*, pages 1490–1516, 2016.

66 Karsten Verbeurgt. Learning sub-classes of monotone DNF on the uniform distribution. In *Proceedings of the 9th Conference on Algorithmic Learning Theory (ALT)*, pages 385–399, 1998.

67 Ian Witten, Eibe Frank, Mark Hall, and Christopher Pal. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.

Algorithms and Adaptivity Gaps for Stochastic k-TSP

Haotian Jiang
Paul G. Allen School of Computer Science & Engineering, University of Washington, Seattle, USA
jhtdavid@cs.washington.edu

Jian Li
Institute for Interdisciplinary Information Sciences, Tsinghua University, Beijing, China
lijian83@mail.tsinghua.edu.cn

Daogao Liu
Department of Physics, Tsinghua University, Beijing, China
liudg16@mails.tsinghua.edu.cn

Sahil Singla
Princeton University and Institute for Advanced Study, Princeton, USA
singla@cs.princeton.edu

Abstract

Given a metric (V, d) and a root $\in V$, the classic k-TSP problem is to find a tour originating at the root of minimum length that visits at least k nodes in V. In this work, motivated by applications where the input to an optimization problem is uncertain, we study two stochastic versions of k-TSP.

In Stoch-Reward k-TSP, originally defined by Ene-Nagarajan-Saket [13], each vertex v in the given metric (V, d) contains a stochastic reward R_v. The goal is to adaptively find a tour of minimum *expected* length that collects at least reward k; here "adaptively" means our next decision may depend on previous outcomes. Ene et al. give an $O(\log k)$-approximation adaptive algorithm for this problem, and left open if there is an $O(1)$-approximation algorithm. We totally resolve their open question, and even give an $O(1)$-approximation *non-adaptive* algorithm for Stoch-Reward k-TSP.

We also introduce and obtain similar results for the Stoch-Cost k-TSP problem. In this problem each vertex v has a stochastic cost C_v, and the goal is to visit and select at least k vertices to minimize the expected *sum* of tour length and cost of selected vertices. Besides being a natural stochastic generalization of k-TSP, this problem is also interesting because it generalizes the Price of Information framework [33] from deterministic probing costs to metric probing costs.

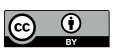
Our techniques are based on two crucial ideas: "repetitions" and "critical scaling". In general, replacing a random variable with its expectation leads to very poor results. We show that for our problems, if we truncate the random variables at an ideal threshold, then their expected values form a good surrogate. Here, we rely on running several repetitions of our algorithm with the same threshold, and then argue concentration using Freedman's and Jogdeo-Samuels' inequalities. Unfortunately, this ideal threshold depends on how far we are from achieving our target k, which a non-adaptive algorithm does not know. To overcome this barrier, we truncate the random variables at various different scales and identify a "critical" scale.

2012 ACM Subject Classification Theory of computation → Stochastic control and optimization; Theory of computation → Approximation algorithms analysis

Keywords and phrases approximation algorithms, stochastic optimization, travelling salesman problem

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.45

Acknowledgements We are thankful to the anonymous reviewers of ITCS 2020 for many helpful comments on the presentation of this paper. Haotian Jiang is supported in part by NSF awards CCF-1749609, CCF-1740551 and DMS-1839116. Jian Li and Daogao Liu are supported in part by the National Natural Science Foundation of China Grant 61822203, 61772297, 61632016, 61761146003, and the Zhongguancun Haihua Institute for Frontier Information Technology and Turing AI Institute of Nanjing. Sahil Singla is supported in part by the Schmidt Foundation.



1 Introduction

Consider a scenario where a salesperson must sell some quota of brushes in order to win a trip to Hawaii. The salesperson knows the time it takes to travel between different cities and the demand at each city. What is the best route to take to sell the quota while spending the least amount of time? This exact scenario was described by Awerbuch et al. [4] to motivate the study of TSP problems where the algorithm has to also decide which cities to visit. A cleaner version of this problem, first introduced by Ravi et al. [31], is the k-TSP problem where we assume that each city has a unit demand. Formally, given a metric (V, d) with a root $\in V$ and a *target* $k \in \mathbb{Z}_{\geq 0}$, the k-TSP problem is to find a tour that originates at the root and visits at least k vertices, while minimizing the total travel time. There is a long line of work trying to design better approximation algorithms for the k-TSP problem [4, 30, 8, 16, 2, 1], and the state-of-the-art is a 2-approximation algorithm due to Garg [17][1].

In this work we consider stochastic versions of the k-TSP problem: what if the salesperson does not know the exact demand in each city, or what if the salesperson need to spend some uncertain time in each city to complete the city's demand? Indeed, there is a long line of work studying classical optimization problems where we begin only with estimates (probability distributions) on the input parameters. The algorithm has to *adaptively probe* parameters (inspect elements) by paying some "cost" before realizing their exact values; here "adaptively" means that our decisions may depend on the outcomes of already probed elements. Such stochastic probing problems have been well-studied in both maximization and minimization settings [11, 18, 19, 22, 20, 23, 5, 24, 6, 29, 3, 25, 13, 15, 33, 9, 21].

There are two natural ways of defining the stochastic k-TSP problem, depending on the type of input uncertainty. In the Stoch-Reward k-TSP problem, first introduced by Ene-Nagarajan-Saket [13], we incorporate uncertainty in the vertex demands. Formally, we assume that the demand at each vertex is drawn independently from a known distribution, and the goal is to *adaptively* find a tour Π that obtains the target demand k, while minimizing the total *expected* travel time[2].

We also study Stoch-Cost k-TSP where each city still has a unit demand, but the salesperson will have to spend an additional *completion* time (drawn from a known distribution) at each vertex before meeting its unit demand. The goal is to adaptively find a tour Π that completes the target demand k, while minimizing the expected *sum* of total travel and completion times. Notice, our algorithm finds the exact completion time of a vertex only after visiting it, and may choose not to complete it (i.e., not meet its unit demand) if the completion time seems too long. This idea of studying stochastic completion times at vertices is not new, and has been previously used in the study of stochastic Orienteering problems, which in some sense is the dual to our Stoch-Cost k-TSP problem [23, 6].

A common theme in the study of stochastic probing problems is to understand the power of adaptivity. Indeed, while the optimal algorithms can fully adapt to the outcomes, and hence may not even have a polynomial-size representation, a *non-adaptive* algorithm makes all its decisions upfront independent of the observed outcomes (except perhaps the stopping time). Being non-adaptive has several benefits like they are simpler to find, easily parallelizable, and have a poly-size representation. So ideally for a probing problem we would like to design non-adaptive algorithms with performance close to the optimal adaptive

[1] A closely-related variant is called the k-MST problem. Both problems are equivalent up to a constant approximation factor.

[2] Our Stoch-Reward k-TSP problem is called the "Stochastic k-TSP" problem in [13]. We rename it to differentiate it from Stoch-Cost k-TSP.

algorithms, or in other words design non-adaptive algorithms with a small *adaptivity gap*. The main results of our work is to show that both the above Stoch-Reward k-TSP and Stoch-Cost k-TSP problems have a constant adaptivity-gap. That is, there exist fixed-tours starting at the root which the algorithm can take until it obtains the target demand k, which guarantee an expected total time at most a constant factor more than the expected total time of the optimal adaptive tour. Moreover, for distributions with polynomial support, we give poly-time algorithms to find such tours.

Our constant adaptivity-gap for Stoch-Reward k-TSP answers the main open question of [13], who showed an $O(\log^2 k)$ bound on the adaptivity gap. The constant adaptivity gap result for Stoch-Cost k-TSP might also seem surprising because it is known that the related Stochastic Orienteering problem has a super-constant adaptivity gap [6].

In the rest of this section we first formally state our problems and results, and then discuss our high-level techniques and other related work.

1.1 Stoch-Reward k-TSP

The following Stoch-Reward k-TSP was first defined by Ene et al. [13].

Stoch-Reward k-TSP. We are given a metric (V, d) with a root $\in V$ and each vertex $v \in V$ has an independent integral[3] stochastic[4] reward $R_v \in \mathbb{Z}_{\geq 0}$. All reward distributions are given as input but the actual reward instantiation R_v is only known when the algorithm visits vertex v. Given a target value $k \in \mathbb{Z}_{\geq 0}$, the goal is to adaptively find a tour Π originating at root that collects at least k reward (i.e., $\sum_{v \in \Pi} R_v \geq k$) while minimizing the expected tour length.

This problem captures several well-studied problems; e.g., it captures the problem of Stoch-Knapsack Cover where the metric (V, d) is a weighted star: given a target k and n items where item $i \in [n]$ has both a deterministic cost $C_i \in \mathbb{R}_{\geq 0}$ and an independent stochastic reward $R_i \in \mathbb{Z}_{\geq 0}$, the Stoch-Knapsack Cover problem is to adaptively obtain a total reward of at least k at the minimum expected cost. This problem was studied by Deshpande et al. [12], and they gave an *adaptive* 2-approximation algorithm. However, even in this special case, it was not know if there is a non-adaptive constant factor approximation algorithm.

The first non-trivial results for the Stoch-Reward k-TSP problem were obtained by Ene et al. [13]. They gave an $O(\log^2 k)$-approximation non-adaptive algorithm and an $O(\log k)$-approximation adaptive algorithm. On the hardness side, however, they only gave a lower bound of e on the adaptivity gap. This left open closing the wide gap on the adaptivity gap for Stoch-Reward k-TSP. We resolve their open question by giving a non-adaptive $O(1)$-approximation algorithm.

▶ **Theorem 1.** *The Stoch-Reward k-TSP problem has a non-adaptive $O(1)$-approximation algorithm.*

The difficulty in Stoch-Reward k-TSP arises because the expected reward is a poor indicator of how much we care about a node. An extreme example is a vertex with a large expected reward, but which is non-zero with nearly zero probability. It is therefore reasonable to truncate the reward distributions at the remaining target reward. However, it is not clear why such an approach would work, and moreover this approach is adaptive as it depends on the remaining target.

[3] This is without loss of generality. Our result also generalizes to the case where rewards are real numbers via re-scaling.

[4] We assume that the distribution is discrete and is given explicitly.

1.2 Stoch-Cost k-TSP

We formally define the Stoch-Cost k-TSP problem.

Stoch-Cost k-TSP. We are given a metric (V, d) with a root $\in V$ and each vertex $v \in V$ has an independent stochastic cost $C_v \in \mathbb{R}_{\geq 0}$. All cost distributions are given as input but the actual cost instantiation C_v is only known when vertex v is visited. Suppose a vertex v can only be *selected* if: (1) v is visited and (2) we are currently[5] at vertex v. The goal is to adaptively find a tour Π originating at root that selects a set S of k visited vertices while minimizing the expected *total* cost, which is the sum of the tour length and the cost of the selected vertices:

$$\mathbb{E}\Big[d(\Pi) + \sum_{v \in S} C_v\Big].$$

Apart from being a natural generalization of the classical k-TSP problem, Stoch-Cost k-TSP is also motivated due to its connections to price of information [33]. In particular, it generalizes the Minimization k-Pandora's Box problem studied in [28, 33]. In this problem we are given a target k and n items, where each item $i \in [n]$ has a known probing price $\pi_i \in \mathbb{R}_{\geq 0}$ and an independent stochastic cost C_i. The exact cost C_i is only revealed after we pay price π_i. The goal is to adaptively probe and select k of the probed items to minimize the expected total selection cost plus probing price. Stoch-Cost k-TSP captures this problem on a star metric where node i is at a distance $\pi_i/2$ from the root. Thus, we can view the Stoch-Cost k-TSP problem as generalizing the price of information framework to a *metric* setting, where the price of probing is not fixed but given by a general metric. Our next result gives a non-adaptive $O(1)$-approximation algorithm for Stoch-Cost k-TSP.

▶ **Theorem 2.** *The Stoch-Cost k-TSP problem has a non-adaptive $O(1)$-approximation algorithm.*

Our main techniques in the proof of Theorem 2 are similar to those for Stoch-Reward k-TSP. In fact, in Section 3 we present a generic framework that can be used to solve both these problems. It will be interesting to find other applications of our framework in future work.

1.3 High-level Techniques

We assume that after re-scaling, the distance between any pair of vertices is at least 1.

Stoch-Reward k-TSP. A standard idea in the design of approximation algorithms on a metric is to operate in phases, where in phase i our algorithm is allowed a budget of 2^i. Intuitively, this corresponds to the algorithm imagining that the optimal adaptive tour has length $\Theta(2^i)$. In each phase, a naïve algorithm would be to collect as much *expected* reward as possible within budget 2^i (say, by solving an instance of the Orienteering problem). However, in general the performance of such a naïve algorithm can be arbitrarily bad. E.g., suppose $k - \sqrt{k}$ reward is easy to get, now for the remaining reward the naïve algorithm prefers a vertex having a reward of k with probability $10/\sqrt{k}$ and 0 otherwise, as opposed to a vertex with a deterministic reward of $\sqrt{k}$ (assuming both are at the same distance).

[5] Up to a factor of 2, this version of the problem is equivalent to the version where restriction (2) is removed.

A natural fix to the above issue is to *truncate* the reward distributions at the *remaining* target reward and then take expectations. Not only is this algorithm *adaptive*, it is not even clear why it would work. Indeed, Ene et al. [13] give an example (see Example 2 in [13]) where this algorithm has an $\Omega(\log k)$-approximation factor. Our first idea is to run $O(1)$ *repetitions* of a bi-criteria Orienteering algorithm using the *same* truncation (i.e., the initial remaining target) for all these repetitions. Our analysis applies Freedman's [14] and Jogdeo-Samuels [27] inequalities to argue concentration, and relies crucially on not updating the remaining target for these $O(1)$ repetitions.

Nevertheless, the above approach depends on the remaining target, which is unknown to a *non-adaptive* algorithm. One could bypass this by truncating the reward distributions at $\log k$ different *scales*, where scale j corresponds to the remaining target being roughly $k/2^j$, applying the previous $O(1)$ repetitions idea at each scale, and visiting the union of all tours. Unfortunately, this immediately loses a $\log k$ approximation factor. Our second idea is "critical scaling" in which we identify a "critical" scale j_{crit} among the $\log k$ possible scales, and only include tours for scales $j_{\mathsf{crit}} - 1$ and j_{crit}. This critical scale j_{crit} roughly (but not quite) corresponds to a "phase transition" from an underestimation to an overestimation of the remaining target. A priori it is not clear why such a "critical" scale can be found non-adaptively, but the concentration properties of the above $O(1)$ repetitions allow us to find it efficiently.

Stoch-Cost k-TSP. We obtain our result for Stoch-Cost k-TSP in a similar way. An immediate challenge, however, is how should we truncate the cost distributions C_v, say even if the remaining target k' is known? One natural way is by looking at $\mathbb{P}[C_v \leq O(2^i/k')]$, where $O(2^i/k')$ is the "average" cost per remaining reward in phase i. But such an approach would fail when some vertices in the optimal tour have costs much smaller than $O(2^i/k')$, while the other vertices have much higher costs. We overcome this by considering $\mathbb{P}[C_v \leq O(2^{i-j})]$ for all possible scales $j \in \{0, \cdots, i + \log n\}$, identifying a "critical" scale j_{crit}, and again only including the tours for scales $j_{\mathsf{crit}} - 1$ and j_{crit}. To identify the critical scale, we need to evaluate the maximum target a tour at a given scale can get within cost budget 2^i with constant probability. We show this can be approximately computed via dynamic programming.

1.4 Further Related Work

There is a long line of work studying the classic k-TSP and the related k-MST problem; we refer the readers to Garg's beautiful 2-approximation paper and the references therein [17].

A formal study on the benefits of adaptivity for stochastic combinatorial optimization problems started with the seminal work of Dean et al. [11]. They showed that for the stochastic knapsack problem, where items sizes are independently drawn and we need to fit them in a knapsack of size B, there is an $O(1)$-approximation non-adaptive algorithm. This factor was later improved to a $(2 + \epsilon)$-approximation in [7, 29]. The minimization version of the stochastic knapsack problem, which is known as the Stoch-Knapsack Cover problem, was studied by Deshpande et al. [12], and is a special case of Stoch-Reward k-TSP as we mentioned before. The unbounded version of Stoch-Knapsack Cover (each item has infinite number of copies) was studied by [26] and they provide an FPTAS for this problem.

Gupta et al. [23] generalized the stochastic knapsack problem to the stochastic orienteering problem, where each stochastic item now resides on a vertex of a given metric, and we need to fit both the tour length and the item sizes inside our budget B. They give an $O(\log \log B)$-approximation non-adaptive algorithm for this problem. Bansal and Nagarajan [6] later showed that this problem has no constant-approximation non-adaptive algorithm. These

works inspired Ene et al. [13] to study Stoch-Reward k-TSP, a natural minimization variant of the stochastic orienteering problem. Prior to our work, it was conceivable that this minimization problem also has a super-constant adaptivity gap, like stochastic orienteering.

Motivated by different applications that solve discrete problems under an uncertain input, other related stochastic probing models have been studied. We refer the readers to Singla's Ph.D. Thesis for a survey [32]. Of particular interest to us is the Price of Information model [33], which was inspired from the work on Pandora's box [34, 28]. Their Minimization k-Pandora's Box problem inspired us to define Stoch-Cost k-TSP, which generalizes the probing costs from being fixed to being on a metric. Although an optimal strategy is known for Minimization k-Pandora's Box, the problem becomes APX-hard on a metric as it generalizes k-TSP.

Organization

We start with some preliminary definitions and lemmas in Section 2. In Section 3, we describe our general framework for both Stoch-Reward k-TSP and Stoch-Cost k-TSP, and prove some key lemmas that will be used throughout the paper. We give our non-adaptive $O(1)$-approximation algorithm for Stoch-Reward k-TSP that proves Theorem 1 in Section 4. The non-adaptive $O(1)$-approximation algorithm for Stoch-Cost k-TSP that proves Theorem 2 is in Section 5.

2 Preliminaries

2.1 Adaptive vs Non-Adaptive Algorithms

Any feasible solution to our stochastic problems can be described by a *decision tree*, where nodes correspond to vertices that are visited and branches correspond to instantiations of the observed random variables. Even if the degree of every vertex is a constant, the size of such decision trees can be exponentially large in its height. These solutions are called *adaptive* because the choice of the next vertex to visit depends on the outcomes of the already visited nodes.

We also consider the special class of *non-adaptive* solutions that is described simply by an ordered list of vertices: the policy involves visiting vertices in the given order until a certain stopping criterion is met. Such non-adaptive solutions are often preferred over adaptive solutions because they are easier to implement.

In this work we only study minimization problems. We compare the performance of our algorithm with that of the optimal *adaptive* algorithm, which is denoted by OPT. We abuse notation and also use OPT to denote the expected objective of the optimal adaptive algorithm. We say an algorithm is α-approximation for $\alpha \geq 1$ if the expected objective of the algorithm is at most $\alpha \cdot \mathsf{OPT}$. Ideally, we want to design non-adaptive algorithms whose performance is comparable to the optimal adaptive algorithm. Since this is not always possible, it is important to bound the *adaptivity gap*, which is the worst-case ratio between the expected objectives of the optimal non-adaptive and the optimal adaptive algorithms.

2.2 Probability Inequalities

Our proofs will require the following probability inequalities. We start with a bound on the median of independent Bernoulli random variables due to Jogdeo and Samuels [27]. Given n independent Bernoulli random variables $X_1, \cdots, X_n$ where X_i has success probability $p_i \in [0, 1]$, let $X := \sum_{i=1}^{n} X_i$ be their sum. Define the median of X to be any integer $m \in \mathbb{Z}_{\geq 0}$ such that $\min\{\mathbb{P}[X \geq m], \mathbb{P}[X \leq m]\} \geq 1/2$.

▶ **Theorem 3** (Theorem 3.2 and Corollary 3.1 [27]). *Let $X = \sum_{i=1}^{n} X_i$ be the sum of n independent Bernoulli random variables where X_i has success probability $p_i \in [0,1]$. If $\mathbb{E}[X]$ is an integer k, then the median of X is also k. If $k < \mathbb{E}[X] < k+1$ for some integer k, then the median of X is either k or $k+1$.*

We will also need the following martingale inequality due to Freedman [14].

▶ **Theorem 4** (Freedman's Inequality, Theorem 1.6 in [14]). *Consider a real-valued martingale sequence $\{X_t\}_{t\geq 0}$ such that $X_0 = 0$, and $\mathbb{E}[X_{t+1}|\mathcal{F}_t] = 0$ for all t, where $\{\mathcal{F}_t\}_{t\geq 0}$ is the filtration defined by the martingale. Assume that the sequence is uniformly bounded, i.e., $|X_t| \leq M$ almost surely for all t. Now define the predictable quadratic variation process of the martingale to be $W_t = \sum_{j=1}^{t} \mathbb{E}[X_j^2|\mathcal{F}_{j-1}]$ for all $t \geq 1$. Then for all $\ell \geq 0$ and $\sigma^2 > 0$ and any stopping time τ, we have*

$$\mathbb{P}\Big[\Big|\sum_{j=0}^{\tau} X_j\Big| \geq \ell \wedge W_\tau \leq \sigma^2 \text{for some stopping time } \tau\Big] \leq 2\exp\Big(-\frac{\ell^2/2}{\sigma^2 + M\ell/3}\Big).$$

▶ **Theorem 5** (Chernoff Bound). *Let $X_1, X_2, \cdots, X_n$ be independent random variables taking values in $[0,1]$ and define $X := \sum_{i\in[n]} X_i$. Then for any $\delta \in [0,1]$, we have*

$$\mathbb{P}\left[X \leq (1-\delta)\cdot\mathbb{E}[X]\right] \leq \exp\left(-\delta^2\cdot\mathbb{E}[X]/2\right).$$

2.3 A Bi-Criteria Algorithm $\mathsf{ALG}_{\mathsf{Bicrit\text{-}Orient}}$ for Orienteering

We formally define the well-known Orienteering problem.

Orienteering. Given a metric (V,d) with root $\in V$, a *profit*[6] $R_v > 0$ for each $v \in V$, and a budget $B > 0$, the goal is to find a tour originating at root of length at most B that maximizes the collected profit.

The state-of-the-art for this NP-hard Orienteering problem is a $(2+\epsilon)$-approximation algorithm [10]. We denote this algorithm as $\mathsf{ALG}_{\mathsf{Orient}}$ and denote the profit of the optimal Orienteering tour as $\mathsf{OPT}_{\mathsf{Orient}}$. For our purposes, however, we also need to find profit at least $\mathsf{OPT}_{\mathsf{Orient}}$ minus an arbitrarily small additive error. To achieve this, the tour found by our algorithm has length $O(1)\cdot B$.

▶ **Lemma 6** (Bi-criteria Orienteering). *There is an efficient algorithm $\mathsf{ALG}_{\mathsf{Bicrit\text{-}Orient}}$ that finds a tour of length $O(1)\cdot B$ while collecting at least $(\mathsf{OPT}_{\mathsf{Orient}} - \epsilon)$ profit, where $\epsilon = 1/\mathrm{poly}(n)$ can be made arbitrarily small.*

3 Our Approach via Critical Scaling and Repetitions

3.1 A Meta-Algorithm and Critical Scaling

Our non-adaptive $O(1)$-approximation algorithms for both the problems, Stoch-Reward k-TSP and Stoch-Cost k-TSP, have the same structure described in Meta-Algorithm $\mathsf{ALG}_{\mathsf{Meta}}$ (Algorithm 1). This algorithm operates in phases where it gets a budget of $O(1)\cdot\gamma^i$ in phase $i \geq 0$ for some constant $\gamma \in (1,2)$.

[6] We use the word "profit" for Orienteering to avoid confusion with the "reward" in Stoch-Reward k-TSP.

In each phase i, $\mathsf{ALG}_{\mathsf{Meta}}$ explores multiple different "scales" in the remaining graph after excluding the set Π of vertices found in the previous phases. Recall, each scale corresponds to truncating the random variables at a different threshold. This is crucial because our non-adaptive algorithm doesn't know the remaining reward to reach the target k. For each different scale, $\mathsf{ALG}_{\mathsf{Meta}}$ obtains a tour of length $O(1) \cdot \gamma^i$ via a sub-procedure $\mathsf{ALG}_{\mathsf{Rep}}$ to which it sends the truncated random variables as arguments (discussed in Section 3.2). Eventually, $\mathsf{ALG}_{\mathsf{Meta}}$ identifies a "critical" scale j_{crit} and appends at the end of Π the two tours corresponding to the scales j_{crit} and $j_{\mathsf{crit}} - 1$. Since we only append two tours, $\mathsf{ALG}_{\mathsf{Meta}}$ uses budget at most $O(1) \cdot \gamma^i$ in phase i.

Algorithm 1 A Meta-Algorithm $\mathsf{ALG}_{\mathsf{Meta}}$.

Pre-processing stage:
set $\gamma \in (1,2), \Pi \leftarrow \emptyset$ and $\ell \leftarrow \mathsf{polylog}(k,n)$;
for *phase* $i = 0, 1, \cdots$ **do**
  set $\Pi_{i,-1} \leftarrow \emptyset$;
  for *scale* $j = 0, \cdots, \ell$ **do**
    set $X_v^j \in [0,1]$ to be the truncation of X_v at scale j for $v \in V \setminus \Pi$, and zero for $v \in \Pi$;
    find tour $\Pi_{i,j} \leftarrow \mathsf{ALG}_{\mathsf{Rep}}\big(\{X_v^j\}_{v \in V \setminus \Pi}\ ,\ i\big)$ of length $O(1) \cdot \gamma^i$;
  end
  identify a "critical" scale j_{crit} and set $\Pi_i \leftarrow \Pi_{i,j_{\mathsf{crit}}} \cup \Pi_{i,j_{\mathsf{crit}}-1}$;
  append tour Π_i to Π, i.e., $\Pi \leftarrow \Pi \circ \Pi_i$;
end

Probing stage:
for *phase* $i = 0, 1, \cdots$ **do**
  visit vertices in the order of Π_i and apply certain Selection and Stopping Criteria;
end
Return set of vertices selected

To analyze the algorithm, we need some notation for any phases $i, i' \geq 1$:

- σ_{i-1}: outcome of vertices visited by $\mathsf{ALG}_{\mathsf{Meta}}$'s in the first $i-1$ phases of the probing stage.
- $u_{i'}(\sigma_{i-1})$: probability that $\mathsf{ALG}_{\mathsf{Meta}}$ enters phase $i'+1$ in the probing stage, conditioning on σ_{i-1}.
- $u^*_{i'}(\sigma_{i-1})$: probability that the cost of OPT is more than $\gamma^{i'}$, conditioning on σ_{i-1}.

Notice $u_{i-1}(\sigma_{i-1})$ denotes the indicator variable that $\mathsf{ALG}_{\mathsf{Meta}}$ enters phase i in the probing stage. The following Lemma 7 is the key to our theorems. Roughly, it says that $\mathsf{ALG}_{\mathsf{Meta}}$ is a constant approximation algorithm if it can ensure that whenever $u^*_i(\sigma_{i-1})$ is small (i.e., OPT has a large success probability within budget γ^i) then $\mathsf{ALG}_{\mathsf{Meta}}$ also succeeds with a constant probability in the first i phases (i.e., only using $O(1) \cdot \gamma^i$ budget). The proof of Lemma 7 is standard (e.g., [13]), and we defer it to Appendix B.

▶ **Lemma 7** (Key Lemma). *If for some universal constants $C > 0$, $\gamma > 1$, any phase $i \geq 1$, and any possible σ_{i-1}, the algorithm ALG_{Meta} satisfies*

$$u_i(\sigma_{i-1}) \leq C \cdot u^*_i(\sigma_{i-1}) + \frac{u_{i-1}(\sigma_{i-1})}{\gamma^2},$$

then ALG_{Meta} is a non-adaptive $O(1)$-approximation algorithm.

All our effort will go in designing $\mathsf{ALG}_{\mathsf{Meta}}$ that satisfies the precondition of Lemma 7.

3.2 $\mathsf{ALG}_{\mathsf{Rep}}$: Constant Repetitions of $\mathsf{ALG}_{\mathsf{Bicrit\text{-}Orient}}$ Suffice

For a fixed scale j in phase i, $\mathsf{ALG}_{\mathsf{Meta}}$ uses $\mathsf{ALG}_{\mathsf{Rep}}$ (Algorithm 2) as a key sub-procedure to find a tour of length $O(1) \cdot \gamma^i$. To achieve this, $\mathsf{ALG}_{\mathsf{Rep}}$ runs a constant number of repetitions of $\mathsf{ALG}_{\mathsf{Bicrit\text{-}Orient}}$ on an Orienteering instance where each vertex v has a profit $w_v = \mathbb{E}[X_v]$ for input random variable $X_v \in [0,1]$ (recall, X_v is the truncated random variable at scale j for vertices outside Π). In each repetition, $\mathsf{ALG}_{\mathsf{Rep}}$ excludes vertices found in previous repetitions.

Algorithm 2 $\mathsf{ALG}_{\mathsf{Rep}}\big(\{X_v\}_{v\in V}\ ,\ i\big)$.

Input: random variables $X_v \in [0,1]$ corresponding to vertex profits and phase i;
Main stage:
set $\gamma \in (1,2)$, $\epsilon \leftarrow 1/10^5$, $C \leftarrow O(1)$, and $w_v = \mathbb{E}[X_v]$;
set $\Pi_i \leftarrow \emptyset$;
for *repetition* $s = 1, \cdots, C$ **do**
  use $\mathsf{ALG}_{\mathsf{Bicrit\text{-}Orient}}$ to find a tour π_s with budget γ^i, profit $\{w_v\}_{v\in V}$, and error ϵ;
  append tour π_s to Π_i, i.e., $\Pi_i \leftarrow \Pi_i \circ \pi_s$;
  reset $w_v = 0$ for $v \in \Pi_i$;
end
Return Π_i

Intuition. In the following, we prove two important properties of $\mathsf{ALG}_{\mathsf{Rep}}$. Recall from Algorithm 2 that Π_i denotes the union of the C repetitions of $\mathsf{ALG}_{\mathsf{Bicrit\text{-}Orient}}$. The first property (Lemma 8) roughly says that if an Orienteering tour of budget γ^i cannot obtain much profit outside Π_i, then OPT also cannot obtain much reward outside Π_i within budget γ^i. The second property (Lemma 9) roughly says that if on the other hand lots of profit can be found by an Orienteering tour outside Π_i, then the tour Π_i obtains a large amount of *expected* reward in its C repetitions, much more than what OPT obtains within budget γ^i. This follows from the property that $\mathsf{ALG}_{\mathsf{Bicrit\text{-}Orient}}$ obtains profit close to the optimal Orienteering tour.

To formally state the above two properties, we need some notation. Consider the Orienteering instance in the remaining graph $V \setminus \Pi_i$ where the budget is γ^i and each vertex $v \in V \setminus \Pi_i$ has profit $\mathbb{E}[X_v]$. Denote $\pi \subseteq V \setminus \Pi_i$ the optimal Orienteering tour for this instance and let

$$T := \sum_{v\in\pi} \mathbb{E}[X_v] \tag{1}$$

be the Orienteering profit obtained by π. For any adaptive strategy ADAP, let $\Pi_i(\mathsf{ADAP}) \subseteq \Pi_i$ denote the random set of vertices visited by ADAP inside the tour Π_i and let $\overline{\Pi}_i(\mathsf{ADAP}) \subseteq V \setminus \Pi_i$ denote the random set of vertices visited by ADAP outside the tour Π_i.

▶ **Lemma 8.** *Suppose we are given independent random variables $X_v \in [0,1]$. Let T be as defined in (1). Then for any adaptive strategy* ADAP *which uses at most γ^i budget and any constant $\alpha > 1$, we have*

$$\mathbb{P}\Big[\sum_{v\in\overline{\Pi}_i(\mathsf{ADAP})} X_v \geq \alpha T\Big] \ \leq\ 2\cdot\exp\left(-\frac{(\alpha-1)^2 T/2}{1+(\alpha-1)/3}\right).$$

Proof of Lemma 8. We construct a martingale for the (random) set $\overline{\Pi}_i(\mathsf{ADAP})$ of vertices visited by ADAP in $V \setminus \Pi_i$ as follows: When ADAP visits a vertex $v \in \overline{\Pi}_i(\mathsf{ADAP})$, the martingale proceeds for one step with martingale difference defined by

$$Z_v \;:=\; X_v - \mathbb{E}[X_v] \;\in\; [-1,1],$$

and the martingale doesn't move when ADAP visits a vertex $v \in \Pi_i$. The stopping time τ is naturally defined as the martingale step when ADAP finishes. Since each $X_v \in [0,1]$, the quadratic variance of the above martingale $W_\tau = \sum_{j=1}^{\tau} \mathbb{E}[X_j^2|\mathcal{F}_{j-1}]$ is bounded by its expectation $\sum_{j=1}^{\tau} \mathbb{E}[X_j|\mathcal{F}_{j-1}]$ which is at most T, i.e., $\sum_{v\in\overline{\Pi}_i(\mathsf{ADAP})} \mathbb{E}[X_v] \le T$. Therefore, applying Freedman's inequality (Theorem 4), we have

$$\begin{aligned}\mathbb{P}\Big[\sum_{v\in\overline{\Pi}_i(\mathsf{ADAP})} X_v \ge \alpha T\Big] \;&\le\; \mathbb{P}\Big[\Big|\sum_{v\in\overline{\Pi}_i(\mathsf{ADAP})} Z_v\Big| \ge (\alpha-1)T \wedge W_\tau \le T\Big] \\ &\le\; 2\cdot\exp\left(-\frac{(\alpha-1)^2T/2}{1+(\alpha-1)/3}\right).\end{aligned}$$

This finishes the proof of Lemma 8. ◀

▶ **Lemma 9.** *Suppose we are given independent random variables $X_v \in [0,1]$. Let T be as defined in (1). Then for any adaptive strategy* ADAP *which uses budget at most γ^i, we have*

$$\sum_{v\in\Pi_i\setminus\Pi_i(\mathsf{ADAP})} \mathbb{E}[X_v] \;\ge\; (C-1)(T-\epsilon)-\epsilon.$$

Proof of Lemma 9. Since ADAP uses budget at most γ^i, the set $\Pi_i(\mathsf{ADAP}) \subseteq \Pi_i$ can always be visited by a tour of length at most γ^i. Consider the first tour π_1 found by Algorithm 2. Since the tour with length at most γ^i that visits the set $\Pi_i(\mathsf{ADAP})$ is a valid Orienteering tour when π_1 is found, it follows from Lemma 6 that $\sum_{v\in\Pi_i(\mathsf{ADAP})} \mathbb{E}[X_v] \le \epsilon + \sum_{v\in\pi_1} \mathbb{E}[X_v]$. For any $s \in \{2,3,\cdots,C\}$, since π is an Orienteering tour in $V\setminus\Pi_i$ of length at most γ^i with $\sum_{v\in\pi} \mathbb{E}[X_v] = T$, Lemma 6 implies that $\sum_{v\in\pi_s} \mathbb{E}[X_v] \ge T-\epsilon$. Therefore, a simple calculation gives

$$\sum_{v\in\Pi_i\setminus\Pi_i(\mathsf{ADAP})} \mathbb{E}[X_v] \;=\; \sum_{v\in\Pi_i} \mathbb{E}[X_v] - \sum_{v\in\Pi_i(\mathsf{ADAP})} \mathbb{E}[X_v] \;\ge\; (C-1)(T-\epsilon)-\epsilon,$$

which finishes the proof of Lemma 9. ◀

4 Stoch-Reward k-TSP

In this section we prove Theorem 1, which is restated below for convenience.

▶ **Theorem 1.** *The Stoch-Reward k-TSP problem has a non-adaptive $O(1)$-approximation algorithm.*

To prove this theorem we carefully choose the parameters of our Meta-Algorithm from last section.

4.1 The Algorithm

Algorithm 3 $\mathsf{ALG}_{\text{Stoch-Reward}}$ for Stoch-Reward k-TSP problem.

Pre-processing stage:
set $\gamma \leftarrow 1.1, \epsilon \leftarrow 1/10^5, \Pi \leftarrow \emptyset$, $\ell = \lfloor \log k \rfloor$, and $C \leftarrow 6000$;
for *phase* $i = 0, 1, \cdots$ **do**
  set $\Pi_{i,-1} \leftarrow \emptyset$;
  for *scale* $j = 0, \cdots, \ell$ **do**
    set profit $w_v^j = \mathbb{E}\left[\min\left\{R_v \cdot 2^j/k, 1\right\}\right] \cdot \mathbf{1}[v \in V \setminus \Pi]$; /* Scale j truncates at $k/2^j$ */
    set $\Pi_{i,j} \leftarrow \emptyset$;
    for *repetition* $s = 1, 2, \cdots, C$ /* Constant repetitions of $\mathsf{ALG}_{\text{Bicrit-Orient}}$ */ **do**
      use $\mathsf{ALG}_{\text{Bicrit-Orient}}$ to find tour $\pi_{i,j,s}$ with budget γ^i, profit $\{w_v^j\}_{v \in V}$, and error ϵ ;
      append tour $\pi_{i,j,s}$ to $\Pi_{i,j}$, i.e., $\Pi_{i,j} \leftarrow \Pi_{i,j} \circ \pi_{i,j,s}$;
      reset $w_v^j = 0$ for $v \in \Pi_{i,j}$;
    end
    /* Check whether j is a "critical" scale */
    use $\mathsf{ALG}_{\text{Orient}}$ to find tour $\pi_{i,j}^{\text{Ori}}$ with budget γ^i and profit $\{w_v^j\}_{v \in V}$;
    set $T_{i,j} \leftarrow \sum_{v \in \pi_{i,j}^{\text{Ori}}} w_v^j$;
    if $T_{i,j} \geq 1/300$ *or* $j = \ell$ **then**
      append tour $\Pi_i := \Pi_{i,j} \cup \Pi_{i,j-1}$ to Π, i.e., $\Pi \leftarrow \Pi \circ \Pi_i$;
      Break ;
    end
  end
end

Probing stage:
for *phase* $i = 0, 1, \cdots$ **do**
  visit vertices in the order of Π_i and apply the following selection and stopping criteria ;
  Selection Criterion: select every vertex visited ;
  Stopping Criterion: total reward reaches k ;
end
Return set of vertices selected

We assume without loss of generality that the stochastic reward $R_v \leq k$ almost surely for each vertex $v \in V$. Our algorithm $\mathsf{ALG}_{\text{Stoch-Reward}}$ for Stoch-Reward k-TSP problem is given in Algorithm 3. It is an instantiation of the Meta-Algorithm $\mathsf{ALG}_{\text{Meta}}$ (Algorithm 1) by setting the phase parameter $\gamma = 1.1$, number of scales $\ell = \lfloor \log k \rfloor$, and number of repetitions $C = 6000$. For each scale j in phase i, we set the random variable X_v^j for any vertex $v \in V \setminus \Pi$ in $\mathsf{ALG}_{\text{Meta}}$ to be the stochastic reward R_v truncated at $k/2^j$ and then scaled down to $[0, 1]$. We identify the "critical" scale j_{crit} in $\mathsf{ALG}_{\text{Meta}}$ as follows: For each scale j in phase i, denote $\Pi_{i,j}$ the $C = 6000$ repetitions of $\mathsf{ALG}_{\text{Bicrit-Orient}}$ and let $T_{i,j}$ be the profit obtained by the 3-approximation Orienteering algorithm $\mathsf{ALG}_{\text{Orient}}$ in $V \setminus (\Pi \cup \Pi_{i,j})$. The "critical" scale j_{crit} is the smallest scale j such that $T_{i,j} \geq 1/300$, i.e., sufficient profit remains outside even after C repetitions. We add the two tours corresponding to the "rich" scale j_{crit} and the "poor"

scale $j_{\mathsf{crit}} - 1$ into Π. In the case when there is no scale j with $T_{i,j} \geq 1/300$, we simply set j_{crit} to be the last scale ℓ. In the probing stage, the selection and the stopping criteria are straightforward: we collect reward from every visited vertex and stop when the total reward reaches k.

4.2 Proof of Theorem 1

Recall from Section 3.1 that to prove Theorem 1, we only need to prove the precondition in Lemma 7. The remainder of this section proves this precondition for $\mathsf{ALG}_{\mathsf{Stoch\text{-}Reward}}$ as given in the following Lemma 10.

▶ **Lemma 10.** *For $\gamma = 1.1$, any phase $i > 0$ in the probing stage of $\mathsf{ALG}_{\mathsf{Stoch\text{-}Reward}}$ satisfies*

$$u_i(\sigma_{i-1}) \leq 100 \cdot u_i^*(\sigma_{i-1}) + \frac{u_{i-1}(\sigma_{i-1})}{\gamma^2}. \tag{2}$$

Before proving Lemma 10, we discuss the high-level intuition of our proof.

Intuition. Assume without loss of generality that $u_i^*(\sigma_{i-1}) < 0.01$, i.e., OPT finds k reward within budget γ^i with probability at least 0.99, as otherwise the lemma trivially holds. Now the plan is to show that with constant probability, $\mathsf{ALG}_{\mathsf{Stoch\text{-}Reward}}$ finds more reward than OPT restricted to budget γ^i, even when all rewards in σ_{i-1} are given to OPT for free. This allows us to focus on the remaining graph with vertex set $V_i := V \setminus \sigma_{i-1}$ where we repeat $\mathsf{ALG}_{\mathsf{Bicrit\text{-}Orient}}$ for different scales.

Our arguments rely on the notion of "richness". We call a scale j "rich" if $T_{i,j} \geq 1/300$ and otherwise "poor". A scale being poor indicates that not much reward can be collected outside the C repetitions of $\mathsf{ALG}_{\mathsf{Bicrit\text{-}Orient}}$ for that scale, in which case we can use Lemma 8 to argue that OPT cannot find much reward outside. A scale being rich implies that each repetition of $\mathsf{ALG}_{\mathsf{Bicrit\text{-}Orient}}$ for that scale finds a significant amount of reward, in which case we can apply Lemma 9 to argue that $\mathsf{ALG}_{\mathsf{Stoch\text{-}Reward}}$ finds much more reward than OPT in the C repetitions for that scale. Our critical scale j_{crit} corresponds to the transition from poor to rich scales. Since the algorithm includes both j_{crit} and $j_{\mathsf{crit}} - 1$, roughly the reason why our analysis works is that we use the poor scale $j_{\mathsf{crit}} - 1$ to argue OPT cannot find much reward outside our tours and we use the rich scale j_{crit} to argue OPT cannot find much more reward inside. The final analysis has to do some case analysis depending on whether the transition ever happens or not.

Proof of Lemma 10. We fix any outcome σ_{i-1} of vertices visited by $\mathsf{ALG}_{\mathsf{Stoch\text{-}Reward}}$ in the first $i-1$ phases in its probing stage. The lemma trivially holds in the case where $u_i^*(\sigma_{i-1}) \geq 0.01$ as we have $100u_i^*(\sigma_{i-1}) \geq 1$. If $u_{i-1}(\sigma_{i-1}) = 0$ which means that $\mathsf{ALG}_{\mathsf{Stoch\text{-}Reward}}$ already collects reward k before entering phase i in the probing stage, then $u_i(\sigma_{i-1}) = 0$ and again the lemma trivially holds. We therefore assume that $u_i^*(\sigma_{i-1}) < 0.01$ and that $u_{i-1}(\sigma_{i-1}) = 1$. Now proving Lemma 10 is equivalent to proving

$$u_i(\sigma_{i-1}) \leq 100u_i^*(\sigma_{i-1}) + 1/\gamma^2. \tag{3}$$

To prove (3), we need the following notation. Denote $V_i \coloneqq V \setminus \sigma_{i-1}$ the vertex set of the remaining graph where vertices in σ_{i-1} are excluded. Denote $\overline{\Pi}_i(\mathsf{OPT}) \subseteq V_i \setminus \Pi_i$ the (random) set of vertices visited by OPT outside $\sigma_{i-1} \cup \Pi_i$ within budget γ^i, and denote $\Pi_i(\mathsf{OPT}) \subseteq \Pi_i$ the (random) set of vertices visited by OPT inside Π_i. We consider three cases:

Case (1): (Scale 0 is rich) $T_{i,0} \geq 1/300$. In this case, our algorithm appends tour $\Pi_i := \Pi_{i,0}$ to Π (recall that $\Pi_{i,-1} = \emptyset$), and these will be the phase i vertices visited in the probing stage. We show that $T_{i,0} \geq 1/300$ implies that each repetition of $\mathsf{ALG}_{\text{Bicrit-Orient}}$ has large expected reward (notice the random rewards are not truncated at scale 0). As we repeat $\mathsf{ALG}_{\text{Bicrit-Orient}}$ for $C = 6000$ times, the tour $\Pi_{i,0}$ has expected reward much larger than the target k.

Since $T_{i,0}$ is the profit of a valid Orienteering tour with length at most γ^i, for each $s \in \{1, \ldots, C\}$ we have $T_{i,0,s} \geq T_{i,0} - \epsilon \geq 1/300 - \epsilon$, where $\epsilon = 1/10^5$ is the small error term for $\mathsf{ALG}_{\text{Bicrit-Orient}}$ in Lemma 6. Thus,

$$\sum_{v \in \Pi_{i,0}} \mathbb{E}\left[\min\left\{R_v/k, 1\right\}\right] \quad \geq \quad 20 - 6000\epsilon \quad \geq \quad 19.$$

Notice that $R_v/k \in [0,1]$, so applying Chernoff bound (Theorem 5) we have

$$1 - u_i(\sigma_{i-1}) \quad \geq \quad \mathbb{P}\Big[\sum_{v \in \Pi_{i,0}} R_v \geq k\Big] \quad = \quad \mathbb{P}\Big[\sum_{v \in \Pi_{i,0}} \min\left\{R_v/k, 1\right\} \quad \geq \quad 1\Big] \quad \geq \quad 0.9,$$

which means $u_i(\sigma_{i-1}) \leq 0.1 \leq 1/\gamma^2$. This proves (3) and finishes the proof of Lemma 10 in this case.

Case (2): (Scale ℓ is poor) $T_{i,j} \leq 1/300$ for every scale $j = 0, \cdots, \ell$. In this case, our algorithm adds $\Pi_i := \Pi_{i,\ell} \cup \Pi_{i,\ell-1}$ to Π, and these will be the phase i vertices visited in the probing stage. We argue that the assumption of $T_{i,\ell} \leq 1/300$ implies that with constant probability OPT finds *no* reward outside $\sigma_{i-1} \cup \Pi_i$ within budget γ^i. If OPT still manages to find k reward, then all k reward must come from vertices in $\sigma_{i-1} \cup \Pi_i$, in which case $\mathsf{ALG}_{\text{Stoch-Reward}}$ also finds k reward. In this case our argument already works with the vertices $\Pi_{i,\ell}$ added to Π, i.e., we do not even need vertices in $\Pi_{i,\ell-1}$.

Since $\mathsf{ALG}_{\text{Orient}}$ is a 3-approximation Orienteering algorithm, for any set of vertices $S \subseteq V_i \setminus \Pi_i$ that can be visited within budget γ^i, we have $\sum_{v \in S} \mathbb{E}\left[\min\left\{R_v \cdot 2^\ell/k, 1\right\}\right] \leq 3T_{i,\ell} \leq 0.01$. Since $\ell = \lfloor \log k \rfloor$, we have $k/2^\ell \in [1,2]$, and therefore

$$\sum_{v \in S} \mathbb{E}[\min\{R_v, 1\}] \quad \leq \quad \sum_{v \in S} \mathbb{E}\left[\min\left\{2 \cdot R_v \cdot 2^\ell/k, 1\right\}\right] \quad \leq \quad 0.02. \tag{4}$$

Recall that $\overline{\Pi}_i(\mathsf{OPT}) \subseteq V_i \setminus \Pi_i$ denotes the (random) set of vertices visited by OPT outside $\sigma_{i-1} \cup \Pi_i$ within budget γ^i. Since each $\min\{R_v, 1\} \in \{0,1\}$, the best probability of obtaining truncated reward at least 1 in $V_i \setminus \Pi_i$ within budget γ^i is achieved by a non-adaptive strategy. Therefore, Markov's inequality together with (4) implies that

$$\mathbb{P}\Big[\sum_{v \in \overline{\Pi}_i(\mathsf{OPT})} \min\{R_v, 1\} \geq 1\Big] \;\leq\; 0.02.$$

Since $u_i^*(\sigma_{i-1}) < 0.01$, we have that with probability at least $1 - 0.01 - 0.02 = 0.97$, OPT finds k reward in V but 0 reward outside $\sigma_{i-1} \cup \Pi_i$. In this case, $\mathsf{ALG}_{\text{Stoch-Reward}}$ also finds k reward among vertices visited in the first i phases. So we have $u_i(\sigma_{i-1}) \leq 1 - 0.97 = 0.03 \leq 1/\gamma^2$, which establishes (3) in this case.

Case (3): (Transition from poor to rich scale at j_{crit}) $T_{i,0} \leq 1/300$ but $T_{i,j} > 1/300$ for some $j \in [\ell]$. In this case, let $j_{\text{crit}} = \min\left\{j \in [\ell] : T_{i,j} > 1/300\right\}$ be our critical scale. The algorithm appends $\Pi_i := \Pi_{i,j_{\text{crit}}-1} \cup \Pi_{i,j_{\text{crit}}}$ to Π. To prove that $\mathsf{ALG}_{\text{Stoch-Reward}}$ will not continue to phase $i+1$ with constant probability, we show that the following two events happen with constant probability:

1. OPT doesn't find too much reward outside $\sigma_{i-1} \cup \Pi_i$ within budget γ^i.
2. $\mathsf{ALG}_{\mathsf{Stoch\text{-}Reward}}$ finds much more reward inside Π_i than OPT does since it is restricted to budget γ^i.

We show that the first event follows from $T_{i,j_{\mathsf{crit}}-1} \leq 1/300$ while the second event follows from $T_{i,j_{\mathsf{crit}}} \geq 1/300$. From these we conclude that with constant probability, $\mathsf{ALG}_{\mathsf{Stoch\text{-}Reward}}$ obtains at least as much reward as OPT restricted to budget γ^i.

We first argue that OPT doesn't find too much reward outside $\sigma_{i-1} \cup \Pi_i$. Specifically, we prove that

$$\mathbb{P}\Big[\sum_{v \in \overline{\Pi}_i(\mathsf{OPT})} R_v < k/2^{j_{\mathsf{crit}}-1}\Big] \;\geq\; 0.5. \tag{5}$$

Notice that $T_{i,j_{\mathsf{crit}}-1} \leq 1/300$ together with the fact that $\mathsf{ALG}_{\mathsf{Orient}}$ is a 3-approximation for $\mathsf{Orienteering}$ implies that for any set of vertices $S \subseteq V_i \setminus \Pi_i$ that can be visited within distance γ^i, we have

$$\sum_{v \in S} \mathbb{E}\left[\min\left\{R_v \cdot 2^{j_{\mathsf{crit}}-1}/k, 1\right\}\right] \leq 0.01.$$

Since the random variables $\min\left\{R_v \cdot 2^{j_{\mathsf{crit}}-1}/k, 1\right\} \in [0,1]$, applying Lemma 8 we have

$$\mathbb{P}\Big[\sum_{v \in \overline{\Pi}_i(\mathsf{OPT})} \min\left\{R_v \cdot 2^{j_{\mathsf{crit}}-1}/k, 1\right\} \geq 1\Big] \quad\leq\quad 2\exp\left(-\frac{0.99^2/2}{0.01 + 0.99/3}\right) \quad\leq\quad 0.5,$$

which immediately implies (5).

Now we argue that inside Π_i, we find much more reward than OPT does when it's restricted to budget γ^i. Specifically, we prove that

$$\mathbb{P}\Big[\sum_{v \in \Pi_i \setminus \Pi_i(\mathsf{OPT})} R_v \geq k/2^{j_{\mathsf{crit}}-1}\Big] \;\geq\; 0.8, \tag{6}$$

where recall that $\Pi_i(\mathsf{OPT}) \subseteq \Pi_i$ is the (random) set of vertices visited by OPT inside Π_i within budget γ^i. Notice that $T_{i,j_{\mathsf{crit}}} > 1/300$ together with Lemma 9 implies that

$$\sum_{v \in \Pi_i \setminus \Pi_i(\mathsf{OPT})} \mathbb{E}[\min\{R_v \cdot 2^{j_{\mathsf{crit}}}/k, 1\}] \quad\geq\quad (6000-1) \cdot (T_{i,j_{\mathsf{crit}}} - \epsilon) - \epsilon \quad\geq\quad 19.$$

Applying Chernoff bound (Theorem 5), we have

$$\mathbb{P}\Big[\sum_{v \in \Pi_i \setminus \Pi_i(\mathsf{OPT})} \min\{R_v \cdot 2^{j_{\mathsf{crit}}}/k, 1\} \geq 2\Big] \;\geq\; 0.8,$$

which implies (6).

Now we complete the proof of (3) in this final case. From (5) and (6) and our assumption that $u_i^*(\sigma_{i-1}) < 0.01$, we have that with probability at least $1 - 0.5 - 0.2 - 0.01 \geq 1/4$, all the following three events hold: (1) $\sum_{v \in \overline{\Pi}_i(\mathsf{OPT})} R_v < k/2^{j_{\mathsf{crit}}-1}$, (2) $\sum_{v \in \Pi_i \setminus \Pi_i(\mathsf{OPT})} R_v \geq k/2^{j_{\mathsf{crit}}-1}$, and (3) OPT obtains at least k reward within budget γ^i. When all these three events hold, $\mathsf{ALG}_{\mathsf{Stoch\text{-}Reward}}$ also finds at least k reward before visiting any vertex from phase $i+1$. Therefore, $u_i(\sigma_{i-1}) \leq 3/4 \leq 1/\gamma^2$, and this completes the proof of Lemma 10. ◀

5 Stoch-Cost k-TSP

In this section we prove Theorem 2, which is restated below for convenience. Throughout this section, we will remove the restriction that a vertex v can only be selected if we are currently at v because this is equivalent to the original problem up to a factor of 2.

▶ **Theorem 2.** *The Stoch-Cost k-TSP problem has a non-adaptive $O(1)$-approximation algorithm.*

Recall, an additional challenge for Stoch-Cost k-TSP is that there is no obvious way to truncate the cost distributions C_v, even if the remaining target k' is known. Truncating at the "average" cost per remaining reward (i.e., $\mathbb{P}[C_v \le O(\gamma^i/k')]$) will fail when some vertices in the optimal tour have costs much smaller than $O(\gamma^i/k')$, while the other vertices have much higher costs. We overcome this by considering $\mathbb{P}[C_v \le O(\gamma^i/2^j)]$ for all possible scales $j \in \{0, \cdots, i \cdot \log\gamma + \log n\}$. To identify a "critical" scale, we evaluate the maximum target a tour at a given scale can get with constant probability within cost budget 2^i. We show this can be approximately computed via dynamic programming.

The rest of this section is devoted to proving Theorem 2.

5.1 The Algorithm

Our algorithm $\mathsf{ALG}_{\text{Stoch-Cost}}$ for Stoch-Cost k-TSP is given in Algorithm 4. $\mathsf{ALG}_{\text{Stoch-Cost}}$ is an instantiation of our Meta-Algorithm $\mathsf{ALG}_{\text{Meta}}$ in Algorithm 1 by setting the phase parameter $\gamma = 1.1$ and number of repetitions $C = 6000$. The number of scales in phase $i \ge 0$ will be $\ell_i = \lfloor i \cdot \log\gamma + \log n \rfloor$. (Notice, unlike Stoch-Reward k-TSP, the number of scales changes with phase.) For scale j in phase i, we set a random variable X_v^j for vertex $v \in V \setminus \Pi$ in $\mathsf{ALG}_{\text{Meta}}$ to be the indicator variable that cost $C_v \le \gamma^i/2^j$.

We identify a "critical" scale $\widetilde{j}_{\text{crit}}$ as follows: For any phase $i \ge 0$ and scale $j \in \{0, \cdots, \ell_i\}$, define $Y_{i,j} \in \mathbb{Z}_{\ge 0}$ to be the maximum number of vertices that can be selected from $\Pi_{i,j} \cup \Pi_{i,j-1}$ with probability at least 0.2 within cost budget $3\gamma^i$. Ideally, we want to compute $Y_{i,j}$ for every scale $j \in \{0, \cdots, \ell_i\}$ and set the critical scale to be the one that maximizes $Y_{i,j}$. Unfortunately, $Y_{i,j}$ cannot be computed efficiently as the corresponding problem is NP-Hard. To get around this issue, we compute an approximate value $\widetilde{Y}_{i,j}$ in Step 15 of $\mathsf{ALG}_{\text{Stoch-Cost}}$ via a dynamic programming sub-procedure ALG_{DP}. We discuss the details of ALG_{DP} in Section 5.3, where we prove the following Lemma 11 which roughly says that the $\widetilde{Y}_{i,j}$ computed by $\mathsf{ALG}_{\text{Stoch-Cost}}$ is a reasonably good approximation of $Y_{i,j}$.

▶ **Lemma 11.** *For any phase $i \ge 0$ and any scale $j \in \{0, \cdots, \ell_i\}$, the approximate value $\widetilde{Y}_{i,j}$ computed in Step 15 of $\mathsf{ALG}_{\text{Stoch-Cost}}$ satisfies that (1) $\widetilde{Y}_{i,j} \ge Y_{i,j}$, and (2) $\widetilde{Y}_{i,j}$ vertices can be selected from $\Pi_{i,j} \cup \Pi_{i,j-1}$ within cost budget $6\gamma^i$ with probability at least 0.2.*

After computing $\widetilde{Y}_{i,j}$ for each scale $j \in \{0, \cdots, \ell_i\}$, we simply set $\widetilde{j}_{\text{crit}}$ to be the scale that maximizes $\widetilde{Y}_{i,j}$ and add the two tours corresponding to scales $\widetilde{j}_{\text{crit}} - 1$ and $\widetilde{j}_{\text{crit}}$ into Π.

In the probing stage, the Stopping Criterion is natural: we stop whenever the number of selected vertices reaches k. The selection process needs some care since not all vertices visited in the previous phases are selected. Our algorithm therefore runs two selection processes consecutively: In Selection-Process 1, we select as many unselected vertices from those visited in the previous phases within total cost γ^i. This is to ensure that we select from σ_{i-1} at least as many vertices as OPT restricted to budget γ^i. In Selection-Process 2, we select as many vertices as possible from Π_i within total cost $6\gamma^i$. The above Lemma 11 guarantees that at least $\widetilde{Y}_{i,\widetilde{j}_{\text{crit}}}$ vertices can be selected in this process with probability at least 0.2.

Algorithm 4 $\mathsf{ALG}_{\textsf{Stoch-Cost}}$ for Stoch-Cost k-TSP problem.

Pre-processing stage:
set $\gamma \leftarrow 1.1$, $\epsilon \leftarrow 1/10^5$, $\Pi \leftarrow \emptyset$ and $C \leftarrow 6000$;
for *phase* $i = 0, 1, \cdots$ **do**
  set $\Pi_{i,-1} \leftarrow \emptyset$;
  for *scale* $j = 0, \cdots, \ell_i$, *where* $\ell_i = \lfloor i \cdot \log \gamma + \log n \rfloor$ **do**
    set profit $w_v^j = \mathbb{P}[C_v \le \gamma^i / 2^j] \cdot 1[v \in V \setminus \Pi]$; /* Scale j "truncates" at $\gamma^i/2^j$ */
    set $\Pi_{i,j} \leftarrow \emptyset$;
    for *repetition* $s = 1, 2, \cdots, C$ /* *Constant repetitions of* $\mathsf{ALG}_{\textit{Bicrit-Orient}}$ */ **do**
      use $\mathsf{ALG}_{\textsf{Bicrit-Orient}}$ to find tour $\pi_{i,j,s}$ with budget γ^i, profit $\{w_v^j\}_{v \in V}$ and error ϵ ;
      append tour $\pi_{i,j,s}$ to $\Pi_{i,j}$, i.e., $\Pi_{i,j} \leftarrow \Pi_{i,j} \circ \pi_{i,j,s}$;
      reset $w_v^j = 0$ for $v \in \Pi_{i,j}$;
    end
    /* Approximately compute the maximum number of vertices that can be selected from $\Pi_{i,j} \cup \Pi_{i,j-1}$ within cost budget $3\gamma^i$ and with probability at least 0.2 */
    find the largest integer $\widetilde{Y}_{i,j} \le n$ such that $\mathsf{ALG}_{\mathsf{DP}}(\widetilde{Y}_{i,j}, 3\gamma^i, \Pi_{i,j} \cup \Pi_{i,j-1}) \ge 0.2$;
  end
  /* Identify a "critical" scale */
  set $\widetilde{j}_{\mathsf{crit}} \leftarrow \arg\max_j \widetilde{Y}_{i,j}$ and $\Pi_i \leftarrow \Pi_{i,\widetilde{j}_{\mathsf{crit}}} \cup \Pi_{i,\widetilde{j}_{\mathsf{crit}}-1}$;
  append tour Π_i to Π, i.e., $\Pi \leftarrow \Pi \circ \Pi_i$;
end

Probing stage:
for *phase* $i = 0, 1, \cdots$ **do**
  set $\sigma_{i-1} \leftarrow \bigcup_{t=0}^{i-1} \Pi_t$;
  visit vertices in the order of Π_i and apply the following selection and stopping criteria ;
  /* Select (unselected) vertices visited in previous phases */
  Selection-Process 1: select as many vertices as possible from σ_{i-1} within total cost γ^i ;
  /* Select vertices visited in the current phase */
  Selection-Process 2: select as many vertices as possible from Π_i within total cost $6\gamma^i$;
  Stopping Criterion: total number of vertices selected reaches k ;
end
Return the set of selected vertices

5.2 Proof of Theorem 2

Recall from Section 3.1 that to prove Theorem 2, we only need to prove the precondition in Lemma 7. The remainder of this section proves this precondition for $\mathsf{ALG}_{\mathsf{Stoch\text{-}Cost}}$ as stated in the following Lemma 12.

▶ **Lemma 12.** *For $\gamma = 1.1$, any phase $i > 0$ in the probing stage of Stoch-Cost k-TSP satisfies*

$$u_i(\sigma_{i-1}) \leq 100u_i^*(\sigma_{i-1}) + \frac{u_{i-1}(\sigma_{i-1})}{\gamma^2}. \tag{7}$$

Before proving Lemma 12, we need some notation.

Notation. For any (possibly adaptive) algorithm ADAP, we say ADAP has a *distance budget* of B if it is allowed to travel a total distance of at most B; we say ADAP has a *cost budget* of B if it is allowed to select vertices up to a total cost of B; we say ADAP has a *total budget* of B if its total distance travelled *plus* the total cost of selecting vertices is restricted to be at most B. An algorithm satisfying a budget constraint is said to be *within* that budget. For any phase i, denote $V_i := V \setminus \sigma_{i-1}$ the set of vertices in the remaining graph where vertices in σ_{i-1} are excluded. For any fixed outcome σ_{i-1} and any target $Y \in \mathbb{Z}_{\geq 0}$, denote $p^*_{i,Y}(\sigma_{i-1})$ the probability that OPT selects at least Y vertices from V_i within total budget γ^i and $p_{i,Y}(\sigma_{i-1})$ the probability that the tour $\Pi_i \subseteq V_i$ found by $\mathsf{ALG}_{\mathsf{Stoch\text{-}Cost}}$ contains Y vertices which can be selected within cost budget $6\gamma^i$.

The proof of Lemma 12 relies on the following Lemma 13, which says that if OPT selects Y vertices in V_i within total budget γ^i with probability at least 0.9, then we can select Y vertices in Π_i within cost budget $6\gamma^i$ with probability at least 0.2. The proof of Lemma 12 from Lemma 13 is standard, see Appendix C.

▶ **Lemma 13.** *For any phase $i \geq 0$, any target $Y \in \mathbb{Z}_{\geq 0}$ and any outcome σ_{i-1} of vertices visited in the previous $i-1$ phases, if $p^*_{i,Y}(\sigma_{i-1}) \geq 0.9$ then we have $p_{i,Y}(\sigma_{i-1}) \geq 0.2$.*

We need some notation to prove Lemma 13.

Notation. We say a vertex $v \in V$ is *qualified* for scale $j \in \{0, \cdots, \ell_i\}$ if its cost $C_v \leq \gamma^i/2^j$. For each scale $j \in \{0, \cdots, \ell_i\}$, denote $\pi^*_{i,j}$ the optimal Orienteering tour in $V_i \setminus \Pi_{i,j}$ with budget γ^i where each vertex $v \in V_i \setminus \Pi_{i,j}$ has profit $\mathbb{P}[C_v \leq \gamma^i/2^j]$. Define $T^*_{i,j} := \sum_{v \in \pi^*_{i,j}} \mathbb{P}[C_v \leq \gamma^i/2^j]$ and $T_{i,j} := \sum_{v \in \Pi_{i,j}} \mathbb{P}[C_v \leq \gamma^i/2^j]$ to be the total Orienteering profit of tour $\pi^*_{i,j}$ and $\Pi_{i,j}$, respectively. Denote $\overline{\Pi}_{i,j}(\mathsf{OPT}) \subseteq V_i \setminus (\Pi_{i,j} \cup \Pi_{i,j-1})$ the (random) set of vertices visited by OPT outside $\sigma_{i-1} \cup (\Pi_{i,j} \cup \Pi_{i,j-1})$ within total budget γ^i, and $\Pi_{i,j}(\mathsf{OPT}) \subseteq \Pi_{i,j} \cup \Pi_{i,j-1}$ the (random) set of vertices visited by OPT inside $\Pi_{i,j} \cup \Pi_{i,j-1}$ within total budget γ^i.

Intuition. We discuss our high-level proof strategy for Lemma 13. Our arguments again rely on the notion of "richness". Recall from above that $T_{i,j}$ denotes the Orienteering profit of tour $\Pi_{i,j}$ at scale j. We call a scale j "rich" if $T_{i,j} \geq Y$ and otherwise "poor". A scale j being poor roughly (but not quite) indicates that OPT cannot find enough low-cost vertices qualified for scale j outside the tour $\Pi_{i,j}$. A scale j being rich implies that $\Pi_{i,j}$ contains enough vertices that are qualified for scale j, which is an immediate consequence of Jogdeo-Samuels inequality (Theorem 3). We plan to find a critical scale j_{crit} that corresponds to the transition from rich to poor scales. Notice that such a critical scale j_{crit} might be different from the critical scale $\widetilde{j}_{\mathsf{crit}}$ found by our algorithm, but we show that it suffices to argue about j_{crit} since $\widetilde{j}_{\mathsf{crit}}$ is only better.

To argue about j_{crit}, we consider the tours corresponding to both scales j_{crit} and $j_{\mathsf{crit}}-1$. Roughly we use the poor scale j_{crit} to argue that OPT cannot find enough low-cost vertices outside these tours and we use the rich scale $j_{\mathsf{crit}}-1$ to argue that we have enough replacements for these low-cost vertices without paying too much cost. Our final analysis is a case analysis depending on whether the transition ever happens or not. The proof here is more involved than that in Section 4 as we also need to take into account the amount of $\mathsf{Orienteering}$ profit outside the $C = 6000$ repetitions of $\mathsf{ALG}_{\mathsf{Bicrit\text{-}Orient}}$ for each scale j.

Proof of Lemma 13. Recall that for any phase $i \geq 0$ and scale $j \in \{0, \cdots, \ell_i\}$, we defined $Y_{i,j}$ to be the maximum number of vertices that can be selected from $\Pi_{i,j} \cup \Pi_{i,j-1}$ with probability at least 0.2 within cost budget $3\gamma^i$. We show in the following that $p^*_{i,Y}(\sigma_{i-1}) \geq 0.9$ implies there exists a "critical" scale $j_{\mathsf{crit}} \in \{0, \cdots, \ell_i\}$ with $Y_{i,j_{\mathsf{crit}}} \geq Y$. This critical scale j_{crit} might be different from the critical scale $\widetilde{j}_{\mathsf{crit}}$ identified by $\mathsf{ALG}_{\mathsf{Stoch\text{-}Cost}}$. But since $\widetilde{j}_{\mathsf{crit}}$ maximizes $\widetilde{Y}_{i,j}$ among all scales $j \in \{0, \cdots, \ell_i\}$, it follows from property (1) in Lemma 11 that $\widetilde{Y}_{i,\widetilde{j}_{\mathsf{crit}}} \geq \widetilde{Y}_{i,j_{\mathsf{crit}}} \geq Y_{i,j_{\mathsf{crit}}} \geq Y$. Now using property (2) in Lemma 11 we have that $\widetilde{Y}_{i,\widetilde{j}_{\mathsf{crit}}} \geq Y$ vertices can be selected from $\Pi_{i,\widetilde{j}_{\mathsf{crit}}} \cup \Pi_{i,\widetilde{j}_{\mathsf{crit}}-1}$ within cost budget $6\gamma^i$ with probability at least 0.2. It follows that $p_{i,Y}(\sigma_{i-1}) \geq 0.2$. Therefore, the existence of a critical scale j_{crit} with $Y_{i,j_{\mathsf{crit}}} \geq Y$ would imply Lemma 13.

In the following, we consider three different cases and prove the existence of such a critical scale j_{crit} with $Y_{i,j_{\mathsf{crit}}} \geq Y$ in each case.

Case (1): (Scale 0 is poor) $T_{i,0} < Y$. We show in this case that $p^*_{i,Y}(\sigma_{i-1}) \geq 0.9$ implies that scale 0 is a "critical" scale with $Y_{i,0} \geq Y$. Notice that any vertex $v \in V_i$ not qualified for scale 0 has cost $C_v > \gamma^i$. Therefore, OPT cannot select any vertex that is not qualified for scale 0 within total budget γ^i.

We start by showing that $T^*_{i,0} \leq 0.1$. To prove this, we assume for the purpose of contradiction that $T^*_{i,0} > 0.1$. It follow from Lemma 8 that

$$\mathbb{P}\Big[\,\big|\{v \in \overline{\Pi}_{i,0}(\mathsf{OPT}) : C_v \leq \gamma^i\}\big| \leq 200T^*_{i,0}\Big] \;\geq\; 0.9. \tag{8}$$

From Lemma 9 we have that

$$\sum_{v \in \Pi_{i,0} \setminus \Pi_{i,0}(\mathsf{OPT})} \mathbb{P}[C_v \leq \gamma^i] \quad\geq\quad (6000-1)\cdot(T^*_{i,0}-\epsilon)-\epsilon \quad\geq\quad 5000T^*_{i,0}.$$

So it follows from Chernoff bound (Theorem 5) that

$$\mathbb{P}\Big[\,\big|\{v \in \Pi_{i,0} \setminus \Pi_{i,0}(\mathsf{OPT}) : C_v \leq \gamma^i\}\big| \geq 500T^*_{i,0}\Big] \;\geq\; 0.9. \tag{9}$$

Since $T_{i,0} < Y$, from Theorem 3 we have that

$$\mathbb{P}\Big[\,\big|\{v \in \Pi_{i,0} : C_v \leq \gamma^i\}\big| \leq Y\Big] \;\geq\; 0.5. \tag{10}$$

Now we count the number of vertices found by OPT that are qualified for scale 0 within total budget γ^i. It follows from union bound that with probability at least 0.2, all three events in (8), (9) and (10) hold, in which case OPT finds at most $Y - 300T^*_{i,0}$ vertices qualified for scale 0 within total budget γ^i. Therefore, in order to select at least Y vertices, OPT needs to select vertices that are not qualified for scale 0 within total budget γ^i, which is a contradiction to the assumption that $p^*_{i,Y}(\sigma_{i-1}) \geq 0.9$.

Therefore we must have $T^*_{i,0} \le 0.1$. Applying Markov's inequality, the probability that OPT finds any vertex qualified for scale 0 in $V_i \setminus \Pi_{i,0}$ is upper bounded by $T^*_{i,0} \le 0.1$. When this happens and when OPT selects Y vertices within total budget γ^i, all vertices selected by OPT are from $\Pi_{i,0}$. Therefore, with probability at least $p^*_{i,Y}(\sigma_{i-1}) - 0.1 \ge 0.8$, we can select Y vertices from $\Pi_{i,0}$ within cost budget γ^i which implies that $Y_{i,0} \ge Y$.

Case (2): (All scales are rich) $T_{i,j} \ge Y$ for every scale $j = 0, \cdots, \ell_i$. In this case we show that ℓ_i is a "critical" scale with $Y_{i,\ell_i} \ge Y$. Notice that selecting any Y vertices qualified for scale ℓ_i has cost at most $Y \cdot \gamma^i/2^{\ell_i} \le 2 \le 6\gamma^i$. Since $T_{i,\ell_i} \ge Y$, it follows from Theorem 3 that with probability no less than 0.5, at least Y vertices in Π_{i,ℓ_i} are qualified for scale ℓ_i (notice we don't even need the tour Π_{i,ℓ_i-1} in this case). This implies that $Y_{i,\ell_i} \ge Y$.

Case (3): (Transition from rich to poor scale at j_{crit}) $T_{i,0} \ge Y$ but $T_{i,j} < Y$ for some $j \in [\ell_i]$. In this case, let $j_{\mathsf{crit}} = \arg\min_j \{j \in [\ell_i] : T_{i,j} < Y\}$. We show in the following that j_{crit} is a "critical" scale with $Y_{i,j_{\mathsf{crit}}} \ge Y$. To prove this, we show that the following two events happen with constant probability:

1. $T_{i,j_{\mathsf{crit}}} < Y$ implies that OPT doesn't find enough vertices *qualified* for scale j_{crit}. This gives a lower bound on the cost of the set of vertices selected by OPT within total budget γ^i.
2. $T_{i,j_{\mathsf{crit}}-1} \ge Y$ implies that $\mathsf{ALG}_{\mathsf{Stoch\text{-}Cost}}$ finds enough vertices *qualified* for scale $j_{\mathsf{crit}} - 1$. This can be used to upper bound the cost of $\mathsf{ALG}_{\mathsf{Stoch\text{-}Cost}}$.

We first consider the sub-case where $T^*_{i,j_{\mathsf{crit}}} \le 0.1$. This is the case where not many vertices qualified for scale j_{crit} can be found outside $\Pi_{i,j_{\mathsf{crit}}} \cup \Pi_{i,j_{\mathsf{crit}}-1}$. In this case, we have $\sum_{v \in \overline{\Pi}_{i,j_{\mathsf{crit}}}(\mathsf{OPT})} \mathbb{P}[C_v \le \gamma^i/2^{j_{\mathsf{crit}}}] \le T^*_{i,j_{\mathsf{crit}}}$. It follows from Markov's inequality that with probability at least 0.9, OPT finds no vertex in $V_i \setminus (\Pi_{i,j_{\mathsf{crit}}} \cup \Pi_{i,j_{\mathsf{crit}}-1})$ that is qualified for scale j_{crit}, in which case any vertex $v \in V_i \setminus (\Pi_{i,j_{\mathsf{crit}}} \cup \Pi_{i,j_{\mathsf{crit}}-1})$ selected by OPT has cost $C_v > \gamma^i/2^{j_{\mathsf{crit}}}$. Since $T_{i,j_{\mathsf{crit}}-1} \ge Y$, it follows from Theorem 3 that with probability at least 0.5, we have $\left|\left\{v \in \Pi_{i,j_{\mathsf{crit}}} \cup \Pi_{i,j_{\mathsf{crit}}-1} : C_v \le \gamma^i/2^{j_{\mathsf{crit}}-1}\right\}\right| \ge Y$. Furthermore, $p^*_Y \ge 0.9$ implies that with probability at least 0.9, OPT selects Y vertices within total budget γ^i. It follows from union bound that all three events above happen with probability at least 0.2, in which case we can select Y vertices from $\Pi_{i,j_{\mathsf{crit}}} \cup \Pi_{i,j_{\mathsf{crit}}-1}$ within cost budget $2\gamma^i$. This implies that $Y_{i,j_{\mathsf{crit}}} \ge Y$.

Now we deal with the other sub-case where $T^*_{i,j_{\mathsf{crit}}} > 0.1$. This represents the situation where many vertices qualified for scale j_{crit} can be found outside $\Pi_{i,j_{\mathsf{crit}}} \cup \Pi_{i,j_{\mathsf{crit}}-1}$. Roughly, Lemma 8 implies in this case that OPT finds at most $200T^*_{i,j_{\mathsf{crit}}}$ low-cost vertices in $V_i \setminus (\Pi_{i,j_{\mathsf{crit}}} \cup \Pi_{i,j_{\mathsf{crit}}-1})$. In general, these vertices might have very tiny cost. However, we show in the following Claim 14 that increasing the cost of each one of these low-cost vertices to $\gamma^i/2^{j_{\mathsf{crit}}}$ will increase their total cost by at most $O(\gamma^i)$. This allows us to lower bound the cost of the set of vertices selected by OPT within total budget γ^i.

▷ Claim 14. We have $200T^*_{i,j_{\mathsf{crit}}} \cdot \gamma^i/2^{j_{\mathsf{crit}}} \le \gamma^i/2$.

Before proving Claim 14, we complete the proof of Lemma 13. Since $T^*_{i,j_{\mathsf{crit}}} > 0.1$, from Lemma 8 we have

$$\mathbb{P}\left[\left|\left\{v \in \overline{\Pi}_{i,j_{\mathsf{crit}}}(\mathsf{OPT}) : C_v \le \gamma^i/2^{j_{\mathsf{crit}}}\right\}\right| \le 200T^*_{i,j_{\mathsf{crit}}}\right] \ \ge \ 0.9. \tag{11}$$

Since $T_{i,j_{\mathsf{crit}}-1} \geq Y$, it follows from Theorem 19 that

$$\mathbb{P}\Big[\,\big|\big\{v \in \Pi_{i,j_{\mathsf{crit}}} \cup \Pi_{i,j_{\mathsf{crit}}-1} : C_v \leq \gamma^i/2^{j_{\mathsf{crit}}-1}\big\}\big| \geq Y\Big] \;\geq\; 0.5. \tag{12}$$

Together with the assumption that $p^*_{i,Y} \geq 0.9$, we have from union bound that with probability at least 0.2, both events in (11) and (12) hold and that OPT selects at least Y vertices within total budget γ^i. When all three events happen, we can replace $Y - |\Pi_{i,j_{\mathsf{crit}}}(\mathsf{OPT})|$ vertices in $\overline{\Pi}_{i,j_{\mathsf{crit}}}(\mathsf{OPT})$ by $Y - |\Pi_{i,j_{\mathsf{crit}}}(\mathsf{OPT})|$ vertices in $(\Pi_{i,j_{\mathsf{crit}}} \cup \Pi_{i,j_{\mathsf{crit}}-1}) \setminus \Pi_{i,j_{\mathsf{crit}}}(\mathsf{OPT})$ that are qualified for scale $j_{\mathsf{crit}}-1$. Claim 14 together with (11) imply that after such replacements, we reach a subset of Y vertices in $\Pi_{i,j_{\mathsf{crit}}} \cup \Pi_{i,j_{\mathsf{crit}}-1}$ with total cost at most $2(\gamma^i + \gamma^i/2) = 3\gamma^i$. So we conclude that with probability at least 0.2, we can select at least Y vertices from $\Pi_{i,j_{\mathsf{crit}}} \cup \Pi_{i,j_{\mathsf{crit}}-1}$ within cost budget $3\gamma^i$. This implies that $Y_{i,j_{\mathsf{crit}}} \geq Y$ and finishes the proof of Lemma 13. ◀

Now we are left to prove Claim 14.

Proof of Claim 14. We consider the tour $\Pi_{i,j_{\mathsf{crit}}}$. Denote respectively $\Pi'_{i,j_{\mathsf{crit}}}(\mathsf{OPT}) \subseteq \Pi_{i,j_{\mathsf{crit}}}$ and $\overline{\Pi}'_{i,j_{\mathsf{crit}}}(\mathsf{OPT}) \subseteq V_i \setminus \Pi_{i,j_{\mathsf{crit}}}$ the (random) set of vertices visited by OPT inside and outside $\Pi_{i,j_{\mathsf{crit}}}$ within total budget γ^i. Since $T_{i,j_{\mathsf{crit}}} < Y$, it follows from Theorem 3 that

$$\mathbb{P}\Big[\,\big|\big\{v \in \Pi_{i,j_{\mathsf{crit}}} : C_v \leq \gamma^i/2^{j_{\mathsf{crit}}}\big\}\big| \leq Y\Big] \;\geq\; 0.5. \tag{13}$$

Since $T^*_{i,j_{\mathsf{crit}}} > 0.1$, Lemma 8 gives

$$\mathbb{P}\Big[\,\Big|\Big\{v \in \overline{\Pi}'_{i,j_{\mathsf{crit}}}(\mathsf{OPT}) : C_v \leq \gamma^i/2^{j_{\mathsf{crit}}}\Big\}\Big| \leq 200T^*_{i,j_{\mathsf{crit}}}\Big] \;\geq\; 0.9. \tag{14}$$

Lemma 9 followed by Chernoff bound gives

$$\mathbb{P}\Big[\,\big|\big\{v \in \Pi_{i,j_{\mathsf{crit}}} \setminus \Pi'_{i,j_{\mathsf{crit}}}(\mathsf{OPT}) : C_v \leq \gamma^i/2^{j_{\mathsf{crit}}}\big\}\big| \geq 600T^*_{i,j_{\mathsf{crit}}}\Big] \;\geq\; 0.9. \tag{15}$$

From the assumption that $p^*_{i,Y} \geq 0.9$, we have

$$\mathbb{P}\Big[\,\big\{\mathsf{OPT} \text{ selects at least } Y \text{ vertices within total budget } \gamma^i\big\}\,\Big] \;\geq\; 0.9. \tag{16}$$

By union bound, all four events in (13)-(16) happen with positive probability, in which case OPT selects at least $400T^*_{i,j_{\mathsf{crit}}}$ vertices that are not qualified for scale j_{crit} within total budget γ^i. This implies that $400T^*_{i,j_{\mathsf{crit}}} \cdot \gamma^i/2^{j_{\mathsf{crit}}} \leq \gamma^i$ from which Claim 14 immediately follows. ◁

5.3 The Dynamic Programming Sub-procedure $\mathsf{ALG}_{\mathsf{DP}}$

In this section, we give our dynamic program $\mathsf{ALG}_{\mathsf{DP}}$ and prove Lemma 11 restated below for convenience.

▶ **Lemma 11.** *For any phase $i \geq 0$ and any scale $j \in \{0, \cdots, \ell_i\}$, the approximate value $\widetilde{Y}_{i,j}$ computed in Step 15 of $\mathsf{ALG}_{\mathit{Stoch\text{-}Cost}}$ satisfies that (1) $\widetilde{Y}_{i,j} \geq Y_{i,j}$, and (2) $\widetilde{Y}_{i,j}$ vertices can be selected from $\Pi_{i,j} \cup \Pi_{i,j-1}$ within cost budget $6\gamma^i$ with probability at least 0.2.*

Recall, our dynamic program $\mathsf{ALG}_{\mathsf{DP}}$ is used to approximately compute the maximum number of vertices that can be selected from a certain tour with probability at least 0.2 within cost budget $3\gamma^i$. Essentially, $\mathsf{ALG}_{\mathsf{DP}}$ solves the following sub-problem: We are given a target $T \in \mathbb{Z}_{\geq 0}$, a budget $B \geq 0$ and n independent non-negative stochastic costs. The goal is to find the probability $P_{T,B}$ that there exists a subset S of size T with sum of its costs at most B. Since this general problem is NP-hard, we give a dynamic program that finds something between $P_{T,B}$ and $P_{T,2B}$. Lemma 11 follows immediately from the following Lemma 15.

▶ **Lemma 15.** *Given n independent non-negative random variables $V = \{C_1, C_2, ..., C_n\}$, a target $T \in \mathbb{Z}_{\geq 0}$ and a budget $B \geq 0$. Let $P_{T,B}$ denote the probability that there exists a subset $S \subseteq V$ of size at least T and $\sum_{i \in S} C_i \leq B$. Then there's an efficient dynamic programming $\mathsf{ALG}_{DP}(T, B, V)$ that outputs a value $\widetilde{P}_{T,B}$ s.t. $P_{T,B} \leq \widetilde{P}_{T,B} \leq P_{T,2B}$.*

Proof of Lemma11. We first prove property (1). Recall that $Y_{i,j}$ is the maximum number of vertices that can be selected from $\Pi_{i,j} \cup \Pi_{i,j-1}$ with probability at least 0.2 within cost budget $3\gamma^i$, and $\widetilde{Y}_{i,j}$ is the largest integer such that $\mathsf{ALG}_{\mathsf{DP}}(\widetilde{Y}_{i,j}, 3\gamma^i, \Pi_{i,j} \cup \Pi_{i,j-1}) \geq 0.2$. Consider the set of stochastic costs in $\Pi_{i,j} \cup \Pi_{i,j-1}$. By definition, we have $P_{Y_{i,j}, 3\gamma^i} \geq 0.2$. It follows from Lemma 15 that $\mathsf{ALG}_{\mathsf{DP}}(Y_{i,j}, 3\gamma^i, \Pi_{i,j} \cup \Pi_{i,j-1}) \geq 0.2$. This implies that $\widetilde{Y}_{i,j} \geq Y_{i,j}$ and proves property (1).

Now we prove property (2). Applying Lemma 15 we have $P_{\widetilde{Y}_{i,j}, 6\gamma^i} \geq \mathsf{ALG}_{\mathsf{DP}}(\widetilde{Y}_{i,j}, 3\gamma^i, \Pi_{i,j} \cup \Pi_{i,j-1}) \geq 0.2$. It follows that $\widetilde{Y}_{i,j}$ vertices can be selected from $\Pi_{i,j} \cup \Pi_{i,j-1}$ within cost budget $6\gamma^i$ with probability at least 0.2. This establishes property (2) and finishes the proof of Lemma 11. ◀

Proof of Lemma 15. We begin by discretizing each C_i to be $\overline{C}_i := \lfloor C_i \cdot n/B \rfloor \in \mathbb{N}$ and define $\overline{P}_{T,n}$ the probability that there exists a subset $S \subseteq V$ s.t. $|S| \geq T$ and $\sum_{i \in S} \overline{C}_i \leq n$. Notice that $\sum_{i \in S} \overline{C}_i \leq n$ implies that $\sum_{i \in S} C_i \leq 2B$. Therefore we have $P_{T,B} \leq \overline{P}_{T,n} \leq P_{T,2B}$. In the following, we give a dynamic programming $\mathsf{ALG}_{\mathsf{DP}}$ that computes the value $\overline{P}_{T,n}$ and we will set $\widetilde{P}_{T,B}$ in the statement of the lemma to be $\overline{P}_{T,n}$. Assume without loss of generality that $\overline{C}_i \leq n+1$ as one can truncate the distribution of $\overline{C}_i$ at $(n+1)$ without changing $\overline{P}_{T,n}$.

Denote $A(i,j)$ the jth smallest value among the first i random variables. We build a DP table where each entry $P[i,j,\ell,m]$ (for $i \in \{1, \cdots, n\}, j \in \{1, \cdots, T\}$, $\ell \in \{0, \cdots, n\}$ and $m \in \{0, \cdots, n\}$) denotes the probability that the smallest j values among the first i random variables sum up to ℓ and the jth smallest value among the first i random variables is equal to m, i.e. $\sum_{s=1}^{j} A[i,s] = \ell$ and $A[i,j] = m$.

Initial values: $\mathsf{ALG}_{\mathsf{DP}}$ initializes certain entries of the DP table as follows.

- **Case 1 (impossible events):** set $P[i,j,\ell,m]$ to be 0 if $j > i$, $m > \ell$ or $m \cdot j < \ell$.
- **Case 2 ($j = 1$):** set $P[i,1,\ell,m] = \prod_{s \in [i]} \mathbb{P}[\overline{C}_s \geq \ell] - \prod_{s \in [i]} \mathbb{P}[\overline{C}_s > \ell]$ if $\ell = m$, and 0 otherwise.

Note that all the entries corresponding to $i = 1$ are already included in the two cases above.

Recursion: $\mathsf{ALG}_{\mathsf{DP}}$ uses the following recursion.

$$P[i,j,\ell,m] = \mathbb{P}[\overline{C_i} > m] \cdot P[i-1,j,\ell,m] + \sum_{u=0}^{m-1} \mathbb{P}[\overline{C_i} = u] \cdot P[i-1,j-1,\ell-u,m]$$
$$+\mathbb{P}[\overline{C_i} = m] \cdot \Big(\sum_{u=0}^{m} P[i-1,j-1,\ell-m,m-u] - \sum_{u=1}^{m} P[i-1,j,\ell-u,m-u] \Big). \tag{17}$$

Output: after computing all the entries of the DP table, $\mathsf{ALG}_{\mathsf{DP}}$ outputs $\overline{P}_{T,n}$ that equals $\sum_{\ell=0}^{n} \sum_{m=0}^{\ell} P[n,T,\ell,m]$.

Now we prove the correctness of $\mathsf{ALG}_{\mathsf{DP}}$. Given the definition of $P[i,j,\ell,m]$, we can immediately verify that the assignment of initial values and the final output are correct if all the entries of the DP table computed from (17) are also correct. To see the correctness of the recursion, we consider the outcome of $\overline{C}_i$. When $\overline{C}_i > m$, in order to satisfy $\sum_{s=1}^{j} A[i,s] = \ell$

and $A[i,j] = m$, one must have that $\overline{C}_i$ is not in the j smallest values among the first i random variables. This verifies the first term in (17). When $\overline{C}_i = u < m$, in order to satisfy $\sum_{s=1}^{j} A[i,s] = \ell$ and $A[i,j] = m$, one must have that $\overline{C}_i$ is one of the jth smallest values among the first i random variables. Also notice that in this case, the $(j-1)$th smallest value among the first $(i-1)$ random variables is still m and that $\sum_{s=1}^{j-1} A[i-1,s] = \ell - \overline{C}_i$. This gives the second term in (17).

Now we verify the last term in (17, which corresponds to the case where $\overline{C}_i = m$. In this case, we might as well select $\overline{C}_i$ as one of the j smallest values among the first i random variables. In order to satisfy $\sum_{s=1}^{j} A[i,s] = \ell$ and $A[i,j] = m$, we need the smallest $(j-1)$ values among the first $(i-1)$ random variables to sum up to $\ell - m$ and the $(j-1)$th smallest value to be at most m, i.e. $\sum_{s=1}^{j-1} A[i,s] = \ell - m$ and $A[i,j-1] \leq m$. The probability of this event is exactly $\sum_{u=0}^{m} P[i-1, j-1, \ell-m, m-u]$. However, in order to ensure that $A[i,j] = m$, we also need the jth smallest value among the first $(i-1)$ random variables to be at least m (i.e. $A[i-1,j] \geq m$) and the outcomes that don't satisfy this condition needs to be excluded from the previous event. Putting everything together, we have the following:

$$\begin{aligned}
&\mathbb{P}\Big\{ \sum_{s=1}^{j-1} A[i-1,s] = \ell - m, A[i-1,j-1] \leq m, A[i-1,j] \geq m \Big\} \\
&= \mathbb{P}\Big\{ \sum_{s=1}^{j-1} A[i-1,s] = \ell - m, A[i-1,j-1] \leq m \Big\} \\
&\qquad - \mathbb{P}\Big\{ \sum_{s=1}^{j-1} A[i-1,s] = \ell - m, A[i-1,j-1] \leq m, A[i-1,j] < m \Big\} \\
&= \sum_{u=0}^{m} P[i-1, j-1, \ell-m, m-u] - \sum_{u=1}^{m} P[i-1, j, \ell-u, m-u].
\end{aligned}$$

This immediately gives the last term in (17) and finishes the proof of Lemma 15. ◀

References

1. Sanjeev Arora and George Karakostas. 2+ ε approximation algorithm for the k-MST problem. In *11th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 754–759. SIAM, 2000.
2. Sunil Arya and Hariharan Ramesh. A 2.5-factor approximation algorithm for the k-MST problem. *Information Processing Letters*, 65(3):117–118, 1998.
3. Arash Asadpour and Hamid Nazerzadeh. Maximizing stochastic monotone submodular functions. *Management Science*, 62(8):2374–2391, 2016.
4. Baruch Awerbuch, Yossi Azar, Avrim Blum, and Santosh Vempala. New Approximation Guarantees for Minimum-Weight k-Trees and Prize-Collecting Salesmen. *SIAM J. Comput.*, 28(1):254–262, 1998.
5. Nikhil Bansal, Anupam Gupta, Jian Li, Julián Mestre, Viswanath Nagarajan, and Atri Rudra. When LP Is the Cure for Your Matching Woes: Improved Bounds for Stochastic Matchings. *Algorithmica*, 63(4):733–762, 2012.
6. Nikhil Bansal and Viswanath Nagarajan. On the Adaptivity Gap of Stochastic Orienteering. In *IPCO*, pages 114–125, 2014.
7. Anand Bhalgat, Ashish Goel, and Sanjeev Khanna. Improved Approximation Results for Stochastic Knapsack Problems. In *SODA*, pages 1647–1665, 2011.
8. Avrim Blum, R. Ravi, and Santosh Vempala. A Constant-factor Approximation Algorithm for the k MST Problem (Extended Abstract). In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, pages 442–448, 1996.

9 Domagoj Bradac, Sahil Singla, and Goran Zuzic. (Near) Optimal Adaptivity Gaps for Stochastic Multi-Value Probing. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM*, pages 49:1–49:21, 2019.

10 Chandra Chekuri, Nitish Korula, and Martin Pál. Improved algorithms for orienteering and related problems. *ACM Transactions on Algorithms (TALG)*, 8(3):23, 2012.

11 Brian C. Dean, Michel X. Goemans, and Jan Vondrák. Approximating the stochastic knapsack problem: The benefit of adaptivity. In *Foundations of Computer Science, 2004. Proceedings. 45th Annual IEEE Symposium on*, pages 208–217. IEEE, 2004.

12 Amol Deshpande, Lisa Hellerstein, and Devorah Kletenik. Approximation Algorithms for Stochastic Submodular Set Cover with Applications to Boolean Function Evaluation and Min-Knapsack. *ACM Trans. Algorithms*, 12(3):42:1–42:28, 2016.

13 Alina Ene, Viswanath Nagarajan, and Rishi Saket. Approximation Algorithms for Stochastic k-TSP. In *37th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

14 David A Freedman. On tail probabilities for martingales. *the Annals of Probability*, 3(1):100–118, 1975.

15 Hao Fu, Jian Li, and Pan Xu. A PTAS for a Class of Stochastic Dynamic Programs. In *45th International Colloquium on Automata, Languages, and Programming (ICALP 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

16 N. Garg. A 3-approximation for the Minimum Tree Spanning K Vertices. In *Proceedings of the 37th Annual Symposium on Foundations of Computer Science*, FOCS '96, pages 302–, 1996.

17 Naveen Garg. Saving an epsilon: a 2-approximation for the k-MST problem in graphs. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 396–402. ACM, 2005.

18 Michel X. Goemans and Jan Vondrák. Stochastic Covering and Adaptivity. In *LATIN 2006: Theoretical Informatics, 7th Latin American Symposium, Proceedings*, pages 532–543, 2006.

19 Sudipto Guha and Kamesh Munagala. Multi-armed Bandits with Metric Switching Costs. In *ICALP*, pages 496–507, 2009.

20 Sudipto Guha and Kamesh Munagala. Adaptive uncertainty resolution in bayesian combinatorial optimization problems. *ACM Transactions on Algorithms (TALG)*, 8(1):1, 2012.

21 Anupam Gupta, Haotian Jiang, Ziv Scully, and Sahil Singla. The Markovian Price of Information. In *Integer Programming and Combinatorial Optimization, IPCO*, pages 233–246, 2019.

22 Anupam Gupta, Ravishankar Krishnaswamy, Marco Molinaro, and R. Ravi. Approximation Algorithms for Correlated Knapsacks and Non-martingale Bandits. In *FOCS*, pages 827–836, 2011.

23 Anupam Gupta, Ravishankar Krishnaswamy, Viswanath Nagarajan, and R. Ravi. Approximation Algorithms for Stochastic Orienteering. In *SODA*, 2012. URL: `http://dl.acm.org/citation.cfm?id=2095116.2095237`.

24 Anupam Gupta and Viswanath Nagarajan. A Stochastic Probing Problem with Applications. In *IPCO*, pages 205–216, 2013.

25 Anupam Gupta, Viswanath Nagarajan, and Sahil Singla. Adaptivity Gaps for Stochastic Probing: Submodular and XOS Functions. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1688–1702. SIAM, 2017.

26 Zhihao Jiang and Haoyu Zhao. An FPTAS for Stochastic Unbounded Min-Knapsack Problem. In *International Workshop on Frontiers in Algorithmics*, pages 121–132. Springer, 2019.

27 Kumar Jogdeo and Stephen M Samuels. Monotone convergence of binomial probabilities and a generalization of Ramanujan's equation. *The Annals of Mathematical Statistics*, 39(4):1191–1195, 1968.

28 Robert Kleinberg, Bo Waggoner, and Glen Weyl. Descending Price Optimally Coordinates Search. *arXiv preprint*, 2016. `arXiv:1603.07682`.

29 Will Ma. Improvements and Generalizations of Stochastic Knapsack and Multi-Armed Bandit Approximation Algorithms: Extended Abstract. In *SODA*, pages 1154–1163, 2014.

30 Sridhar Rajagopalan and V Vazirani. Logarithmic approximation of minimum weight k trees. *Unpublished manuscript*, 1995.

31 Ramamurthy Ravi, Ravi Sundaram, Madhav V Marathe, Daniel J Rosenkrantz, and Sekharipuram S Ravi. Spanning trees-short or small. *SIAM Journal on Discrete Mathematics*, 9(2):178–200, 1996.

32 Sahil Singla. *Combinatorial Optimization Under Uncertainty: Probing and Stopping-Time Algorithms*. PhD thesis, Carnegie Mellon University, 2018.

33 Sahil Singla. The Price of Information in Combinatorial Optimization. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 2018.

34 Martin L. Weitzman. Optimal search for the best alternative. *Econometrica: Journal of the Econometric Society*, pages 641–654, 1979.

A Missing Proofs in Section 2

▶ **Lemma 6** (Bi-criteria Orienteering). *There is an efficient algorithm* $\mathsf{ALG}_{\textit{Bicrit-Orient}}$ *that finds a tour of length* $O(1) \cdot B$ *while collecting at least* $(\mathsf{OPT}_{\textit{Orient}} - \epsilon)$ *profit, where* $\epsilon = 1/\mathrm{poly}(n)$ *can be made arbitrarily small.*

Proof of Lemma 6. Assume without loss of generality that $\mathsf{OPT}_{\mathsf{Orient}} > 0$. Denote $\rho = 3$ the approximation factor of k-TSP algorithm $\mathsf{ALG}_{k\text{-}\mathsf{TSP}}$ from [8]. Denote $R_{\max} := \max_{v \in V} R_v$ and $R_{\min} := \min_{v \in V} R_v$ the maximum and minimum profit in the Orienteering instance. Notice, $\mathsf{OPT}_{\mathsf{Orient}} \in [R_{\min}, n \cdot R_{\max}]$.

$\mathsf{ALG}_{\mathsf{Bicrit\text{-}Orient}}$ applies binary search in $[R_{\min}, n \cdot R_{\max}]$, starting with profit target $(R_{\min} + n \cdot R_{\max})/2$. For each profit target λ, $\mathsf{ALG}_{\mathsf{Bicrit\text{-}Orient}}$ runs $\mathsf{ALG}_{k\text{-}\mathsf{TSP}}$ with target reward λ to obtain a tour Π_λ whose length is denoted as $\ell(\Pi_\lambda)$. $\mathsf{ALG}_{\mathsf{Bicrit\text{-}Orient}}$ performs binary search over $\lambda \in [R_{\min}, n \cdot R_{\max}]$ until finding two values $\lambda_l < \lambda_h \le \lambda_l + \epsilon$ such that $\ell(\Pi_{\lambda_l}) \le \rho B$ and $\ell(\Pi_{\lambda_h}) > \rho B$, in which case $\mathsf{ALG}_{\mathsf{Bicrit\text{-}Orient}}$ returns the tour Π_{λ_l}. Here we assumed without loss of generality that $\ell(\Pi_{n \cdot R_{\max}}) > \rho B$ as otherwise $\mathsf{ALG}_{\mathsf{Bicrit\text{-}Orient}}$ can simply return the tour $\Pi_{n \cdot R_{\max}}$. Notice that $\ell(\Pi_{\lambda_h}) > \rho B$ implies that $\mathsf{OPT}_{\mathsf{Orient}} < \lambda_h$ and therefore $\mathsf{ALG}_{\mathsf{Bicrit\text{-}Orient}}$ finds reward at least $\lambda_l \ge \lambda_h - \epsilon > \mathsf{OPT}_{\mathsf{Orient}} - \epsilon$. The length of the tour found by $\mathsf{ALG}_{\mathsf{Bicrit\text{-}Orient}}$ is $\ell(\Pi_{\lambda_l}) \le \rho B = O(1) \cdot B$. This finishes the proof of Lemma 6. ◀

B Missing Proofs in Section 3

▶ **Lemma 7** (Key Lemma). *If for some universal constants* $C > 0$, $\gamma > 1$, *any phase* $i \ge 1$, *and any possible* σ_{i-1}, *the algorithm* $\mathsf{ALG}_{\textit{Meta}}$ *satisfies*

$$u_i(\sigma_{i-1}) \le C \cdot u_i^*(\sigma_{i-1}) + \frac{u_{i-1}(\sigma_{i-1})}{\gamma^2},$$

then $\mathsf{ALG}_{\textit{Meta}}$ *is a non-adaptive* $O(1)$*-approximation algorithm.*

Proof of Lemma 7. For any phase $i \ge 0$, denote u_i the probability that $\mathsf{ALG}_{\mathsf{Meta}}$ enters phase $i+1$ in the probing stage and u_i^* the probability that OPT has cost more than γ^i. Taking expectation over σ_{i-1} for the pre-condition of Lemma 7, we have $u_i \le C \cdot u_i^* + u_{i-1}/\gamma^2$. It follows that

$$\sum_{i \ge 1} u_i \cdot \gamma^i \le C \cdot \sum_{i \ge 1} u_i^* \cdot \gamma^i + u_0/\gamma + 1/\gamma \cdot \sum_{i \ge 1} u_i \cdot \gamma^i,$$

which gives

$$(1-1/\gamma)\cdot\sum_{i\geq 1}u_i\cdot\gamma^i \;\leq\; O(1)\cdot\sum_{i\geq 1}u_i^*\cdot\gamma^i+1/\gamma. \tag{18}$$

We also notice that

$$\mathsf{OPT} \;\geq\; \sum_{i\geq 0}(u_i^*-u_{i+1}^*)\cdot\gamma^i \;=\; (1-1/\gamma)\cdot\sum_{i\geq 1}u_i^*\cdot\gamma^i+1,$$

and that

$$\mathsf{ALG}_{\mathsf{Meta}} \;\leq\; O(1)\cdot\sum_{i\geq 0}(u_i-u_{i+1})\cdot\gamma^{i+1} \;=\; O(1)\cdot\sum_{i\geq 0}u_i\cdot\gamma^i.$$

It follows from (18) that $\mathsf{ALG}_{\mathsf{Meta}}\leq O(1)\cdot\mathsf{OPT}$. This finishes the proof of Lemma 7. ◀

C Missing Proofs in Section 5

▶ **Lemma 12.** *For $\gamma=1.1$, any phase $i>0$ in the probing stage of Stoch-Cost k-TSP satisfies*

$$u_i(\sigma_{i-1})\leq 100u_i^*(\sigma_{i-1})+\frac{u_{i-1}(\sigma_{i-1})}{\gamma^2}. \tag{7}$$

Proof of Lemma 12. We fix any outcome σ_{i-1} of vertices visited by $\mathsf{ALG}_{\mathsf{Stoch\text{-}Cost}}$ in the first $i-1$ phases of its probing stage. The lemma trivially holds in the case where $u_i^*(\sigma_{i-1})\geq 0.01$ as we have $100u_i^*(\sigma_{i-1})\geq 1$. If $u_{i-1}(\sigma_{i-1})=0$ which means that $\mathsf{ALG}_{\mathsf{Stoch\text{-}Cost}}$ already selects k vertices before entering phase i in the probing stage, then $u_i(\sigma_{i-1})=0$ and again the lemma trivially holds. We therefore assume that $u_i^*(\sigma_{i-1})<0.01$ and that $u_{i-1}(\sigma_{i-1})=1$. Now proving Lemma 12 is equivalent to proving

$$u_i(\sigma_{i-1})\leq 100u_i^*(\sigma_{i-1})+1/\gamma^2. \tag{19}$$

Denote $k(\sigma_{i-1})$ the remaining target at the beginning of phase i in the probing stage of $\mathsf{ALG}_{\mathsf{Stoch\text{-}Cost}}$. We first consider Selection-Process 1 and denote $N_{\mathsf{old}}(\sigma_{i-1})$ the number of vertices selected from σ_{i-1} in this process. We assume without loss of generality that $N_{\mathsf{old}}(\sigma_{i-1})<k(\sigma_{i-1})$, as otherwise our algorithm has already selected k vertices after Selection-Process 1 and (19) immediately follows. Since Selection-Process 1 uses cost budget γ^i to select as many unselected vertices from σ_{i-1} as possible, OPT can select at most $N_{\mathsf{old}}(\sigma_{i-1})+k-k(\sigma_{i-1})$ vertices from σ_{i-1} within total budget γ^i. Denote $N_{\mathsf{new}}(\sigma_{i-1}):=k(\sigma_{i-1})-N_{\mathsf{old}}(\sigma_{i-1})$ the remaining target for $\mathsf{ALG}_{\mathsf{Stoch\text{-}Cost}}$ after Selection-Process 1. It follows that in order to select k vertices within total budget γ^i, OPT needs to select at least $N_{\mathsf{new}}(\sigma_{i-1})$ vertices from V_i within total budget γ^i. Therefore, $u_i^*(\sigma_{i-1})<0.01$ implies that OPT selects at least $N_{\mathsf{new}}(\sigma_{i-1})$ vertices from V_i within total budget γ^i with probability at least 0.99. Applying Lemma 13 with $Y=N_{\mathsf{new}}(\sigma_{i-1})$, it follows that our algorithm finds at least $N_{\mathsf{new}}(\sigma_{i-1})$ vertices from V_i which can be selected within cost budget $6\gamma^i$ with probability at least 0.2. This implies that $u_i(\sigma_{i-1})\leq 0.8\leq 1/\gamma^2$ and (19) is established. ◀

Strategic Payments in Financial Networks

Nils Bertschinger
Systemic Risk Group, Frankfurt Institute of Advanced Studies, Germany
Institute for Computer Science, Goethe University Frankfurt, Germany
bertschinger@fias.uni-frankfurt.de

Martin Hoefer
Institute for Computer Science, Goethe University Frankfurt, Germany
mhoefer@cs.uni-frankfurt.de

Daniel Schmand
Institute for Computer Science, Goethe University Frankfurt, Germany
schmand@cs.uni-frankfurt.de

Abstract

In their seminal work on systemic risk in financial markets, Eisenberg and Noe [13] proposed and studied a model with n firms embedded into a network of debt relations. We analyze this model from a game-theoretic point of view. Every firm is a rational agent in a directed graph that has an incentive to allocate payments in order to clear as much of its debt as possible. Each edge is weighted and describes a liability between the firms. We consider several variants of the game that differ in the permissible payment strategies. We study the existence and computational complexity of pure Nash and strong equilibria, and we provide bounds on the (strong) prices of anarchy and stability for a natural notion of social welfare. Our results highlight the power of financial regulation – if payments of insolvent firms can be centrally assigned, a socially optimal strong equilibrium can be found in polynomial time. In contrast, worst-case strong equilibria can be a factor of $\Omega(n)$ away from optimal, and, in general, computing a best response is an NP-hard problem. For less permissible sets of strategies, we show that pure equilibria might not exist, and deciding their existence as well as computing them if they exist constitute NP-hard problems.

2012 ACM Subject Classification Theory of computation → Algorithmic game theory and mechanism design

Keywords and phrases Nash Equilibrium, Financial Network, Systemic Risk, Price of Anarchy, Equilibrium Computation

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.46

Related Version A full version of the paper is available at `https://arxiv.org/abs/1908.01714`, see [6].

Acknowledgements We thank Pascal Lenzner and Steffen Schuldenzucker for valuable discussions and feedback on the results of this paper. NB thanks Dr. h. c. Maucher for funding his position.

1 Introduction

The last major financial crisis and its aftermath have highlighted the systemic risks and resulting hazards for society that arise in financial markets, which are characterized by different, highly interconnected financial institutions. Over the last decade an increased research effort has been underway to analyze, understand, and manage the systemic risks in financial markets. Main aspects of interest are contagion effects and cascading defaults, as well as recommendations for suitable regulation on a national and international level.

A prominent approach in the area of systemic risk stems from the seminal work by Eisenberg and Noe [13]. Here the set V of financial institutions (or *firms*) is the node set of a directed graph $G = (V, E)$. The directed and weighted edges $e \in E$ express the debt

relations among firms. In addition, each firm has non-negative *external assets*, which capture the value of property rights (such as real-estate, gold, business and mortgage loans, etc.) that the firm has acquired from non-financial institutions. Eisenberg and Noe discuss a clearing mechanism for such a market, in which every firm v uses its available assets to pay its debt. The clearing follows basic balance sheet identities and evolves rather mechanically. It is commonly assumed that payments are distributed among all creditors in a pro-rata fashion, i.e., for each interbank debt (i.e., each outgoing edge) firm v allocates a proportional share of its available assets. Similarly, a firm receives the payments for its incoming edges (so-called *internal assets*), which in turn are also paid proportionally to outgoing edges (until all debt is paid or all assets are distributed). In this way, clearing payments constitute a fixed point of the system. Eisenberg and Noe discuss existence and uniqueness issues of such fixed point payments.

While the model has been studied intensively over the last two decades, there are a plethora of open questions, especially towards understanding strategic and game-theoretic issues. In this paper, our focus are the strategic incentives and their consequences for clearing mechanisms in financial markets. For example, rather than using pro-rata payments, financial institutions often put priority on clearing certain debts first. In this way, a firm v could profit substantially when using a suitable priority-based payment scheme – by clearing certain debt first, the money propagates through the network and returns to v in the form of additional internal assets. In the Eisenberg-Noe model, it is easy to see that the intuitive pro-rata clearing mechanism is not always incentive compatible, and strategic incentives of this kind can arise frequently.

In this paper, we study the properties of priority-based clearing mechanisms for financial markets. We focus on payment schemes that constitute a pure Nash or even a strong equilibrium in the underlying strategic payment game for the firms. In this way, firms have no unilateral or coalitional incentives to deviate from the payment schemes proposed by the clearing mechanism. Depending on the granularity of priorities, the resulting games have different properties. In particular, if priorities are implemented over debt contracts, existence of a pure Nash or a strong equilibrium becomes strongly NP-hard to decide. Instead, if firms can assign each unit of money arbitrarily to any open debt, a strong equilibrium always exists and can be computed in strongly polynomial time. Moreover, in this case, there is even a strong equilibrium that maximizes the sum of all assets of all firms.

Our results imply interesting insights for bankruptcy settlement of insolvent firms. It turns out that only insolvent firms face a potential strategic decision about where to allocate money in order to maximize the internal assets through network effects (or, equivalently, minimize the remaining debt after clearing the network). In case there is a benevolent and centralized bankruptcy settlement, we show that it can implement a clearing mechanism with monotone payment strategies that leads to a socially optimal clearing state. It comes with the additional guarantee of giving no coalition of firms an incentive to pay their debts differently.

Instead, if clearing payments are determined by suitable negotiation in a decentralized fashion resulting in some arbitrary Nash or strong equilibrium, the total amount of internal assets available to the firms in the system can deteriorate drastically. Similar problems arise if a centralized clearing mechanism is restricted to payment schemes based on priorities over single loans (rather than units of money). This can lead to non-existence of pure equilibria in the resulting games. Even if equilibria exist, they can be undesirable since the total amount of internal assets of all firms can be drastically smaller than in an optimal solution. This shows a marked contrast between centralized and decentralized bankruptcy settlement and highlights how the structure of permissible payment strategies impacts the performance and the structural properties of the clearing mechanism.

1.1 Contribution

In this paper, we study the properties of priority-based clearing mechanisms for financial markets. We assume the network of liabilities is given, but insolvency resolution is driven by strategic considerations. In particular, we analyze payment schemes that constitute a pure Nash or even a strong equilibrium in the underlying strategic payment game for the firms.

Below, we introduce basic preliminaries of the Eisenberg-Noe model. In addition, we introduce two classes of priority-based payment strategies for the firms and analyze the resulting clearing mechanisms. For an *edge-ranking strategy*, a firm ranks its debt contracts and assigns its assets in order of the ranking. As a superset of strategies, we consider *coin-ranking strategies*, where money is considered in units ("coins"). Instead of contracts, each firm ranks single coin payments to the contracts. By letting the value of a coin approach zero, coin-ranking strategies become equivalent to monotone strategies, where the payments of a firm are simply a monotone function of its total available assets.

In Section 2 we present structural insights on the clearing states for a strategy profile in an edge- or coin-ranking game. In such a profile, each firm is choosing an edge- or coin-ranking strategy to pay its debt. For each such profile, we prove that the possible clearing states form a lattice with respect to the vector of assets of each firm (Theorem 3). In particular, there is a unique clearing state that pointwise maximizes the assets available to each firm (given this strategy profile). In the full version [6] we show that it can be computed in strongly polynomial time. We assume that this state defines the assets and, thus, the utility of each firm in the strategy profile. Similar properties were shown in [13] for profiles composed of pro-rata payment strategies.

In Section 3 we study coin-ranking games and strategic choice of payments. Our interest lies in the existence, computational complexity, and social quality of equilibria. We show that there always is a strategy profile that represents a strong equilibrium, in which no coalition of firms has an incentive to deviate. Furthermore, it maximizes social welfare, i.e., the sum of all assets or total revenue available to all firms. Such a strong equilibrium can be computed in strongly polynomial time (Theorem 6). It can be represented compactly, even though the strategy shall rank all (possibly pseudo-polynomially many) coins that a firm might have available. In contrast, it is strongly NP-hard to find a best-response strategy for a single firm in a given arbitrary strategy profile of a coin-ranking game (Theorem 9).

For worst-case equilibria and the strong price of anarchy, we show that the deterioration of social welfare in a strong equilibrium compared to a social optimum is tightly characterized by the min-max length of cycles in any social optimum (Theorem 11). This implies that networks with optimal money circulation composed of small cycles yield a small inefficiency in strong equilibria. In contrast, a worst-case Nash equilibrium, which is stable only against unilateral deviations, can be arbitrarily worse than a social optimum, even in simple games with a constant number of firms (Proposition 10).

In Section 4 we study equilibria in edge-ranking games, where all firms are restricted to play edge-ranking strategies. Restricting the strategy space to rankings over contracts can have devastating consequences for the existence and social quality of equilibria. In edge-ranking games, pure Nash and strong equilibria can be absent, and deciding their existence is strongly NP-hard (Theorem 14). The same hardness applies for computing a social optimum, and for computing a pure Nash or strong equilibrium when it is guaranteed to exist. Even the best strong equilibrium can be a factor of $\Omega(n)$ worse than the social optimum in terms of social welfare (Proposition 16). For pure Nash equilibria, even the best one can be arbitrarily worse than a social optimum (Proposition 17).

1.2 Related Work

Financial Networks. On a conceptual level, we study issues of strategic choice and computational complexity in financial networks. There have been works addressing computational complexity of diverse issues, such as pricing options with [3] and without information asymmetry [7], finding clearing payments with credit default swaps [27], or estimating the number of defaults when providing a shock in the financial system [20]. In addition, many extensions to the model by Eisenberg and Noe have been proposed in the literature on financial markets. However, even models including cross-holdings of equity [28], default costs [26], or debt contracts of different maturities [15] follow the idea of the basic approach that all contracts have to be cleared consistently, i.e., clearing payments locally adhere to the rather mechanical clearing rule and constitute a fixed point solution globally. Indeed, Barucca et al. [5] have recently shown that many of the above models can be unified in terms of self-consistent network valuations. A well-known result of such models is the "robust-yet-fragile" property exhibited by financial networks, i.e., contagion arises in an all-or-nothing fashion akin to the formation of a giant connected component in random graph models [17]. This provides important insights into systemic risk and advises the need for macro-prudential regulation.

Accordingly, the rather mechanical pro-rata payments are also usually presumed in models studying contagion effects arising from overlapping portfolios [8, 9]. In this case, distressed firms are selling assets which in turn decreases the value of these assets by market impact. Here, it is commonly assumed that firms mechanically sell all their assets in a pro-rata fashion. In turn, the resulting market impact is modeled as a known function parametrized by the market depth or liquidity of each asset. In contrast, especially decisions regarding the portfolio composition of financial firms yield substantial potential for strategic consideration in reality.

To our knowledge, strategic aspects are currently reflected mostly in models of network formation [14, 1]. A three period economy is assumed where firms can invest into risky assets. To do so, they strategically decide to borrow funds from outside investors as well as other firms. Thereby a network of financial cross-holdings is endogenously formed as each firm maximizes their expected profit. The results show that risk-seeking firms tend to over-connect leading to stronger contagion and systemic risk as compared to the socially optimal risk-sharing allocation. Note that in this case, strategic aspects only play a role in the formation of inter-bank relations whereas the clearing mechanism is assumed to follow the same process as in [13]. Another strategic variant of the model by Eisenberg and Noe has been considered in [2] in the form of a two-period model. Firms strategically store some amount from their first period endowment with an interest rate in order to increase the available assets in the second period. The clearing in both rounds is based on pro-rata payments. Each firm optimizes a function of the remaining debts in both periods.

Flow Games. On a more technical level, our game-theoretic approach is related to a number of existing game-theoretic models based on flows in networks. In cooperative game theory, there are several notions of flow games based on a directed flow network. Existing variants include games, where edges are players [12, 21, 22, 18, 11, 4], or each player owns a source-sink pair [25, 24]. The total value of a coalition C is the profit from a maximum (multi-commodity) flow that can be routed through the network if only the players in C are present. There is a rich set of results on structural characterizations and computability of solutions in the core, as well as other solution concepts for cooperative games. In contrast to our work, these games are non-strategic. Instead, here we consider each player as a single node with a strategic decision about flow allocation.

More recently, a class of strategic flow games has been proposed in [23, 19]. There is a capacitated flow network with a set of sources nodes. At each source node, a given amount of flow enters the network. Each node of the network is owned by a single player. Each player always owns a designated sink node, as well as one or more additional nodes from the network. A player can choose a flow strategy for each of her nodes. The flow strategy specifies, for every node v and every $x \geq 0$, how an incoming flow of x at v is distributed onto the outgoing edges (if any). Each flow strategy needs to fulfill flow conservation constraints at every node, subject to capacity on the outgoing edges. Each player aims to maximize the incoming flow at its sink node.

For these games there exist a number of Σ_2^p-completeness results for, e.g., determining the value of a game in a two-player Stackelberg variant, or determining the existence of a pure Nash equilibrium in a multi-player variant. In the latter game, computing a best response can also be NP-hard. Our approach is related to these games. However, motivated by financial networks we assume each firm is a single (source) node. The firm optimizes the incoming flow at its node (without it being a sink node). We study the computational complexity and social quality of equilibria. Moreover, strategic incentives arise mainly from cycles in the network – a condition absent in the existing work on max-flow games [23, 19] where the network is assumed to be acyclic.

The problem of calculating a clearing state for a given strategy profile in our games is closely related to the notion of a stable flows. In the stable flow problem, each node is equipped with an externally given preference order over both incoming and outgoing arcs. There always exists a stable flow, and the set of stable flows forms a lattice [16]. In fact, there is an augmenting path algorithm for computing a stable flow with polynomial running time [10].

1.3 Financial Networks with Payment Strategies

Network Model. We consider a financial network model due to Eisenberg and Noe [13]. There is a network $G = (V, E)$ with node set V of *institutions* or *firms*. Each firm $v \in V$ has *external assets* of value $a_v^x \geq 0$. Moreover, the firms are related via a set E of *liabilities*. Each liability $(u, v) \in E$ is a directed edge from firm $u \in V$ to firm $v \in V$. The *weight* $c(e) \geq 0$ of some edge $e = (u, v)$ is the amount of money that u owes to v. We follow standard notation in graph theory and denote by $E^+(v) = \{(v, u) \in E\}$ and by $E^-(v) = \{(u, v) \in E\}$ the set of outgoing and incoming edges of $v \in V$, respectively. The *total liabilities* $\ell(v)$ of firm v is the total amount of money firm v owes to other firms, i.e.,

$$\ell(v) = \sum_{e \in E^+(v)} c(e).$$

We strive to understand issues of computational complexity. As such, we will assume that all numbers in the input, i.e., all a_v^x and $c(e)$, are integer numbers.

We consider clearing mechanisms based on strategic payments decisions. A *money flow* g_e on edge e satisfies $0 \leq g_e \leq c(e)$. Given a money flow on each edge, the *internal assets* of firm v are the total incoming money from other firms, i.e.,

$$a_v^i = \sum_{e \in E^-(v)} g_e.$$

The *total assets* of v are the sum of external and internal assets $a_v = a_v^x + a_v^i$. A firm is *insolvent* if its total assets are strictly smaller than its total liabilities, i.e., $a_v < \ell(v)$.

Eisenberg and Noe define a clearing mechanism with money flows given by *pro-rata* payments. In their clearing mechanism, each firm v distributes its total assets a_v proportionally on its outgoing edges until all debt is paid. More formally, every edge $e \in E^+(v)$ is assigned a money flow of $g_e(a_v) = \min\left\{c(e), a_v \cdot \frac{c(e)}{\ell(v)}\right\}$. Firm v keeps the surplus $a_v - \ell(v) \geq 0$ for itself, if any.

Payment Strategies. In this paper, we analyze incentives when firms strategically manipulate their payments. As such, we study *money flow games* defined as follows. Each firm $v \in V$ chooses as a *strategy* a parametrized flow function $f_e(y)$ for every outgoing edge $e \in E^+(v)$ and every $y \geq 0$. The strategy $\mathbf{f}^v = (f_e(y))_{e \in E^+(v)}$ must satisfy for every $y \geq 0$

$$0 \leq f_e(y) \leq c(e) \qquad \text{(capacity constraint)} \tag{1}$$

$$\sum_{e \in E^+(v)} f_e(y) = \min\{y, \ell(v)\} \qquad \text{(no-fraud constraint)} \tag{2}$$

Intuitively, the strategy specifies, for every possible value $y \geq 0$ of total assets available to firm v, how v will allocate these assets to pay its debts. The capacity constraint ensures that no debt is overpaid, the no-fraud constraint requires that v does not embezzle assets as long as there is unpaid debt. This definition includes pro-rata payments as one possible strategy profile. Given a *strategy profile* $\mathbf{f} = (\mathbf{f}^v)_{v \in V}$, a *clearing state* $\mathbf{a} = (a_v)_{v \in V}$ is a vector of assets such that

$$a_v = a_v^x + \sum_{e=(u,v) \in E^-(v)} f_e(a_u) \qquad \text{(fixed point constraint)} \tag{3}$$

holds for all nodes $v \in V$. Equivalently, this ensures that for a given strategy profile $\mathbf{f}$, there is a clearing state $\mathbf{a}$ such that the flow $\mathbf{g}$ defined by $g_e = f_e(a_u)$ is in fact a money flow. The *utility* of firm v is a_v, i.e., v's goal is to choose a strategy to maximize its total assets in the clearing state.

▶ **Proposition 1.** *If all $f_e(y)$ are continuous, there exists at least one clearing state.*

The proof is a straightforward application of Brouwer's fixed point theorem and thus omitted. If strategies are not continuous, it is easy to construct examples where no clearing state exists. Still, even for a continuous strategy profile $\mathbf{f}$, there could be multiple clearing states $\mathbf{a}$ (even for pro-rata profiles). Given sufficiently complex strategy profiles with compact representation, computation of a clearing state might even become computationally difficult.

In the rest of the paper, we focus on a set of rich and meaningful strategy spaces, for which we can single out a unique clearing state with a simple algorithm. An intuitive and well-motivated class of strategies can be derived via *rankings* or *seniorities*.

Edge-Ranking Games. In an *edge-ranking game*, each player $v \in V$ spends its assets to pay its debts according to a strict and total order over $E^+(v)$, which we represent by a permutation $\pi_v = (e_1, e_2, \ldots)$. v first pays all debt of edge $e_1 = \pi_v(1)$, then $e_2 = \pi_v(2)$, etc. until all debt is paid or it runs out of assets. Formally, $f_{e_i}(y) = \min\{c(e_i), \max\{0, y - \sum_{j<i} c(e_j)\}\}$. The *edge-ranking strategy* of v is fully described by the ranking π_v, hence we denote a strategy profile in edge-ranking games by $\boldsymbol{\pi} = (\pi_v)_{v \in V}$.

Coin-Ranking Games. As a strict superset of such strategies, consider the case where each player $v \in V$ can spend its assets to pay its debts in a monotone fashion. In coin-ranking games, we rely on integrality of all values for c_e and a_v^x, and interpret money flow as being

discretized into "coins" of value 1. Thus, for a *coin-ranking strategy*, the parametrized flow functions $f_e(y)$ for every outgoing edge $e \in E^+(v)$ are defined on the non-negative integer numbers $f_e(y) : \mathbb{N}_0 \to \mathbb{N}_0$. They are characterized by capacity and no-fraud constraints, and, for every $y, y' \in \mathbb{N}_0$ with $y \geq y'$

$$f_e(y) \geq f_e(y') \qquad \text{(monotonicity constraint)} \tag{4}$$

Note that, by letting the value of a coin tend to 0, coin-ranking strategies become arbitrary *monotone strategies* $f_e(y) : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$.

Coin-ranking strategies generalize edge-ranking strategies. Maybe counterintuitively, every coin-ranking game is also a special edge-ranking game – replacing each edge e with weight $c(e)$ many multi-edges of unit weight expands a coin-ranking game into an equivalent edge-ranking game. There is a one-to-one correspondence between coin-ranking strategies in the original game and edge-ranking strategies in the expanded game. Intuitively, for a coin-ranking strategy in the original game, a player v pays the first coin of assets to the multi-edge $\pi_v(1)$, the second coin to $\pi_v(2)$, etc. in the expanded edge-ranking game until all debt is paid or v runs out of assets. The expansion of the game implies a pseudo-polynomial blowup in representation size, but nevertheless, the structural equivalence is very useful for characterizing and analyzing solutions to coin-ranking games.

Note that the representation of a coin-ranking strategy might require pseudo-polynomial size even in the original non-expanded game. We discuss this issue below in Section 3.1. It turns out that in every coin-ranking game, we can restrict attention to a subset of compactly representable coin-ranking strategies.

Clearing States and Utilities. For a given strategy profile $\boldsymbol{\pi}$ in an edge- or coin-ranking game, we determine the utility using a clearing state $\hat{\mathbf{a}}$, where we choose the one that maximizes the total revenue in the network, i.e., the sum of total assets available to all firms

$$\textsc{Rev}(\boldsymbol{\pi}, \mathbf{a}) = \sum_{v \in V} a_v = \sum_{v \in V} a_v^x + \sum_{e=(u,v) \in E} f_e(a_u) = \sum_{v \in V} a_v^x + \sum_{e \in E^+(v)} f_e(a_v).$$

Note that the choice of $\hat{\mathbf{a}}$ and thus, the resulting revenue significantly depends on the strategy profile $\boldsymbol{\pi}$. In many cases, for a fixed strategy profile $\boldsymbol{\pi}$, the clearing state $\hat{\mathbf{a}}$ is unique, and there is no choice based on maximum revenue. Moreover, in case there are several clearing states for a strategy profile $\boldsymbol{\pi}$ in an edge-ranking game, it turns out that they can be arranged into a lattice with coordinate-wise maximum inducing a partial order. We choose the coordinate-wise maximal clearing state, since it maximizes $\textsc{Rev}(\boldsymbol{\pi}, \mathbf{a})$ as a natural measure of social welfare. In this sense, the properties of edge-ranking games mirror the conditions shown for pro-rata payments in [13]. We prove these conditions in the subsequent section.

2 Clearing States

2.1 Circulation Structure in Money Flow Games

We observe a useful *circulation representation* of clearing states in money flow games. Given a strategy profile $\mathbf{f}$, the fixed point and no-fraud constraints imply for any clearing state $\mathbf{a}$ the conservation of money flow. Now using an auxiliary source s, we can represent all money flows in $\mathbf{a}$ in the form of a circulation. We build a *circulation network* G' by adding node s, for every $v \in V$ we add an auxiliary edge (v, s) with capacity $c((v, s)) = \infty$, and for every

$v \in V$ with $a_v^x > 0$ we add an auxiliary edge (s, v) with $c((s, v)) = a_v^x$. In this way, external assets of v become internal assets via a flow on edge (s, v). Surplus assets of w become a flow on edge (w, s), i.e., the internal assets of s.

▶ **Proposition 2.** *For every clearing state* $\mathbf{a}$ *of a strategy profile* $\mathbf{f}$ *in a money flow game, the flow in* G' *can be decomposed and represented as a circulation. The auxiliary source* s *has assets of* $a_s = \sum_{v \in V} a_v^x$*, and all auxiliary edges* (s, v) *are saturated.*

Proof. This is a simple consequence of fixed point and no-fraud constraints. Surplus at firm $v \in E$ exists only if v pays all debt

$$\sum_{e \in E^+(v)} f_e(a_v) = \min(a_v, \ell(v)) \ .$$

Moreover, the total external assets constitute the total net revenue:

$$\begin{aligned} \sum_{v \in V} a_v^x &= \sum_{v \in V} a_v^x + \sum_{v \in V} \sum_{e=(u,v) \in E^-(v)} f_e(a_u) - \sum_{v \in V} \sum_{e=(u,v) \in E^-(v)} f_e(a_u) \\ &= \sum_{v \in V} a_v - \sum_{e \in E^+(v)} f_e(a_v) \quad = \quad \sum_{v \in V} \max\{0, a_v - \ell(v)\} \end{aligned}$$

The net revenue of every firm gets routed to the auxiliary source s and constitutes the assets a_s. Thus $a_s = \sum_{v \in V} \max\{0, a_v - \ell(v)\} = \sum_{v \in V} a_v^x$, and all auxiliary edges (s, v) are saturated. Overall, by routing the surplus assets to s, we obtain exact flow conservation at every node. As such, the flow can be decomposed and represented as a circulation. ◀

2.2 Structure for Monotone Strategies

Lattice Structure. Consider an arbitrary money-flow game and a strategy profile $\mathbf{f}$ of monotone strategies. Let $\mathcal{A}$ be the set of feasible clearing states for a strategy profile $\mathbf{f}$. We show that $(\mathcal{A}, \geq)$ forms a lattice with the coordinate-wise comparison. Formally, $\mathbf{a} \geq \mathbf{a}'$ iff $a_v \geq a'_v$ for all v; and $\mathbf{a} > \mathbf{a}'$ iff $a_v \geq a'_v$ for all v and $a_v > a'_v$ for at least one v.

▶ **Theorem 3.** *For every strategy profile* $\mathbf{f}$ *in a money-flow game with monotone strategies, the pair* $(\mathcal{A}, \geq)$ *forms a lattice.*

Proof. Consider $A = \left\{ \mathbf{a} \;\middle|\; 0 \leq a_v \leq a_v^x + \sum_{e \in E^-(v)} c(e), \quad \forall v \in V \right\}$, a superset of all possible asset vectors. Obviously, $(A, \geq)$ forms a lattice with the coordinate-wise comparison defined above. For a given strategy profile $\mathbf{f}$, the map $g : A \to A$ with $g(\mathbf{a})_v = a_v^x + \sum_{e=(u,v) \in E^-(v)} f_e(a_u)$ is a monotone function for every firm $v \in V$, since for every edge $e = (u, v) \in E$ the strategy of firm u implies that f_e is monotone in a_u. Obviously, the set of clearing states $\mathcal{A}$ is the set of fixed-point asset vectors of g. The result follows by applying the Knaster-Tarski theorem. ◀

Solvent Firms. The previous result implies uniqueness of $\hat{\mathbf{a}}$ for a given monotone strategy profile $\mathbf{f}$. We observe another interesting property, which shows that in monotone money flow games the maximal clearing state $\hat{\mathbf{a}}$ is unique as long as all insolvent firms stick to their strategy. Since every strategy satisfies capacity and no-fraud constraints, the payments of solvent firms remain the same if they receive the same assets, and vice versa. Consequently, strategies of solvent firms have no impact on the asset vector $\hat{\mathbf{a}}$. For any solvent firm v, every strategy constitutes a best response.

▶ **Proposition 4.** *For a given money flow game, consider any monotone strategy profile* $\mathbf{f}$*, the corresponding clearing state* $\hat{\mathbf{a}}$*, and any solvent firm* v *with* $\hat{a}_v \geq \ell(v)$*. Every strategy* $\mathbf{f}'^v$ *is a best response for* v *against the other strategies* $\mathbf{f}_{-v}$ *and results in the same clearing state* $\hat{\mathbf{a}}$*.*

Proof. Consider a deviation $\mathbf{f}'^v$, the resulting state $\mathbf{f}' = (\mathbf{f}'^v, \mathbf{f}^{-v})$ and the resulting revenue-maximizing clearing state $\mathbf{a}'$. Suppose that $\hat{\mathbf{a}} \neq \mathbf{a}'$. Firm v is solvent under $\hat{\mathbf{a}}$, thus $\sum_{e \in E^+(v)} f_e(\hat{a}_v) = l(v)$ and $f_e(\hat{a}_v) = c(e)$. Using Theorem 3 we can assume w.l.o.g. that $\hat{\mathbf{a}} > \mathbf{a}'$ (i.e., $\hat{a}_u \geq a'_u$ for all firms u, and $\hat{a}_w > a'_w$ for at least one firm w). We construct an equivalent game, in which we remove all edges in $E^+(v)$ and instead increase external assets to $\bar{a}^x(u) = a^x_u + c(e)$ for all u with $(v,u) \in E^+(v)$. Observe that $\hat{\mathbf{a}}$ is still a feasible clearing state in the constructed game. However, any clearing state $\mathbf{a}$ in the original game with $a_v \geq l(v)$ induces $f(a_v) = c(e)$ independent of the chosen strategies of v due to the non-fraud condition. Thus, any clearing state $\mathbf{a}$ with $a_v \geq l(v)$ in the new game is still a feasible clearing state in the old game. We conclude that $\hat{\mathbf{a}}$ is a feasible clearing state under $\mathbf{f}'$. This is a contradiction to the maximality of $\mathbf{a}'$. ◀

3 Coin-Ranking Games

3.1 Representation

An instance of a money flow game is given by the network G and integer numbers for edge weights $c(e)$ and external assets a^x_u. Hence, the representation of the instance is logarithmic in input numbers for $c(e)$ and a^x_u. In contrast, if we consider arbitrary coin-ranking strategies $\mathbf{f}^v$ for firm v, this specifies a ranking over all *coins* of value 1. This is linear in $\sum_{e \in E^+(v)} c(e)$ and, thus, only pseudo-polynomial in the instance representation. Our first observation is that in every coin-ranking game, we can restrict attention to threshold-ranking strategies with a polynomial representation. A *threshold-ranking* strategy $\pi^t_v = (\pi_v, \tau_v)$ is composed of a permutation π_v over $E^+(v)$ and a vector of thresholds $\tau_v = (\tau_e)_{e \in E^+(v)}$ with $0 \leq \tau_e \leq c(e)$ for every $e \in E^+(v)$.

The interpretation is as follows. In π^t_v, firm v first pays τ_e to every edge $e \in E^+(v)$, sequentially in the order given by π_v. Then, it pays the remaining $c(e) - \tau_e$ to every edge in the order given by π_v. That is, v first considers edge $\pi_v(1)$ and pays the first $\tau_{\pi_v(1)}$ coins to this edge. The next $\tau_{\pi_v(2)}$ coins are paid to edge $\pi_v(2)$ etc. until $\sum_{j=1}^{|E^+(v)|} \tau_{\pi_v(j)}$ coins are paid to the edges (or v runs out of assets). Then, the remaining $c(\pi_v(1)) - \tau_{\pi_v(1)}$ coins are paid to edge $\pi_v(1)$, then the next $c(\pi_v(2)) - \tau_{\pi_v(2)}$ coins to $\pi_v(2)$ etc.

Indeed, we can restrict attention to threshold-ranking strategies in coin ranking games. The formal proof is deferred to the full version [6].

▶ **Proposition 5.** *For every strategy profile* $\mathbf{f}$ *in a coin-ranking game with clearing state* $\hat{\mathbf{a}}$ *and every firm* v*, there is a threshold-ranking strategy* π^t_v *such that the profile* $(\pi^t_v, \boldsymbol{\pi}_{-v})$ *has the same clearing state* $\hat{\mathbf{a}}$ *and, thus, the same utilities for all firms.*

3.2 Existence and Computation of Equilibria

Our first result is that in every coin-ranking game there is a strong equilibrium that maximizes the total revenue of all firms. This strong equilibrium can be computed in polynomial time. In particular, consider the instance (G, c, a^x) as a money flow game and an arbitrary clearing state, i.e., a circulation of maximum value in the circulation network G'. This circulation is also a clearing state of a strong equilibrium in threshold-ranking strategies.

▶ **Theorem 6.** *For every coin-ranking game, there is a strong equilibrium with money flows that maximize the total revenue in the network. The strong equilibrium can be computed in polynomial time.*

Proof. Consider the circulation network $G' = (V, E')$. An optimal circulation $\mathbf{f}^*$ that maximizes the total flow value saturates all outgoing auxiliary edges from s. Hence, it maximizes the total assets of all firms

$$\sum_{e \in E'} f^*_e = 2 \sum_{v \in V} a^x_v + \sum_{e \in E} f^*_e = \sum_{v \in V} a^x_v + \textsc{Rev}(\mathbf{f}^*) \ .$$

$\mathbf{f}^*$ can be computed in strongly polynomial time [29]. Since all edge weights are integral, we can assume all f^*_e are integral. We can turn this circulation into a clearing state for a strategy profile with threshold-ranking strategies. Every firm v chooses an arbitrary order π_v over $E^+(v)$ and sets thresholds $\tau_e = f^*_e$. Clearly, in this strategy profile the optimal circulation corresponds to the maximum-revenue clearing state $\hat{\mathbf{a}}$.

Let us prove that this strategy profile $\boldsymbol{\pi}$ is a strong equilibrium. A coalition $C \subseteq V$ of firms has a profitable deviation $\boldsymbol{\pi}'_C = (\pi'_v)_{v \in C}$ if upon joint deviation of C to π'_C, the resulting assets $\mathbf{a}'$ in the new profile $(\boldsymbol{\pi}'_C, \boldsymbol{\pi}_{-C})$ are strictly better, i.e., $a'_v > \hat{a}_v$ for every $v \in C$. We will show that no coalition $C \subseteq V$ has a profitable deviation.

Suppose for contradiction that there is a coalition C with a profitable deviation. Examine the new profile $(\boldsymbol{\pi}'_C, \boldsymbol{\pi}_{-C})$ and consider a firm $v \in C$. Since $a'_v > \hat{a}_v$, there must be an edge $(u, v) \in E^-(v)$ that has strictly more incoming flow in the new profile $f'_e(a'_u) > f_e(a_u)$. Now consider node u. If $u \in C$, then $a'_u > a_u$, so there is again some edge $E^-(u)$ that has strictly more incoming flow in the new profile. Otherwise, if $u \notin C$, then u still plays the threshold-ranking strategy obtained from $\mathbf{f}^*$. Since this is a monotone strategy, a higher flow on (u, v) can only occur if u has larger assets. Thus, $a'_u > a_u$, so there is again some edge $E^-(u)$ that has more incoming flow in the new profile.

We can repeat this argument indefinitely. As such, there must be a cycle of edges that all have more flow under $(\boldsymbol{\pi}'_C, \boldsymbol{\pi}_{-C})$ than under $\boldsymbol{\pi}$. Such a cycle, however, can be used to increase the flow circulation. This contradicts that $\hat{\mathbf{a}}$ represents an optimal circulation. ◀

▶ Remark 7. For the profitable deviation, we can even allow arbitrary continuous strategies and any choice of clearing state for the deviation profile. As such, the strategy profile obtained from $\mathbf{f}^*$ is a strong equilibrium even in general money flow games (with suitable choice of clearing state).

▶ Remark 8. If one allows deviations that weakly improve the coalition (i.e., $a'_u \geq \hat{a}_u$ for all $u \in C$ and $a'_v > \hat{a}_v$ for at least one $v \in C$), it is a simple exercise to see that there are coin-ranking games, in which no stable state (termed super-strong equilibrium) exists.

While it is computationally easy to compute a socially optimal strong equilibrium, computing a best-response strategy for a general strategy profile can be strongly NP-hard, since best responses can provide answers to computationally hard decision problems. For the following result, we assume the coin-ranking game is given in the edge-ranking representation as a network with unit-weight multi-edges. Note that the edge weights in our instances of interest can be restricted to the set $\{0, 1\}$. Thus, our construction needs no multi-edges, and the representation incurs no overhead. A proof of the following theorem is given in the full version [6].

▶ **Theorem 9.** *For a given strategy profile* $\mathbf{f}$ *of a coin-ranking game, deciding whether a given firm* v *has a best response resulting in assets at least* k *is strongly NP-complete. This holds even in coin-ranking games without external assets and all edge weights in* $\{0, 1\}$.

3.3 Total Revenue of Equilibria

In this section, we analyze the total revenue in pure Nash and strong equilibria. We relate this value to the social optimum, i.e., the total sum of assets for all firms in the best strategy profile. Clearly, since we proved existence of a system optimal strong equilibrium, the price of stability for Nash and strong equilibria are both 1. We bound the prices of anarchy for Nash and strong equilibria. The proof of the next Proposition is given in the full version [6].

▶ **Proposition 10.** *The price of anarchy for Nash equilibria is unbounded, even in coin-ranking games without external assets.*

The total revenue depends crucially on the emergence of cycles in the strategy profile. This requires an effort that is inherently coalitional, as such it might be unsurprising that in general Nash equilibria fail to provide good revenue guarantees.

To analyze the quality of strong equilibria, we again consider the coin-ranking game in the form of unit-weight multi-edges. Consider an optimal circulation $\mathbf{f}^*$ of maximum total revenue in the circulation network G'. Since we have unit-weight edges, we can assume that the optimal circulation has binary flows on each edge. Let $\mathcal{C}(\mathbf{f}^*) = \{C_1, \dots, C_k\}$ be a decomposition of $\mathbf{f}^*$ into cycles of unit flow. We denote by

$$d = \min_{\mathbf{f}^*, \mathcal{C}(\mathbf{f}^*)} \max_{C \in \mathcal{C}(\mathbf{f}^*)} |C_i|$$

the min-max size of any cycle, in any decomposition $\mathcal{C}(\mathbf{f}^*)$ of any optimal circulation $\mathbf{f}^*$.

▶ **Theorem 11.** *In coin-ranking games, the strong price of anarchy is at most d.*

Proof. Consider an optimal circulation $\mathbf{f}^*$ and a decomposition $\mathcal{C}(\mathbf{f}^*)$ such that all flow cycles $C_i \in \mathcal{C}(\mathbf{f}^*)$ have size at most $|C_i| \leq d$. The total revenue in $\mathbf{f}^*$ is given by

$$\textsc{Rev}(\mathbf{f}^*) = \sum_{C_i \in \mathcal{C}(\mathbf{f}^*)} |C_i| - \sum_{v \in V} a_v^x$$

since the circulation also accounts for the assets of the auxiliary source s.

Now consider a strong equilibrium $\boldsymbol{\pi}$ in the coin-ranking game with clearing state $\hat{\mathbf{a}}$. It yields a binary money flow in the network. Suppose there is a cycle $C_i \in \mathcal{C}(\mathbf{f}^*)$ such that $f_e(\hat{a}_u) = 0$ for all $e = (u, v) \in C_i$. Then the firms in this cycle have an incentive to jointly deviate and place the edges of C_i on first position in their ranking. Then the clearing state $\hat{\mathbf{a}}$ will emerge as before, adding a flow of 1 along the cycle C_i. This is a profitable deviation for the firms of C_i.

Consequently, for every cycle $C_i \in \mathcal{C}(\mathbf{f}^*)$ there must be at least one edge $e = (u, v) \in C_i$ such that $f_e(a_u) = 1$. Thus, the revenue in the strong equilibrium $\boldsymbol{\pi}$ is

$$\textsc{Rev}(\boldsymbol{\pi}, \hat{\mathbf{a}}) \geq \sum_{C_i \in \mathcal{C}(\mathbf{f}^*)} 1$$

and the ratio is at most d. ◀

▶ **Proposition 12.** *For every $d \geq 2$, there is a coin-ranking game with strong price of anarchy of $d - 1$.*

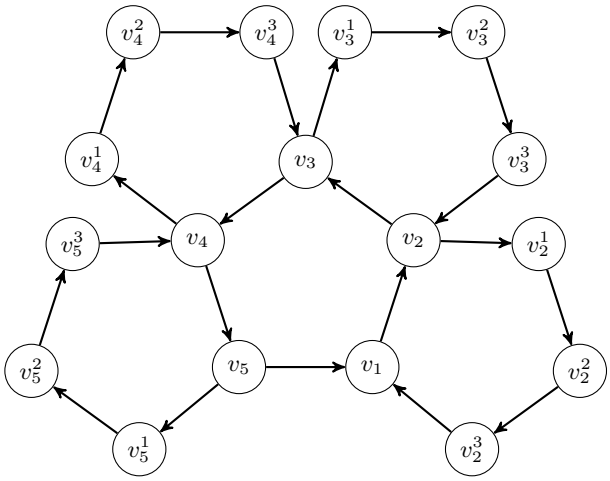


Figure 1 A coin-ranking game with $d = 5$ and a strong price of anarchy of $d - 1 = 4$.

Proof. The game is given by a graph G with $d + (d-1)(d-2)$ firms. G is constructed as follows. The firms $v_1, \dots, v_d$ are called *central firms* and they form a cycle of length d. For each $i = 1, \dots, d-1$, there are firms $(v_{i,j})_{j=1,\dots,d-2}$ that form additional cycles of length d with the edge (v_i, v_{i+1}). Thus, the set of edges is given by

$$E = \{(v_i, v_{i+1}) \mid i \in \{1, \dots, d-1\}\} \cup \{(v_d, v_1)\}$$
$$\cup \bigcup_{i=2,\dots,d} \left((v_i, v_i^1) \cup \{(v_i^j, v_i^{j+1}) \mid j = 1, \dots, d-3\} \cup (v_i^{d+2}, v_{i-1})\right) .$$

All edges have unit weight. An example of the instance with $d = 5$ is depicted in Figure 1. Observe that only firms v_i, $i = 2, \dots, d$ have multiple outgoing edges. We claim that $\pi_i = \big((v_i, v_{i+1}), (v_i, v_i^1)\big)$ and $\pi_d = \big((v_d, v_1), (v_d, v_d^1)\big)$ is a strong equilibrium. In order to see this, let $\hat{\mathbf{a}}$ be the clearing state corresponding to $\boldsymbol{\pi}$. The clearing state is given by

$$\hat{a}_v = \begin{cases} 1 & \text{if } v = v_1, v_2, \dots, v_d \ , \\ 0 & \text{otherwise} \ , \end{cases}$$

that is, $\textsc{Rev}(\boldsymbol{\pi}, \hat{\mathbf{a}}) = d$. Now, assume there is a non-empty coalition of player $S \subseteq (v_2, \dots, v_d)$ that all strictly increase their assets by a joint deviation. Note that v_d only has a single incoming edge, so $v_d \notin S$. Thus, the cycle $v_d, v_d^1, \dots, v_d^{d-2}, v_{d-1}$ cannot carry any flow. We conclude that v_{d-1} has only a single edge that can carry flow. Iterating this argument yields $S = \emptyset$, a contradiction.

However, the optimal flow emerges with the strategy profile $\pi_i = \big((v_i, v_i^1), (v_i, v_{i+1})\big)$ and $\pi_d = \big((v_d, v_d^1), (v_d, v_1)\big)$. It is easy to observe that this yields a total revenue of $(d-1)d$. Thus, the strong price of anarchy in this instance is $d - 1$. ◀

4 Edge-Ranking Games

4.1 Existence and Computation of Equilibria

With coin-ranking strategies we assume that firms have flexibility in allocation of single units of assets. In this section, we focus on a more restricted class of strategies, in which firms simply rank their outgoing edges and allocate assets in order of this ranking until they run

Table 1 Utility matrix for firms v_2 and v_3 in the proof of Proposition 13.

π_{v_2} \ π_{v_3}	$((v_3,v_1),(v_3,v_8))$	$((v_3,v_8),(v_3,v_1))$
$((v_2,v_1),(v_2,v_5))$	4 4	3 4
$((v_2,v_5),(v_2,v_1))$	2 5	3 3

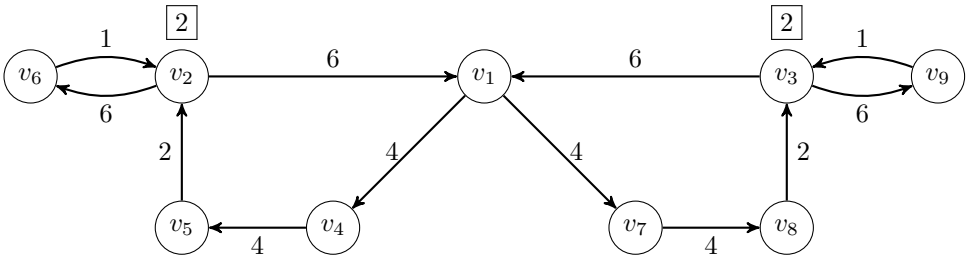


Figure 2 An edge-ranking game without a pure Nash equilibrium.

out of assets of all debts are paid for. In contrast to coin-ranking games, the restriction to rankings over edges (with different weights) can destroy the existence of (optimal) stable states. In fact, there are even games without a single pure Nash equilibrium.

▶ **Proposition 13.** *There is an edge-ranking game without a pure Nash equilibrium.*

Proof. Consider the game in Figure 2. The capacities of the edges are depicted next to the edges. Firms v_2 and v_3 each have external assets of 2, the other firms have 0 external assets. Firms v_1, v_2 and v_3 are the only ones with multiple outgoing edges. The strategy choices of the other firms are fixed. Due to the symmetry of the network, we can assume w.l.o.g. $\pi_{v_1} = ((v_1,v_4),(v_1,v_7))$. There are two possible strategy choices for each of the nodes v_2 and v_3. Checking all four resulting strategy profiles yields the utility matrix shown in Table 1 for firms v_2 and v_3. Inspecting the utilities, we see that there is no pure Nash equilibrium. ◀

The following theorem shows that a number of natural decision and optimization problems in edge-ranking games are indeed computationally intractable. Note that for coin-ranking games, these problems are either trivial (a strong equilibrium always exists) or can be solved in polynomial time (a strong equilibrium, which also represents a profile with maximum total revenue, can be computed in strongly polynomial time). The proof of the following theorem is deferred to the full version [6].

▶ **Theorem 14.** *Given an edge-ranking game the following problems are strongly NP-hard:*

1. *Deciding whether a pure Nash equilibrium exists or not.*
2. *Deciding whether a strong equilibrium exists or not.*
3. *Computing a pure Nash equilibrium, when it is guaranteed to exist.*
4. *Computing a strong equilibrium, when it is guaranteed to exist.*
5. *Computing a strategy profile with maximum total revenue.*

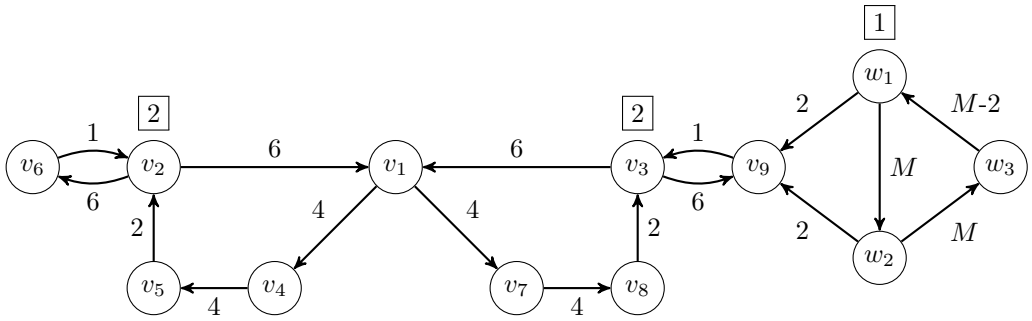


Figure 3 Edge-Ranking Game with unbounded Price of Stability.

▶ Remark 15. It is unclear whether the problem of deciding existence of a pure Nash equilibrium in an edge-ranking game is in NP or not, due to NP-hardness of verification that a firm plays a best response (see Theorem 9). It is easy to see that the decision problem is in Σ_2^p. The problem could be Σ_2^p-complete, similar to related decision problems in strategic max-flow games [23, 19]. Proving such a result is an interesting open problem.

4.2 Total Revenue of Equilibria

For edge-ranking games, the lower bound on the price of anarchy observed for coin-ranking games does apply, i.e., the price of anarchy can be unbounded. The restriction to edge-ranking strategies can have a drastic effect even on the quality of the best equilibrium in case it exists. In particular, in edge-ranking games the strong price of stability can be as high as $\Omega(n)$, and the price of stability might even be unbounded.

▶ **Proposition 16.** *For every $\varepsilon > 0$, there is an edge-ranking game with strong price of stability of at least $n/2 - \varepsilon$.*

Proof. We consider a slight modification of the instance in Proposition 12. In contrast to the instance described in the proof of Proposition 12, the edges $(v_1, v_n), (v_n, v_1), (v_1, v_2)$ have a weight $M + 1$ and all other edges a weight of M. The only node with more than a single outgoing edge is still v_1. If $\pi_{v_1} = ((v_1, v_n), (v_1, v_2))$, player v_1 gets assets of $M + 1$, which is optimal. The total revenue for $\boldsymbol{\pi}$ is $2M + 2$. $\boldsymbol{\pi}$ is the only Nash equilibrium and the only strong equilibrium.

In contrast, for profile $\boldsymbol{\pi}'$ with $\pi'_{v_1} = ((v_1, v_2), (v_1, v_n))$, firm v_1 only gets a revenue of M, but the total revenue is nM. Thus, the strong price of stability is $nM/(2M + 2) = n/2 - n/(2M + 2)$, which is at least $n/2 - \varepsilon$ for $M \geq n/(2\varepsilon) - 1$. ◀

▶ **Proposition 17.** *There is an edge-ranking game with unbounded price of stability.*

Proof. Consider the game in Figure 3, which uses the game without pure equilibrium from Figure 2. We add three firms. w_1 has external assets equal to 1, w_2 and w_3 no external assets. These firms have a cycle C of edges (w_1, w_2) and (w_2, w_3) with weight $M \gg 2$, as well as edge (w_3, w_1) with weight $M - 2$. In addition, there are edges (w_1, v_9) and (w_2, v_9) of weight 2.

In an optimal circulation, w_1 and w_2 prioritize the edges of C, leading to total revenue of $\Theta(M)$. In contrast, a pure Nash equilibrium can only exist if the w-firms ensure that the external assets of w_1 are routed to v_9, in which case a Nash equilibrium can exist (as observed

in the proof of Theorem 14). Clearly, both w_1 and w_2 have an incentive to deviate towards C. Hence, if *either* w_1 *or* w_2 places the edge to v_9 in first rank and the other does not, a unilateral deviation suffices to close C – thereby leaving the v-nodes with instability. However, if *both* w_1 *and* w_2 play strategies $\pi_{w_1} = ((w_1, v_9), (w_1, w_2))$ and $\pi_{w_2} = ((w_2, v_9), (w_2, w_3))$, no unilateral deviation can lead to flow along C. In this case, a pure Nash equilibrium evolves. Obviously the total revenue in this equilibrium is at most a constant. Hence, the price of stability is as large as $\Omega(M)$. ◀

5 Conclusions

In this paper, we have studied clearing mechanisms for financial networks and analyzed their properties from a computational and game-theoretic perspective. Our main results show that in these games, solvent firms have no strategic incentives, i.e., the game is played exclusively among insolvent firms. If firms are using coin-ranking strategies, every social optimum that maximizes the sum of all assets in the network constitutes a strong equilibrium. Moreover, it can be computed in strongly polynomial time. This result implies that a centralized bankruptcy settlement can achieve a clearing state, in which the social welfare is maximized and no coalition of firms gets incentivized to deviate. In contrast, when considering decentralized clearing and arbitrary strong equilibria, the social welfare depends on the length of cycles in the money circulation of a social optimum. For pure Nash equilibria, the deterioration in social welfare can be severe due to the lack of coordination among firms. Alternatively, when restricting the strategy spaces to edge-ranking strategies, we show that pure Nash and strong equilibria can be absent, hard to compute, and highly undesirable in terms of social welfare.

There are many open problems that arise from our work. For example, real-life markets involve a number of complex financial products (such as derivatives, credit-default swaps, etc.). Their impact on stability and computational complexity of financial markets is only beginning to attract attention in the literature. In this context, there are a variety of important game-theoretic aspects with respect to pricing, information revelation, or network creation, which are crucial for understanding financial markets and represent interesting avenues for future work.

References

1. Daron Acemoglu, Asuman Ozdaglar, and Alireza Tahbaz-Salehi. Systemic Risk in Endogenous Financial Networks. Available at SSRN: `https://ssrn.com/abstract=2553900`, January 22 2015. Columbia Business School Research Paper No. 15-17.
2. Nizar Allouch and Maya Jalloul. Strategic Default in Financial Network. Technical report, School of Economics, University of Kent, 2017. Studies in Economics 1721.
3. Sanjeev Arora, Boaz Barak, Markus Brunnermeier, and Rong Ge. Computational complexity and information asymmetry in financial products. *Comm. ACM*, 54(5):101–107, 2011.
4. Yoram Bachrach and Jeffrey Rosenschein. Power in threshold network flow games. *Auton. Agents and Multi-Agent Syst.*, 18(1):106–132, 2009.
5. Paolo Barucca, Marco Bardoscia, Fabio Caccioli, Marco D'Errico, Gabriele Visentin, Stefano Battiston, and Guido Caldarelli. Network Valuation in Financial Systems, 2016. `arXiv:1606.05164`.
6. Nils Bertschinger, Martin Hoefer, and Daniel Schmand. Strategic Payments in Financial Networks, 2019. `arXiv:1908.01714`.
7. Mark Braverman and Kanika Pasricha. The computational hardness of pricing compound options. In *Proc. 5th Symp. Innov. Theoret. Comput. Sci. (ITCS)*, pages 103–104, 2014.

8 Rodrigo Cifuentes, Gianluigi Ferrucci, and Hyun Song Shin. Liquidity risk and contagion. Bank of England, Working Paper No. 264, 2005.

9 Rama Cont and Eric Schaanning. Indirect Contagion and Systemic Stress Testing. Available at SSRN: `https://ssrn.com/abstract=2541114`, June 13 2017.

10 Ágnes Cseh and Jannik Matuschke. New and Simple Algorithms for Stable Flow Problems. *Algorithmica*, 81(6):2557–2591, 2019.

11 Xiaotie Deng, Toshihide Ibaraki, and Hiroshi Nagamochi. Algorithmic Aspects of the Core of Combinatorial Optimization Games. *Math. Oper. Res.*, 24(3):751–766, 1999.

12 Pradeep Dubey and Lloyd Shapley. Totally balanced games arising from controlled programming problems. *Math. Prog.*, 29(3):245–267, 1984.

13 Larry Eisenberg and Thomas Noe. Systemic Risk in Financial Systems. *Manag. Sci.*, 47(2):236–249, 2001.

14 Maryam Farboodi. Intermediation and Voluntary Exposure to Counterparty Risk. Available at SSRN: `https://ssrn.com/abstract=2535900`, August 1 2014.

15 Tom Fischer. No-Arbitrage Pricing under Systemic Risk: Accounting for Cross-Ownership. *Math. Finance*, 24(1):97–124, 2014.

16 Tamás Fleiner. On Stable Matchings and Flows. *Algorithms*, 7(1):1–14, 2014.

17 Prasanna Gai and Sujit Kapadia. Contagion in financial networks. *Proc. Royal Soc. London A: Math. Phys. Eng. Sci.*, 466(2120):2401–2423, 2010.

18 Daniel Granot and Frieda Granot. On Some Network Flow Games. *Math. Oper. Res.*, 17(4):792–841, 1992.

19 Shibashis Guha, Orna Kupferman, and Gal Vardi. Multi-Player Flow Games. In *Proc. 17th Conf. Auton. Agents and Multi-Agent Syst. (AAMAS)*, pages 104–112, 2018.

20 Brett Hemenway and Sanjeev Khanna. Sensitivity and computational complexity in financial networks. *Algorithmic Finance*, 5(3-4):95–110, 2016.

21 Ehud Kalai and Eitan Zemel. Generalized Network Problems Yielding Totally Balanced Games. *Oper. Res.*, 30(5):998–1008, 1982.

22 Ehud Kalai and Eitan Zemel. Totally Balanced Games and Games of Flow. *Math. Oper. Res.*, 7(3):476–478, 1982.

23 Orna Kupferman, Gal Vardi, and Moshe Vardi. Flow Games. In *Proc. 37th Conf. Found. Software Tech. Theor. Comput. Sci. (FSTTCS)*, pages 38:1–38:16, 2017.

24 Evangelos Markakis and Amin Saberi. On the core of the multicommodity flow game. *Decis. Support Syst.*, 39(1):3–10, 2005.

25 Christos Papadimitriou. Algorithms, Games and the Internet. In *Proc. 33rd Symp. Theory Comput. (STOC)*, pages 749–753, 2001.

26 L. C. G. Rogers and L. A. M. Veraart. Failure and Rescue in an Interbank Network. *Manag. Sci.*, 59(4):882–898, 2013.

27 Steffen Schuldenzucker, Sven Seuken, and Stefano Battiston. Finding Clearing Payments in Financial Networks with Credit Default Swaps is PPAD-complete. In *Proc. 8th Symp. Innov. Theoret. Comput. Sci. (ITCS)*, pages 32:1–32:20, 2017.

28 Teruyoshi Suzuki. Valuing Corporate Debt: The Effect of Cross-Holdings of Stock and Debt. *J. Oper. Res. Soc. Japan*, 2, 2002.

29 Éva Tardos. A strongly polynomial minimum cost circulation algorithm. *Combinatorica*, 5(3):247–256, 1985.

Fault Tolerant Subgraphs with Applications in Kernelization

William Lochet
University of Bergen, Norway
william.lochet@uib.no

Daniel Lokshtanov
University of California, Santa Barbara, USA
daniello@ucsb.edu

Pranabendu Misra
Max Planck Institute for Informatics, Saarbrücken, Germany
pmisra@mpi-inf.mpg.de

Saket Saurabh
Institute of Mathematical Sciences, HBNI and IRL 2000 ReLaX, Chennai, India
University of Bergen, Bergen, Norway
saket@imsc.res.in

Roohani Sharma
Institute of Mathematical Sciences, HBNI, India
roohani@imsc.res.in

Meirav Zehavi
Ben-Gurion University of the Negev, Beer-Sheva, Israel
meiravze@bgu.ac.il

Abstract

In the past decade, the design of *fault tolerant data structures* for networks has become a central topic of research. Particular attention has been given to the construction of a subgraph H of a given digraph D with as fewest arcs/vertices as possible such that, after the failure of any set F of at most $k \geq 1$ arcs, testing whether $D - F$ has a certain property $\mathcal{P}$ is equivalent to testing whether $H - F$ has that property. Here, reachability (or, more generally, distance preservation) is the most basic requirement to maintain to ensure that the network functions properly. Given a vertex $s \in V(D)$, Baswana et al. [STOC'16] presented a construction of H with $\mathcal{O}(2^k n)$ arcs in time $\mathcal{O}(2^k nm)$ where $n = |V(D)|$ and $m = |E(D)|$ such that for any vertex $v \in V(D)$: if there exists a path from s to v in $D - F$, then there also exists a path from s to v in $H - F$. Additionally, they gave a tight matching lower bound. While the question of the improvement of the dependency on k arises for special classes of digraphs, an arguably more basic research direction concerns the dependency on n (for reachability between a pair of vertices $s, t \in V(D)$) – which are the largest classes of digraphs where the dependency on n can be made sublinear, logarithmic *or even constant*? Already for the simple classes of directed paths and tournaments, $\Omega(n)$ arcs are mandatory. Nevertheless, we prove that "almost acyclicity" suffices to eliminate the dependency on n *entirely* for a broad class of dense digraphs called *bounded independence digraphs*. Also, the dependence in k is only a polynomial factor for this class of digraphs. In fact, our sparsification procedure extends to preserve parity-based reachability. Additionally, it finds notable applications in Kernelization: we prove that the classic Directed Feedback Arc Set (DFAS) problem as well as Directed Edge Odd Cycle Transversal (DEOCT) (which, in sharp contrast to DFAS, is W[1]-hard on general digraphs) admit polynomial kernels on bounded independence digraphs. In fact, for any $p \in \mathbb{N}$, we can design a polynomial kernel for the problem of hitting all cycles of length ℓ where ($\ell \mod p = 1$). As a complementary result, we prove that DEOCT is NP-hard on tournaments by establishing a combinatorial identity between the minimum size of a feedback arc set and the minimum size of an edge odd cycle transversal. In passing, we also improve upon the running time of the sub-exponential FPT algorithm for DFAS in digraphs of bounded independence number given by Misra et at. [FSTTCS 2018], and give the first sub-exponential FPT algorithm for DEOCT in digraphs of bounded independence number.

11th Innovations in Theoretical Computer Science Conference (ITCS 2020).
Editor: Thomas Vidick; Article No. 47; pp. 47:1–47:22

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

2012 ACM Subject Classification Theory of computation → Data structures design and analysis; Theory of computation → Parameterized complexity and exact algorithms; Theory of computation → Sparsification and spanners

Keywords and phrases sparsification, kernelization, fault tolerant subgraphs, directed feedback arc set, directed edge odd cycle transversal, bounded independence number digraphs

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.47

Funding *Daniel Lokshtanov*: European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant no. 715744), and United States - Israel Binational Science Foundation grant no. 2018302.
Saket Saurabh: European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant no. 819416), and Swarnajayanti Fellowship grant DST/SJF/MSA-01/2017-18.
Meirav Zehavi: Israel Science Foundation (ISF) individual research grant (grant no. 1176/18) and Israel Binational Science Foundation (BSF) start up grant.

1 Introduction

In most real-life applications, even the most reliable networks are highly prone to unexpected failures of a small number of links that connect their nodes. In the past decade, the design of *fault tolerant data structures* for networks has become a central topic of research [7, 9, 13, 47, 44, 12, 16, 17, 18, 11, 26, 45, 46]. Generally, the scenario under study concerns the design of a structure that, after the failure of any set F of at most $k \geq 1$ arcs (representing links) in a given digraph D (representing a network), should provide a fast answer to certain types of queries that address the properties of $D - F$. The most common queries of this form address the *reachability* between two vertices, or, more generally, the length of a shortest path existent, if any, between them. Indeed, reachability (or, more generally, distance preservation) is the most basic requirement to maintain to ensure that the network functions properly. In this context, particular attention has been given to the case where the data structure should consist of a subgraph or a minor of D with as fewest arcs/vertices as possible [7, 47, 9, 11, 8, 45, 13]. Then, queries can be answered by standard means as the usage of BFS or Dijkstra's algorithm. In particular, these simple data structures are of interest as they also double as *sparsifiers*. The study of various graph sparsifiers – such as *flow-sparsifiers* [38] which are closely related to the aforementioned data structures – is a fundamental, active area of research in computer science and structural graph theory [22, 5, 29, 38, 15].

More concretely, in the Fault-Tolerance (S,T)-Reachability problem (or FTR(S,T) for short), we are given a digraph D, two (not necessarily disjoint) terminals sets $S, T \subseteq V(D)$, and a positive integer k. The objective is to construct a subgraph H of D with minimum number of arcs/vertices such that, after the failure of any set of at most k arcs in D, the following property is preserved for any two vertices $s \in S$ and $t \in T$: if there still exists a directed path from s to t in D, then there also still exists a directed path from s to t in H. Clearly, a trivial lower bound on the number of arcs in H is $m = \Omega(n^2)$. For the case where $|S| = 1$ and $T = V(D)$, Baswana et al. [9] presented a construction of a subgraph H with $\mathcal{O}(2^k n)$ arcs in time $\mathcal{O}(2^k nm)$ where $n = |V(D)|$ and $m = |E(D)|$. Additionally, they gave a tight matching lower bound: for any $n, k \in \mathbb{N}$ where $n \geq 2^k$, there exists a digraph on n vertices where H must have $\Omega(2^k n)$ arcs.

Naturally, the question of the improvement of the dependency on k arises for special classes of digraphs. However, an arguably more radical research direction to pursue concerns the dependency on n.

Which are the largest classes of digraphs for which FTR(S,T) admits subgraphs whose size dependency on n can be made sublinear, logarithmic *or even constant*?

At first glance, when we consider the simplest sparsest digraph existent, this pursuit seems futile. Indeed, already in the case where $S = \{s\}$, $T = \{t\}$, $k = 1$ and D is a directed path from s to t, the only solution is to choose $H = D$. At second glance, when we consider the simplest densest digraph existent, again we reach a dead-end: for S, T and k as before, define D as the tournament obtained by adding, to a directed path $s = v_1 \to v_2 \to \ldots \to v_n = t$, all arcs going from v_i to v_j for every $j + 1 < i$; then, to construct H, we must select the entire path.

We show that "almost acyclicity" suffices to eliminate the dependency on n *entirely* for a broad class of dense digraphs called *bounded independence number digraphs*. Furthermore, one can achieve a polynomial dependence in terms of k for this digraph class.

To step beyond the strict confinement of tournaments where *all* relations (arcs) between the input entities (vertices) must be both *present* and *known*, Fradkin and Seymour [33] initiated the study of bounded independence digraphs. Formally, for any integer $\alpha \geq 1$, the class of α-bounded independence digraphs, denoted by $\mathcal{D}_\alpha$, is defined as follows.

$$\mathcal{D}_\alpha = \{D \mid D \text{ is a digraph and the maximum size of an independent set in } D \text{ is at most } \alpha\}.$$

chitnis2014directedFor this class of digraphs, Fradkin and Seymour [33] studied the k-Disjoint Paths problem, and showed that it admits a polynomial time algorithm for any fixed value of k. Observe that $\mathcal{D}_\alpha$ is *hereditary*, and for $\alpha = 1$, it coincides with the class of tournaments. Furthermore, even for $\alpha = 2$, it contains digraphs with a linear fraction of vertex pairs that have no arc between them – thus, it can accommodate the lack of a large number of links/relations.

Our main technical contribution is the following combinatorial lemma.

▶ **Lemma 1.1.** *Given a digraph $D \in \mathcal{D}_\alpha$, positive integers k and ℓ, and $S \subseteq V(D)$ such that every strongly connected component of $D - S$ has at most ℓ vertices, the* Fault-Tolerance (S,S)-Reachability (FTR(S,S)) *problem admits a solution H on $|S|^2(k\ell)^{\mathcal{O}(4^{\alpha\ell^2})}$ vertices. Furthermore, such a solution H can be found in polynomial time.*

In particular, when $D - S$ is acyclic, $\ell = 1$. Thus, if $|S|$ and ℓ are independent of n (such as the case where $|S| = |T| = \ell = 1$ discussed earlier), the dependency on n is eliminated. (We remark that a solution for Fault-Tolerance (S,T)-Reachability where $S \neq T$ is subsumed by a solution for Fault-Tolerance $(S \cup T, S \cup T)$-Reachability.) Note that we extend the class of digraphs dealt with beyond acyclicity at two fronts: enabling S to be a modulator, thus $D - S$ rather than D should be "almost acyclic"; enabling the strongly connected components to be of size that is ("small" but) larger than 1.

In fact, our result generalizes to *parity reachability*. More precisely, in the Fault-Tolerance (S,T)-Parity Reachability problem, we are given a digraph D, two terminal sets $S, T \subseteq V(D)$, positive integers k and p, and a non-negative integer r. The objective is to construct a subgraph H of D with as few arcs/vertices as possible, such that, after the failure of any set of at most k arcs in D, the following property is preserved for any two vertices $s \in S$ and $t \in T$: if there exists a directed path from s to t in D whose length q satisfies $(q \mod p = r)$, then there also exists a directed path from s to t in H whose length q' satisfies $(q' \mod p = r)$. For this problem, we prove the following combinatorial lemma.

▶ **Lemma 1.2.** *Given a digraph $D \in \mathcal{D}_\alpha$, positive integers k, ℓ, p, a non-negative integer r, and $S \subseteq V(D)$ such that every strongly connected component of $D - S$ has at most ℓ vertices, the* Fault-Tolerance (S,S)-Parity Reachability *problem admits a solution H on $(|S|\alpha\ell pk)^{\mathcal{O}(4^{\alpha\ell^2})}$ vertices. Furthermore, such a solution H can be found in polynomial time.*

1.1 Applications in Kernelization

Directed Feedback Arc Set. From the perspective of Parameterized Complexity, with the exception of Directed Multicut, the Directed Feedback Arc/Vertex Set (DFA/VS) problem is the most well studied parameterized problem on digraphs. (On general digraphs, the vertex and arc versions of the problem are equivalent [23].) Formally, this problem is defined as follows.

Directed Feedback Arc Set (DFAS)	**Parameter:** k
Input: A digraph D and a non-negative integer k.	
Question: Does there exist $S \subseteq E(D)$ of size at most k such that $D - S$ is a DAG?	

We remark that this problem is among Karp's 21 original NP-complete problems [35]. Already a decade ago, the DFAS problem has been shown to be *fixed-parameter tractable (FPT)* parameterized by the solution size k [19]. Specifically, Chen et al. [19] developed an algorithm that solves DFAS in time $\mathcal{O}(k!4^k k^4 mn)$, based on the powerful machinery of important separators [23]. Since then, the quest to assert the existence of a polynomial kernel for this problem has been unfruitful. Over the years, it has been repeatedly posed as a major challenge in the subfield of Kernelization [23, 28, 42, 41] (also see [1] for a number of workshops and schools where it was posed as an open problem). In fact, the two specific problems whose polynomial kernelization complexity is completely unknown and their resolution is raised most frequently are DFAS and Multiway Cut [23, 28]. At the front of parameterized algorithms, the recent work by Lokshtanov et al. [39] improved upon the polynomial factor of the aforementioned algorithm by the design of an $\mathcal{O}(k!4^k k^5(m+n))$-time algorithm. It is known that unless the Exponential Time Hypothesis (ETH) is false, parameterized by the treewidth tw of the underlying undirected graph, DFAS cannot be solved in time $2^{o(\mathsf{tw}\log \mathsf{tw})} \cdot n^{\mathcal{O}(1)}$. However, it is unknown whether DFAS is solvable in time $2^{o(k \log k)} \cdot n^{\mathcal{O}(1)}$. In this regard, the only lower bound known is of $2^{\Omega(k)} \cdot n^{\mathcal{O}(1)}$ under the ETH [23, 39].

Particular attention has been given to the parameterized complexity of DFAS on tournaments. The classical complexity (NP-hardness) of DFAS on tournaments has a curious history. More than two decades ago, this problem was conjectured to be NP-hard by Bang-Jensen and Thomassen [6]. In 2008, Ailon et al. [2] proved that this problem does not admit a polynomial-time algorithm unless NP$\subseteq$BPP. Later, the reduction of Ailon et al. [2] was derandomized independently by Alon [3] and Charibt et al. [14], to prove that DFAS on tournaments is NP-hard. With respect to Parameterized Complexity, Alon et al. [4] proved that DFAS on tournaments admits a sub-exponential time parameterized algorithm (with running time $2^{\mathcal{O}(\sqrt{k}\log^2 k)} \cdot n^{\mathcal{O}(1)}$), to which end they introduced the method of chromatic coding. Later, the $\log^2 k$ factor in the exponent was shaved in independent works by Feige [30] and Karpinski and Schudy [36]. Fomin and Pilipczuk [32] presented a general approach, based on a bound on the number of k-cuts in transitive tournaments, to achieve the same running time for DFAS on tournaments. Based on this approach, Misra et al. [43] developed a sub-exponential time parameterized algorithm for DFAS on digraphs in $\mathcal{D}_\alpha$, with running time $2^{\mathcal{O}(\alpha^2\sqrt{k}\log(\alpha k))} \cdot n^{\mathcal{O}(\alpha)}$. Yet, the (arguably more) intriguing question

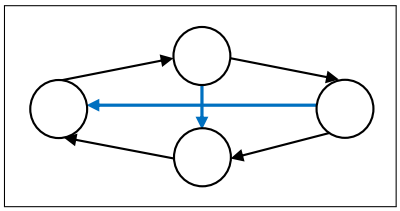


Figure 1 A directed edge odd cycle transversal (in blue) that is not a directed feedback arc set.

of the existence of a polynomial kernel for DFAS on digraphs in $\mathcal{D}_\alpha$ remained unsolved. On tournaments, Bessy et al. [10] have proved that DFAS admits a linear-vertex kernel (improving upon polynomial kernels given in [4, 27]). Based on our combinatorial lemma (Lemma 1.1), we establish the following theorem.

▶ **Theorem 1.3.** DFAS *on* $\mathcal{D}_\alpha$ *admits a kernel of size* $k^{\mathcal{O}(4^\alpha)}$.

In addition to its rich history in theoretical studies, the elimination of directed feedback loops is highly relevant to rank aggregation, Voting Theory, the resolution of inconsistencies in databases, and the prevention of deadlocks [48, 10, 34, 37, 19, 31]. While in a wide-variety of applications, *most* relations between the entities in a network are both present and known, it is generally unrealistic (in real-world partial and noisy data) that *all* relations will be so. Then, the usage of a bounded independence digraphs naturally comes into play. In passing, using Theorem 1.3, we also improve the running time for DFAS on digraphs in $\mathcal{D}_\alpha$, given by Misra et al. [43], by eliminating the dependence of α in the exponent of n. That is, we have the following theorem.

▶ **Theorem 1.4.** DFAS *on* $\mathcal{D}_\alpha$ *can be solved in* $2^{f(\alpha)\sqrt{k}\log k} \cdot n^{\mathcal{O}(1)}$, *where* $f(\alpha)$ *is some function of* α *and* n *is the number of vertices in* D.

Directed Edge Odd Cycle Transversal. The Directed Edge Odd Cycle Transversal (DEOCT) problem is the parity-based version of DFAS, formally defined as follows. (On general digraphs, the vertex and arc versions of the problem are equivalent [40]).

Directed Edge Odd Cycle Transversal (DEOCT)	**Parameter:** k
Input: A digraph D and a non-negative integer k.	
Question: Does there exist $S \subseteq E(D)$ of size at most k such that $D-S$ has no odd cycle?	

Observe that a tournament has no directed cycle if and only if it has no directed triangle (a cycle on three vertices). In turn, this simple observation implies that, given a tournament D, any subset S of the *vertices* of D has the following property: $D - S$ is a DAG if and only if it has no directed odd cycle. Thus, the vertex versions of DFAS and DEOCT on tournaments are equivalent. However, for DFAS and DEOCT the situation is not so clear. Indeed, it is not difficult to come up with a tournament D and a subset of *arcs* S of D such that $D - S$ is not a DAG, yet it has no directed odd cycle (see, e.g., Fig. 1). Nonetheless, we are able to prove that given a tournament D and a subset S of the *arcs* of D such that $D - S$ has no directed odd cycle, there exists a subset of arcs S' of D such that $D - S'$ is a DAG and $|S'| \leq |S|$. In particular, we thus establish the following result.

▶ **Theorem 1.5.** DEOCT *on tournaments is* NP*-hard.*

The question of the parameterized complexity of DEOCT was explicitly stated as an open problem [24] for the first time in 2007, immediately after the announcement of the first parameterized algorithm for DFAS. Since then, the problem has been re-stated several

times [20, 21, 42, 41]. Recently, Lokshtanov et al. [40] proved that DEOCT is W[1]-hard. Specifically, this means that DEOCT is highly unlikely to be FPT or admit a kernel of any size (even exponential in k). Based on the parity-based generalization of our combinatorial lemma (Lemma 1.2), we establish a polynomial kernel for DEOCT on $\mathcal{D}_\alpha$, which stands in sharp contrast to its aforementioned status on general digraphs.

▶ **Theorem 1.6.** DEOCT *on* $\mathcal{D}_\alpha$ *admits a kernel of size* $(\alpha k)^{\mathcal{O}(4^{4\alpha^3})}$.

In fact, we present combinatorial results stronger than Lemma 1.2 that yield a polynomial kernel for a more general version of DEOCT, where instead of hitting directed odd cycles, the objective is to hit directed cycles whose length ℓ satisfies $(\ell \mod p = 1)$ for an integer $p \in \mathbb{N}$ given as input.[1]

Modulo p Directed Cycle Transversal (mod(p)-DCT) **Parameter:** k
Input: A digraph D and non-negative integers k and p.
Question: Does there exist $S \subseteq E(D)$ of size at most k such that $D - S$ has no cycle of length $1 \mod p$?

▶ **Theorem 1.7.** mod(p)-DCT *on* $\mathcal{D}_\alpha$ *admits a kernel of size* $(p\alpha k)^{\mathcal{O}(4^{\alpha^3 p^2})}$.

Having Theorem 1.6 at hand, we also show how to employ the general approach of Fomin and Pilipczuk [32] to derive a sub-exponential time parameterized algorithm for DEOCT on digraphs in $\mathcal{D}_\alpha$.

▶ **Theorem 1.8.** DEOCT *on* $\mathcal{D}_\alpha$ *admits an algorithm with running time* $2^{\mathcal{O}(f(\alpha)\sqrt{k}\log k)} \cdot n^{\mathcal{O}(1)}$, *where* $f(\alpha)$ *is a function of* α *and* n *is the number of vertices in* D.

1.2 Towards the proof of Lemmas 1.1: Cut Preserving Sets

The most central notion in this paper is of a *cut preserving set*. Informally, for a digraph D, a pair of vertices s, t and an integer k, a set $\mathcal{Z} \subseteq V(D)$ is called a k-cut preserving set[2] for (s, t) in D if it preserves all (s, t)-arc cuts of size at most k. That is, A is an (s, t)-arc cut with at most k arcs in D if and only if A is a such a cut in $D[\mathcal{Z}]$. Observe that the graph induced on such a k-cut preserving set $\mathcal{Z}$ is a candidate solution for FTR($\{s\}, \{t\}$) problem. Clearly $V(D)$ is a k-cut preserving set for any pair of vertices s, t. The intent is to have such a set of "small" size. Towards this, let us discuss some properties that suffice for $\mathcal{Z}$ to be a k-cut preserving set for (s, t) in D.

Since $\mathcal{Z} \subseteq V(D)$, any (s, t)-arc cut of D is an (s, t)-arc cut of $D[\mathcal{Z}]$. For the other direction, we need the property that, for any $A \subseteq E(D)$ of size at most k, the existence of an (s, t)-path in $D - A$ implies the existence of an (s, t)-path in $D[\mathcal{Z}] - A$. Let us now see which properties suffice to imply the above property. We begin with a special case. Suppose there is a "large" flow from s to t in D. In particular, suppose there are at least $k + 1$ internally vertex-disjoint (s, t)-paths in D. Then, in $\mathcal{Z}$ it is enough to keep the vertices of some $k + 1$ vertex-disjoint (s, t)-paths, as no arc set of size at most k can hit all these paths. The more involved case occurs when the flow from s to t in D is at most k. Consider any (s, t)-path P in D. Ideally (if we did not have a size constraint on $\mathcal{Z}$) we would have preserved all the

[1] Note that a fundamental difference between this result and Lemma 1.2 is that the latter only works for any modulo and not just 1.

[2] This is not the way it is defined later. However, for the sake of exposition, we start with this definition and refine it to have properties that also guarantee this property implicitly.

vertices of P in $\mathcal{Z}$. Clearly, this can be expensive in terms of the size of $\mathcal{Z}$. Nevertheless, we can merge the ideas above (the "large-flow idea" and the "keep-full-path idea") to get the desired result. To see this, let P be a (s,t)-path in D. Let $\mathcal{Z}$ be a set of vertices such that, either all the vertices of P are in $\mathcal{Z}$ or if the vertices of a (u,v)-subpath of P are not in $\mathcal{Z}$, then there are $k+1$ internally vertex-disjoint (u,v)-paths in $D[\mathcal{Z}]$. That is, if the vertices of a subpath are missing in $\mathcal{Z}$, then $\mathcal{Z}$ contains a witness of a large flow for the endpoints of this subpath. Observe that such a set $\mathcal{Z}$ suffices to be a k-cut preserving set for (s,t) in D. This is because if P is an (s,t)-path in $D-A(A \subseteq E(D)$ and $|A| \leq k)$, then either all the vertices of P are in $\mathcal{Z}$ or for any missing (u,v)-subpath of P, since there are $k+1$ vertex-disjoint (u,v)-paths in $D[\mathcal{Z}]$, at least one still remains in $D[\mathcal{Z}]-A$. Thus, in $D[\mathcal{Z}]-A$, one can find an (s,t)-path: for the missing subpaths of P in $\mathcal{Z}$, there exists some (other) path between the same endpoints in $D[\mathcal{Z}]-A$ which together yield an (s,t)-walk (and hence an (s,t)-path) in $D[\mathcal{Z}]-A$. These properties are formalized in Definition 3.1.

1.2.1 About Computing k-Cut Preserving Sets

Next we give an intuition for how one can compute such k-cut preserving sets for a digraph $D \in \mathcal{D}_\alpha$, each of whose strongly connected component has size at most ℓ. For exposition purposes, consider (for now), only the case where D is acyclic (i.e. $\ell = 1$). With a certain technical argument, the general case reduces to this one. Moreover, we use the definition of a k-cut preserving set from the beginning of this section for this illustration as it allows us to convey our ideas in a clearer manner.

The proof will use induction on α. As the base case, consider the case when $\alpha = 1$, that is, D is a transitive tournament. As D is transitive, there exists a topological ordering of the vertices of D. Consider the set S of vertices between s and t in this ordering. Note that any path from s to t only uses vertices in S. So, either S is smaller than $k+1$, and then $S \cup \{s,t\}$ is a k-cut preserving set for (s,t), or it can be seen that there is no arc-cut for (s,t) of size at most k. In the latter case, the union of $\{s,t\}$ and any subset of $k+1$ vertices of S is a k-cut preserving set for (s,t); indeed, in the subgraph induced by the union there is still no arc-cut for (s,t) of size at most k.

Now, let us hint at how the inductive step of the proof works. First, we note that, if $P_1, \ldots, P_{k+1}$ are $k+1$ internally vertex-disjoint (s,t)-paths, then $\mathcal{Z} = \cup_{i \in [k+1]} P_i$ is a k-cut preserving set for the pair (s,t), because there is no arc-cut of (s,t) in both D and $D[\mathcal{Z}]$ of size at most k. Moreover, since D is acyclic and $D \in \mathcal{D}_\alpha$, if these paths exist, then Observation 2.1 implies that we can assume that all these paths are shorter than $2\alpha+1$ and thus $|\mathcal{Z}| \leq k(2\alpha+1)$.

The last argument means that we can assume the existence of a (s,t)-vertex cut of size at most k. For simplicity, suppose that $\{c_1, c_2\}$ is a minimal (s,t)-vertex -cut. Since $\{c_1, c_2\}$ is a vertex cut, any path from s to t in D can be decomposed as a path from s to c_i, a path from c_i to c_j and then a path from c_j to t, where i and j are two indices (possibly equal) in $\{1,2\}$. Here, we mean that none of the three paths contains c_i (or c_j) as an internal vertex. For $i \in \{1,2\}$, let S_i be the union of the set of vertices of the paths from s to c_i that intersect $\{c_1, c_2\}$ only on the last vertex, and T_i be the union of the set of vertices of the paths from c_i to t that intersect $\{c_1, c_2\}$ only on the first vertex. Finally, for distinct $i, j \in \{1,2\}$, let $C_{i,j}$ be the union of the set of vertices of the paths from c_i to c_j. Because of the last remark on how any path from s to t can be decomposed, taking the union of six k-cut preserving sets-namely, for each $i, j \in \{1,2\}$, $i \neq j$, for (s, c_i) in $D[S_i]$, (c_i, t) in $D[T_i]$ and (c_i, c_j) in $D[C_{i,j}]$- gives a k-cut preserving set for (s,t) in D. Now, the question is how to use the induction hypothesis to find a k-cut preserving set for each of these pairs. Consider first

the digraph induced by the vertices in S_1. Because $\{c_1, c_2\}$ is a minimal (s,t)-vertex cut, the only vertices of S_1 that can possibly have "outgoing arcs towards" t in S_1 are s and c_1. Moreover, since $\{c_1, c_2\}$ is a minimal (s,t)-vertex cut, there exists a path from c_1 to t in D and thus t is reachable from any vertex of S_1. However, since D is acyclic, this means that there is no arc from t to any of the vertices of S_1, else we would get a closed walk and thus a cycle. This implies that $D[S_1 \setminus \{s, c_1\}] \in \mathcal{D}_{\alpha-1}$ as any independent set of $S_1 \setminus \{s, c_1\}$ can be extended with t. We cannot apply the induction hypothesis to find a k-cut preserving set for (s, c_1) in S_1 because the independence number of $D[S_1]$ could be equal to α, however the above shows the spirit of the arguments that will be used to find subgraphs with smaller independence number where we can apply the induction hypothesis. A similar argument would also give that the independence number of $D[T_1 \setminus \{c_1, t\}]$ is at most $\alpha - 1$ as any independent set can be extended using s.

The previous argument does not apply to $C_{1,2}$, because the vertices of $C_{1,2}$ can be adjacent to s or t (some vertices of $C_{1,2}$ can be adjacent to s and some can be adjacent to t). This is the case that requires a stronger and more technical definition for a k-cut preserving set. In particular, we need to understand what happens to the vertices of D that are on a path from s to t but do not belong to a k-cut preserving set for this pair.

1.3 Deriving Polynomial Kernels for DFAS

Let us now briefly explain how to derive a polynomial kernel for DFAS when the input digraph belongs to $\mathcal{D}_\alpha$, from our result on fault-tolerant subgraphs. First note that if $D \in \mathcal{D}_\alpha$ then every induced cycle in D has length at most $2\alpha+1$. Let (D, k) be an instance of DFAS, and consider a maximal set of arc disjoint induced cycles in D. If this set consists of more than k cycles, then any solution to (D, k) has to pick one arc per cycle, and (D, k) is a NO instance. If not, let S be the union of these cycles. S is a set of less than $(2\alpha+1) \cdot k$ vertices such that $D - S$ is acyclic. Therefore, we can apply our result to find a solution H to the problem of Fault-Tolerance (S, S)-Reachability of size at most $|S|^2 k^{\mathcal{O}(4^\alpha)}$. We claim that H is the desired kernel. Indeed, suppose that A is a set of arcs such that $H - A$ is acyclic, but $D - A$ contains a cycle. By construction of S, this cycle must use vertices of S. However, we know that if a path exists between two vertices of S in $D - A$, then such a path also exists in $H - A$. This implies the existence of a closed walk in $H - A$, a contradiction.

2 Preliminaries

For standard notations and terminology that is not defined here, we refer to [25].

Sets. For positive integer i, j, $[i]$ denotes the set $\{1, \ldots, i\}$ and $[i, j]$ denote the set $\{i, i+1, \ldots, j\}$. For a set S, S^2 denotes the set of ordered pairs of S, that is $S^2 = \{(u, v) \mid u \in S, v \in S\}$.

Digraphs. For a digraph D, $V(D)$ denotes the vertex set of D and $E(D)$ denotes the arc set of D. For any $X \subseteq V(D)$ (resp. $X \subseteq E(D)$), $D - X$ denotes the digraph obtained by deleting the vertices (resp. edges) of X. For any $v \in V(D)$, $N^+_D(v)$ (resp. $N^-_D(v)$) denotes the set of out-neighbours (resp.in-neighbours) of v in D, that is $N^+_D(v) = \{u \in V(D) \mid (v, u) \in E(D)\}$ (resp. $N^-_D(v) = \{u \in V(D) \mid (u, v) \in E(D)\}$). Whenever the digraph D is clear from the context, we drop the subscript D in $N^+_D(v)$ (resp. $N^-_D(v)$). For any $X, Y \subseteq V(D)$, $E(X, Y)$ denotes the set of arcs of D with tail in X and head in Y, that is, $E(X, Y) = \{(u, v) \in E(D) \mid u \in X, v \in Y\}$. A digraph D is called *strongly connected* if for each $u, v \in V(D)$ there

is a path from u to v and, a path from v to u in D. A set $X \subseteq V(D)$ is called a *strongly connected component* of D if $D[X]$ is a strongly connected digraph and for each $X' \supseteq X$, $D[X']$ is not a strongly connected digraph. A *tournament* is a digraph where there is exactly one arc between each pair of vertices. A digraph with no cycles is called a *directed acyclic graph (dag)*. A tournament with no cycles is called a *transitive tournament*.

Paths. A *path* P is a graph such that there exists an ordering $(v_1, \ldots, v_q)$ of its vertex set $V(P)$ such that $E(P) = \{(v_i, v_{i+1}) \mid i \in [q-1]\}$. Such a path is called a (v_1, v_q)-*path*, v_1, v_q are called the *end-points of* P and $v_2, \ldots, v_{q-1}$ are called the *internal vertices of* P. A path P is *even* (resp. odd) if the number of arcs/edges in it is even (resp. odd). We say that P is a path in the digraph D if P is a subgraph of D. We say that P is an *induced path* in D if P is an induced subgraph of D. For paths P and P', by $P \circ P'$ we denote the composition of P and P', that is, the path obtained by appending P' after P. For paths $P, P_1, P_2, \ldots, P_q$ such that $P = P_1 \circ P_2 \circ \ldots \circ P_q$, we say that $P_1 \circ P_2 \circ \ldots \circ P_q$ is a *partition* of P. For a digraph D and $X \subseteq V(D)$, we say that a (u, v)-path P in D is X-*free* if none of the internal vertices of P are from X. The X-*based partition of* P in D is the partition $P = P_1 \circ \ldots \circ P_q$ such the union of the end-points of P_i, $i \in [q]$, is exactly the set $(X \cap V(P)) \cup \{u, v\}$. A *semi-$X$-based partition* of P, $P = P_1 \circ \ldots \circ P_q$, is such that the end-points of the paths P_i, $i \in [q]$, are a subset of $(X \cap V(P)) \cup \{u, v\}$. Paths $\{P_1, \ldots, P_q\}$ are *internally vertex-disjoint* if for all distinct $i, j \in [q]$, the sets of internal vertices of P_i and P_j are disjoint.

Vertex and Arc Cuts. For a digraph D and $u, v \in V(D)$, a (u, v)-*arc cut* is a set of arcs of D, say X, such that $D - X$ has no (u, v)-path. A (u, v)-*vertex cut* is a set of vertices of D, say Y, such that $D - Y$ has no (u, v)-path and $u, v \notin Y$ if $(u, v) \notin E(D)$.

▶ **Observation 2.1.** *Let $D \in \mathcal{D}_\alpha$. The length of the shortest cycle in D is at most $2\alpha + 1$. Also, the length of any induced path in D is at most $2\alpha + 1$.*

In this article, we focus of the proof of Lemma 1.1 alone. The proofs of Lemma 1.2 and Theorems 1.6, 1.8, 1.5, 1.6, 1.7, 1.8 will be made available later in the full version of the paper.

3 Finding Small k-Cut Preserving Sets

We give the precise definition of a k-cut preserving set here.

▶ **Definition 3.1** (k-Cut Preserving Set). *For digraph D, an ordered pair (u, v) of vertices of D and a positive integer k, $\{u, v\} \subseteq \mathcal{Z} \subseteq V(D)$ is a* k-cut preserving set for (u, v) in D *if the following holds. For any (u, v)-path P in D, there exists a semi-$\mathcal{Z}$-based partition $P_1 \circ \ldots \circ P_d$ of P with the following two properties. For each $i \in [d]$, P_i is an (s_i, t_i)-path in D with $s_i, t_i \in \mathcal{Z}$. Moreover, either $V(P_i) \subseteq \mathcal{Z}$ or there exists a list $\mathcal{L}_i$ of $k+1$ internally vertex-disjoint $(V(D) \setminus \mathcal{Z})$-free (s_i, t_i)-paths. A list $\mathcal{L}_i$ with the above property is called a* replacement kit for P_i in $\mathcal{Z}$. *Such a semi-$\mathcal{Z}$-based partition of P is called a* $\mathcal{Z}$-replacement witness for P.

Before moving to the computational aspects of a k-cut preserving set, we give the following lemma that can be considered as the main utility of k-cut preserving sets, and relate to the intuition we gave in the previous section.

▶ **Lemma 3.2.** *Let D be a digraph, $u, v \in V(D)$ and $\mathcal{Z}$ be a k-cut preserving set for (u, v) in D. For any set $A \subseteq E(D)$ of at most k arcs, if there exists a (u, v)-path in $D - A$, then there also exists one in $D[\mathcal{Z}] - A$.*

Proof. Consider some $A \subseteq E(D)$ such that $|A| \leq k$. Suppose there exists a (u,v)-path P in $D - A$. Since $\mathcal{Z}$ is a k-cut preserving set for the pair (u,v), there exists a semi-$\mathcal{Z}$-based partition $P = P_1 \circ \ldots \circ P_d$ such that for each $j \in [d]$, P_j is an (s_j, t_j)-path, $s_j, t_j \in \mathcal{Z}$ and, either $V(P_j) \subseteq \mathcal{Z}$, in which case P_j is a path in $D[\mathcal{Z}] - A$, or there exist $k+1$ internally vertex- disjoint (s_j, t_j)-paths in $D[\mathcal{Z}]$. In the later case, at least one of the $k+1$ paths is in $D[\mathcal{Z}] - A$ (because $|A| \leq k$). This implies the existence of a walk from u to v (and hence also a (u,v)-path) in $D[\mathcal{Z}] - A$. This concludes the proof. ◀

The main goal of this section is to prove the following lemma.

▶ **Lemma 3.3** (k-Cut Preserving Lemma)**.** *Let D be an acyclic digraph, and $u, v \in V(D)$ be such that $N^-(u) = N^+(v) = \emptyset$. Additionally, let $D - \{u,v\} \in \mathcal{D}_\alpha$. Then there exists a k-cut preserving set for (u,v) in D of size at most $f(\alpha)$, where $f(1) = k^3 + 5k^2 + 3k$ and for $\alpha > 1$, $f(\alpha) = k^2 g(\alpha) + 2kh(\alpha)$, $g(\alpha) = (2k + (k + kf(\alpha-1))^2) f(\alpha-1)$ and $h(\alpha) = (k^2+k)g(\alpha) + kf(\alpha-1)$. Moreover, such a set can be found in time $n^{\mathcal{O}(1)}$, where $n = |V(D)|$.*

Note that $V(D)$ is always a k-cut preserving set for any pair of vertices (u,v) in D, for any k. We now define a notation, for the sake of convenience, that will be used throughout this section. For any digraph D, $u, v \in V(D)$ and $X \subseteq V(D)$, let $ver_D(u, v; X)$ denote the union of the sets of vertices of all X-free (u,v)-paths in D. Observe that $ver_D(u,v;X) \cap X \subseteq \{u,v\}$. We begin by making an observation that forms the base line for computing small sized k-cut preserving sets using an appropriate induction.

▶ **Observation 3.4.** *Let D be a digraph, $u, v \in V(D)$, $\mathcal{Z} \subseteq V(D)$ and k be a positive integer. Let P be a (u,v)-path in D, and $P = P_1 \circ \ldots \circ P_d$ be a semi-$\mathcal{Z}$-based partition of P. If for each $i \in [d]$, there is a $\mathcal{Z}_i$-replacement witness for P_i in D_i, for some $\mathcal{Z}_i \subseteq \mathcal{Z}$ and D_i subgraph of D, then there is a $\mathcal{Z}$-replacement witness for P.*

Proof. For each $i \in [d]$, let $P_I = P_{i,1} \circ \ldots \circ P_{i,c_i}$ be a $\mathcal{Z}_i$-replacement witness for $\mathcal{Z}_i$ in D_i. Then, consider the semi-$\mathcal{Z}$-based partition $P = P_{1,1} \circ \ldots \circ P_{1,c_1} \circ P_{2,1} \circ \ldots \circ P_{2,c_2} \circ \ldots \circ P_{d,1} \circ \ldots \circ P_{d,c_d}$. Then, for each $i \in [d]$ and $j \in [c_i]$, either $V(P_{i,j}) \subseteq \mathcal{Z}_i \subseteq \mathcal{Z}$, or there exists a list $\mathcal{Z}_{i,j}$ containing $k+1$ internally vertex-disjoint $(V(D_i) \setminus \mathcal{Z}_i)$-free $(x_{i,j}, y_{i,j})$-paths in D_i such that $P_{i,j}$ is a $(x_{i,j}, y_{i,j})$-path. Since $\mathcal{Z}_i \subseteq \mathcal{Z}$ and D_i is a subgraph of D, the paths in $\mathcal{L}_{i,j}$ are $(V(D) \setminus \mathcal{Z})$-free and exist in D. ◀

Next, we give two lemmas (Lemmas 3.5 and 3.6) that basically use Observation 3.4 in a more concrete setting required to prove the k-Cut Preserving Lemma by induction on the size of the maximum independent set in the digraph.

▶ **Lemma 3.5.** *Let D be a digraph, $u, v \in V(D)$ and k be a positive integer. Let C be some (u,v)-vertex cut in D. For each $c \in C$, let $\mathcal{Z}(u,c)$ (resp. $\mathcal{Z}(c,v)$) be a k-cut preserving set for (u,c) (resp. (c,v)) in $D[ver_D(u,c;C)]$ (resp. $D[ver_D(c,v;C)]$). For each $(c,c') \in C^2$, $c \neq c'$, let $\mathcal{Z}(c,c')$ be a k-cut preserving set for (c,c') in $D[ver_D(c,c';C)]$. Then, $\mathcal{Z} := \bigcup_{c \in C} (\mathcal{Z}(u,c) \cup \mathcal{Z}(c,v)) \cup \bigcup_{(c,c') \in C^2, c \neq c'} \mathcal{Z}(c,c')$ is a k-cut preserving set for (u,v) in D.*

Proof. First observe, from the definition of a k-cut preserving set and the construction of $\mathcal{Z}$, that $C \subseteq \mathcal{Z}$. Consider any (u,v)-path P in D. Let $P = P_1 \circ \ldots \circ P_q$ be the C-based partition of P. Since $C \subseteq \mathcal{Z}$, $P_1 \circ \ldots \circ P_q$ is a semi-$\mathcal{Z}$-based partition of P. Then P_1 is a C-free (u, c_1)-path in D for some $c_1 \in C$, P_q is a C-free (c_2, v)-path in D for some $c_2 \in C$, and for each $i \in [2, q-1]$, P_i is a C-free $(c_{j_i}, c_{j_{i'}})$-path in D, for some

$c_{j_i}, c_{j_i'} \in C$, $j_i \neq j_i'$. Thus, P_1 is a (u, c_1)-path in $D[ver_D(u, c_1; C)]$, P_q is a (c_2, v)-path in $D[ver_D(c_2, v; C)]$, and for each $i \in [2, q-1]$, P_i is a $(c_{j_i}, c_{j_i'})$-path in $D[ver_D(c_{j_i}, c_{j_i'}; C)]$. Since $\mathcal{Z}(u, c_1), \mathcal{Z}(c_2, v), \cup_{i \in [2,q-1]} \mathcal{Z}(c_{j_i}, c_{j_i'}) \subseteq \mathcal{Z}$, we are done by Observation 3.4. ◀

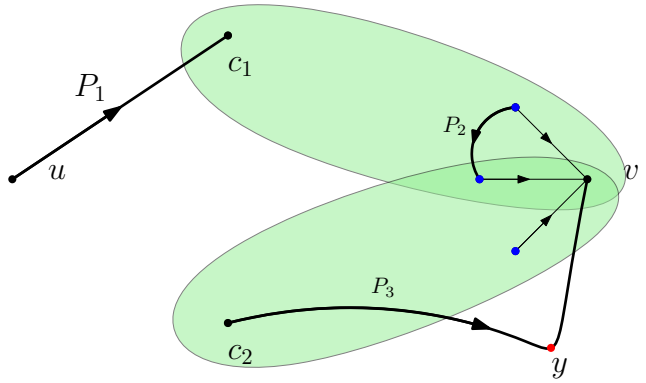


Figure 2 (c_1, c_2) is a (u, v) vertex-cut, the green parts correspond to the $\mathcal{Z}(c_i, v)$ and the blue vertices are the vertices of X. P_1 is a path of Type $(u, \square)$, P_2 is a path of Type $(\boxtimes, \boxtimes)$ and P_3 is a path of Type $(\square, \boxminus, v)$ with $y \in Y$.

▶ **Lemma 3.6.** *Let D be a digraph, $u, v \in V(D)$, and k be a positive integer. Let C be some (u, v)-vertex cut in D. For each $c \in C$, let $\mathcal{Z}(u, c)$ (resp. $\mathcal{Z}(c, v)$) be a k-cut preserving set for (u, c) (resp. (c, v)) in $D[ver_D(u, c; C)]$ (resp. $D[ver_D(c, v; C)]$). Let $X = N_D^-(v) \cap \bigcup_{c \in C} \mathcal{Z}(c, v)$. For each $(\mathtt{a}, \mathtt{b}) \in (C \cup X)^2$, $\mathtt{a} \neq \mathtt{b}$, let $\mathcal{Z}(\mathtt{a}, \mathtt{b})$ be a k-cut preserving set for $(\mathtt{a}, \mathtt{b})$ in $D[ver_D(\mathtt{a}, \mathtt{b}; C \cup N_D^-(v))]$. Then, $\mathcal{Z} := \bigcup_{c \in C} (\mathcal{Z}(u, c) \cup \mathcal{Z}(c, v)) \cup \bigcup_{(\mathtt{a},\mathtt{b}) \in (C \cup X)^2, \mathtt{a} \neq \mathtt{b}} \mathcal{Z}(\mathtt{a}, \mathtt{b})$ is a k-cut preserving set for (u, v) in D.*

Proof. First observe that $\{u, v\} \cup C \cup X \subseteq \mathcal{Z}$. Let $Y = N_D^-(v) \setminus X$. We begin by defining some special types of paths (see Figure 2).

1. A path P *is of Type* $(u, \square)$ *(resp.* $(\square, v)$*)* if it is a C-free (u, c)-path (respectively (c, v)-path) in D for some $c \in C$.
2. A path P *is of Type* $(\boxtimes, \boxtimes)$ if it is a $(C \cup N_D^-(v))$-free $(\mathtt{a}, \mathtt{b})$-path in D for some $(\mathtt{a}, \mathtt{b}) \in (C \cup X)^2$.
3. A path P *is of Type* $(\square, \boxminus, v)$ if it is a (c, v)-path in D for some $c \in C$ and there exists $y \in V(P) \cap Y$ such that the (c, y)-subpath of P is C-free.[3]

We now begin with the proof of the lemma. Let P be some (u, v)-path. We need to show that there is a $\mathcal{Z}$-replacement witness for P. Let $P = P_1' \circ \ldots \circ P_q'$ be the $(C \cup X)$-based partition of P. If P is not Y-free, that is, $V(P) \cap Y \neq \emptyset$, let $s' \in [q]$ be the least integer such that $V(P_{s'}'] \cap Y \neq \emptyset$. If P is Y-free, let $s' = q$. Let $s \leq s'$ be the largest integer such that P_s is an $(\mathtt{a}, \mathtt{b})$-path, where $\mathtt{a} \in C$ and $\mathtt{b} \in C \cup X \cup \{v\}$. WE first show that such a s always exists. From the definition of s', either there exists some $y \in Y$ in $V(P_{s'}')$ or $v \in V(P_{s'}')$. In the later case, since C is a (u, v)-vertex cut, there exists $c \in C$ such that c appears on P. Since $P = P_1' \circ \ldots \circ P_q'$ is a $C \cup X$-based partition of P, there exists $s \leq s'$ such that P_s is a $(\mathtt{a}, \mathtt{b})$-path where $\mathtt{a} \in C$. In the former case again, since $y \in Y \subseteq N_D^-(v)$ and C is a (u, v)-vertex cut using previous arguments the existence of the desired s is guaranteed.

[3] Specifically, if there exists $y \in V(P) \cap Y$ with this property, then the first vertex of P that belongs to Y also has that property.

Consider the partition $P = P_1 \circ \ldots \circ P_s$, such that $P_i = P'_i$, if $i < s$ and $P_s = P'_s \circ P'_{s+1} \circ \ldots \circ P'_q$. Observe that, since $C \cup X \subseteq \mathcal{Z}$, $P = P_1 \circ \ldots \circ P_s$ is a semi-$\mathcal{Z}$-based partition of P.

▷ Claim 3.7. P_1 is a Type $(u, \square)$ path, for each $i \in [2, s-1]$, P_i is a Type $(\boxtimes, \boxtimes)$ path and, P_s is either a Type $(\square, v)$ or Type $(\square, \boxminus, v)$ path.

Proof. Recall that $P = P'_1 \circ \ldots \circ P'_q$ is the $(C \cup X)$-based partition of P. Thus, we have the following.

1. For each $i \in [q]$, P'_i is $(C \cup X)$-free path.
2. For each $i \in [2, q-1]$, P'_i is a $(\mathtt{a}, \mathtt{b})$-path, where $(\mathtt{a}, \mathtt{b}) \in (C \cup X)^2$.
3. Since C is a (u, v)-vertex cut in D and $X \subseteq N_D^-(v)$, P'_1 is a (u, c)-path for some $c \in C$.
4. From the choice of s, for each $i \in [s-1]$, $V(P'_i) \cap Y = \emptyset$. Since for $i \in [s-1]$, $P_i = P'_i$ and $X \cup Y = N_D^-(v)$, P_i is $(C \cup N_D^-(v))$-free.

Thus, from Points 2 and 4, for each $i \in [s-1]$, P_i is of Type $(\boxtimes, \boxtimes)$. Also, from Points 3 and 4, P_1 is of Type $(u, \square)$. We now show that P_s is of Type $(\square, v)$ or $(\square, \boxminus, v)$. From the choice of s and the construction of P_s, P_s is a (c, v)-path for some $c \in C$. If P is Y-free, then P_s is of Type $(\square, v)$, otherwise, P_s is of Type $(\square, \boxminus, v)$. ◁

For each $i \in [s]$, define $\mathcal{Z}_i$ and D_i as follows.

$$\mathcal{Z}_i = \begin{cases} \mathcal{Z}(u, c) & \text{if } i = 1,\ P_1 \text{ is a } (u, c)\text{-path}, c \in C \\ \mathcal{Z}(\mathtt{a}, \mathtt{b}) & \text{if } i \in [2, s-1],\ P_i \text{ is a } (\mathtt{a}, \mathtt{b})\text{-path}, (\mathtt{a}, \mathtt{b}) \in (C \cup X)^2 \\ \mathcal{Z}(c, v) & \text{if } i = s,\ P_s \text{ is a } (c, v)\text{-path}, c \in C \end{cases}$$

$$D_i = \begin{cases} D[ver_D(u, c; C)] & \text{if } i = 1, P_1 \text{ is a } (u, c)\text{-path}, c \in C \\ D[ver_D(\mathtt{a}, \mathtt{b}; (C \cup N_D^-(v))] & \text{if } i \in [2, s-1],\ P_i \text{ is a } (\mathtt{a}, \mathtt{b})\text{-path}, (\mathtt{a}, \mathtt{b}) \in (C \cup X)^2 \\ D[ver_D(c, v)] & \text{if } i = s,\ P_s \text{ is a } (c, v)\text{-path}, c \in C \end{cases}$$

Recall the construction of $\mathcal{Z}$ from the lemma statement. Observe that for each $i \in [s]$, $\mathcal{Z}_i \subseteq \mathcal{Z}$. From Observation 3.4, to give a $\mathcal{Z}$-replacement witness for P, it is enough to give a $\mathcal{Z}_i$-replacement witness for each P_i, in D_i, $i \in [s]$. Thus, the following claim will finish the proof of the lemma.

▷ Claim 3.8. For each $i \in [s]$, P_i has a $\mathcal{Z}_i$-replacement witness in D_i.

Proof. We prove the claim using the following cases.

- **Case $i = 1$:** From Claim 3.7, P_1 is a C-free (u, c)-path in D for some $c \in C$. Thus, P_1 is a (u, c)-path in D_1. Since Z_1 is a k-cut preserving set for (u, c) in D_i, there exists a $\mathcal{Z}_1$-replacement witness for P_1 in D_1.
- **Case $i \in [2, s-1]$:** From Claim 3.7, when $i \in [2, s-1]$, then P_i is a $(C \cup N_D^-(v))$-free $(\mathtt{a}, \mathtt{b})$-path in D for some $(\mathtt{a}, \mathtt{b}) \in (C \cup X)^2$. Thus, P_i is an $(\mathtt{a}, \mathtt{b})$-path in D_i. Since Z_i is a k-cut preserving set for $(\mathtt{a}, \mathtt{b})$ in D_i, there exists a $\mathcal{Z}_i$-replacement witness for P_i in D_i.
- **Case $i = s$:** From Claim 3.7, P_s is of either Type $(\square, v)$ or Type $(\square, \boxminus, v)$.
  - **P_s is of Type $(\square, v)$:** From the definition of Type $(\square, v)$, P_s is a C-free (c, v)-path in D, for some $c \in C$. Thus, P_s is a (c, v)-path in D_s. Since $\mathcal{Z}_s$ is a k-cut preserving set for (c, v) in D_s, there exists a $\mathcal{Z}_s$-replacement witness for P_s in D_s.

- P_s **is of Type** $(\square, \boxminus, v)$**:** From the definition of Type $(\square, \boxminus, v)$, P_s is a (c, v)-path in D, for some $c \in C$, and there exists $y \in V(P) \cap Y$ such that the (c, y)-subpath of P is C-free. Let $P_s^\dagger$ be the (c, y)-subpath of P. Recall that $Y = N_D^-(v) \setminus X$. Consider the (c, v)-path in D, denoted by $\widetilde{P_s}$, obtained by appending the arc (y, v) at the end of $P_s^\dagger$. That is, $\widetilde{P_s} = P_s^\dagger \circ (y, v)$. Since $P_s^\dagger$ is a C-free path, so is $\widetilde{P_s}$. Thus $\widetilde{P_s}$ is a (c, v)-path in D_s. Since $\mathcal{Z}_s$ is a k-cut preserving set for (c, v) in D_s, there exists a semi-$\mathbb{Z}_s$-based partition of $\widetilde{P_s}$ which is a $\mathcal{Z}_s$-replacement witness for $\widetilde{P_s}$ in D_s. Let $\widetilde{P_s} = \widetilde{P_{s,1}} \circ \ldots \circ \widetilde{P_{s,r}}$ be one such partition. Since $y \in Y = N_D^-(v) \setminus X$ and $\mathcal{Z}_s \subseteq X$, $y \notin \mathcal{Z}_s$. Thus, y is an internal vertex of $\widetilde{P_{s,r}}$. Let $\widetilde{P_{s,r}}$ be an (x, v)-path. Clearly, $x \in \mathcal{Z}_s$ because $\widetilde{P_s} = \widetilde{P_{s,1}} \circ \ldots \circ \widetilde{P_{s,r}}$ is a semi-$\mathcal{Z}_s$-based partition. Let $P_{s,r}^\dagger$ be the (x, v)-subpath of $P_{s,r}$. We claim that $P_s = \widetilde{P_{s,1}} \circ \ldots \circ \widetilde{P_{s,r-1}} \circ P_{s,r}^\dagger$ is a semi-$\mathcal{Z}_s$-based partition of P_s and is also a $\mathcal{Z}_s$-replacement witness for P_s in D_s. It is clear from the discussion above that $P_s = \widetilde{P_{s,1}} \circ \ldots \circ \widetilde{P_{s,r-1}} \circ P_{s,r}^\dagger$ is a semi $\mathcal{Z}_s$-based partition of P_s. We will now show that it is a $\mathcal{Z}_s$-replacement witness for P_s in D_s.
  Since $\widetilde{P_s} = \widetilde{P_{s,1}} \circ \ldots \circ \widetilde{P_{s,r}}$ is a $\mathcal{Z}_s$-replacement witness for $\widetilde{P_s}$, we have that for each $j \in [r]$, either $V(\widetilde{P_{s,j}}) \subseteq \mathcal{Z}_s$ or there exists a list $\mathcal{L}_j$ containing $k+1$ vertex disjoint paths from the start vertex of $\widetilde{P_{s,j}}$ to its end vertex. Also, since $y \notin \mathcal{Z}_s$ and y is an internal vertex of $\widetilde{P_{s,r}}$, $V(\widetilde{P_{s,r}}) \not\subseteq \mathcal{Z}_s$. Thus, there is a list $\mathcal{L}_r$ containing $k+1$ vertex disjoint (x, v)-paths (recall x and v are the start and end vertices, respectively, of $\widetilde{P_{s,r}}$). Since $P_s = \widetilde{P_{s,1}} \circ \ldots \circ \widetilde{P_{s,r-1}} \circ P_{s,r}^\dagger$, and $P_{s,r}^\dagger$ is an (x, v)-path, from the above discussion for each $j \in [r-1]$, either $V(\widetilde{P_{s,j}}) \subseteq \mathcal{Z}_s$ or there exists a list $\mathcal{L}_j$ containing $k+1$ vertex disjoint paths from the start vertex of $\widetilde{P_{s,j}}$ to its end vertex. Also, there exists a list, $\mathcal{L}_r$, containing $k+1$ vertex disjoint paths from the start vertex of $P_{s,r}^\dagger$ to its end vertex. This completes the proof of the claim. ◁

As argued earlier, this completes the proof of the lemma. ◀

3.1 Finding a Small k-Cut Preserving Set for a Pair with Large Flow

As explained in Section 1.2, the proof of Lemma 3.3 will distinguish whether there is a k vertex-cut for (s, t) or not. The case where there is a no k vertex-cut is the easiest one, and will be dealt with the following lemma by simply keeping $k+1$ vertex disjoint paths.

▶ **Lemma 3.9.** *Let $D \in \mathcal{D}_\alpha$ be an acyclic digraph and $u, v \in V(D)$ be such that each (u, v)-vertex cut in D has size at least $k+1$. Then, a k-cut preserving set for (u, v) in D of size at most $(2\alpha - 1)(k+1) + 2$ exists and is computable in $n^{\mathcal{O}(1)}$ time, where $n = |V(D)|$.*

Proof. Since every (u, v)-vertex cut in D has size at least $k+1$, from Menger's Theorem, there are at least $k+1$ vertex-disjoint (u, v)-paths in D. Let $Q'_1, \ldots, Q'_{k+1}$ be a collection of some $k+1$ of these paths. We will now obtain a collection of $Q_1, \ldots Q_{k+1}$ vertex disjoint paths where the length of each Q_i is at most $2\alpha + 1$. To this end, we define each Q_i as some shortest (u, v)-path using the vertices of $V(Q'_i)$. We first claim that the length of Q_i is at most $2\alpha + 1$. For the sake of contradiction, suppose not. Then, from Observation 2.1, there exist $x, y \in V(Q_i)$ such that $(x, y) \in E(D)$. Since D is acyclic, x appears before y in the path Q_i. This contradicts that Q_i is a shortest (u, v)-path in $V(Q'_i)$. Let $\mathcal{Z} = \bigcup_{i \in [k+1]} V(Q_i)$. Clearly, $\{u, v\} \subseteq \mathcal{Z}$ and $|\mathcal{Z}| \leq (2\alpha - 1)(k+1) + 2$. The size bound follows because the length of each Q_i is at most $2\alpha + 1$, and u, v are the vertices common in each Q_i. To show that $\mathcal{Z}$ is a k-cut preserving set for (u, v) in D, consider the semi-$\mathcal{Z}$-based partition of P that is P itself. Then, $\{Q_1, \ldots, Q_{k+1}\}$ is the list for P containing $k+1$ internally vertex-disjoint $(V(D) \setminus \mathcal{Z})$-free (u, v)-paths. ◀

3.2 Finding a Small k-Cut Preserving Set of a Pair in a Tournament

As explained before, the proof of Lemma 3.3 will use induction on α. The next lemma handles the base case where $\alpha = 1$. It is somewhat more complicated compared to the arguments in Section 1.2; the reason for the complication is that we consider the digraph D such that the $D - \{u, v\} \in \mathcal{D}_\alpha$. Thus D is not "exactly" a tournament. This is required in the inductive case for the proof of Lemma 3.3.

▶ **Lemma 3.10.** *Let D be an acyclic digraph. Let $u, v \in V(D)$ be such that $N^+(u) = N^-(v) = \emptyset$ and $D - \{u, v\}$ is a tournament. Then, a k-cut preserving set for (u, v) in D of size at most $k^3 + 5k^2 + 3k$ exists and is computable in polynomial time.*

Proof. If all (u, v)-vertex cuts in D have size at least $k+1$, then the correctness follows from Lemma 3.9. Thus, for the rest of the proof assume that there is a (u, v)-vertex-cut in D of size at most k. Let $C = \{c_1, \ldots, c_\ell\}$ be a minimal (u, v)-vertex cut in D of size $\ell \leq k$.

▷ Claim 3.11. $C \subseteq N_D^+(u) \cup N_D^-(v)$.

Proof. Suppose not. Then, there exists $c_i \in C$ such that $c_i \notin N^+(u) \cup N^-(v)$. Since C is a minimal (u, v)-vertex cut in D, there exists a path, say P, from u to v in $D - (C \setminus \{c_i\})$. Let u' be the first vertex on P after u and v' be the last vertex of P before v. Since $D - \{u, v\}$ is an acyclic tournament, $(u', v') \in E(D)$. Since $u', v' \notin C$, we get a (u, v)-path in $D - C$, contradicting that C is a (u, v)-vertex cut in D. ◁

Let $I = \{i \in [\ell] \mid c_i \in N_D^-(v)\}$ and $J = \{j \in [\ell] \mid c_j \in N_D^+(u)\}$. For all $i \in I$, let $U_i = ver_D(u, c_i; C)$ and $D_i = D[U_i]$. For all $j \in J$, let $V_j = ver_D(c_j, v; C)$ and $D_j = D[V_j]$. For all $(i, j) \in [\ell]^2$, $i \neq j$, let $Q_{i,j} = ver_D(c_i, c_j; \emptyset)$ and $D_{i,j} = D[Q_{i,j}]$.

For each $i \in I$ (resp. $j \in J$, resp. $(i, j) \in [\ell]^2$, $i \neq j$), we will compute a k-cut preserving set $\mathcal{Z}_i$ (resp. $\mathcal{Z}_j$, resp. $\mathcal{Z}_{i,j}$) of (u, c_i) (resp. (c_j, v), resp. (c_i, c_j)) in D_i (resp. D_j, resp. $D_{i,j}$) of size at most $2k + 3$ (resp. $2k + 3$, resp. $k + 3$). The procedure to do so is as follows.

- **Computing $\mathcal{Z}_i$, $i \in I$:** First observe that U_i is a candidate for $\mathcal{Z}_i$. Thus, if $|U_i| \leq 2(k+1)$, set $\mathcal{Z}_i = U_i$. Otherwise, we have that $|U_i| \geq 2k + 3$. Since $D - \{u, v\}$ is an acyclic tournament, let π be the unique topological ordering of $D - \{u, v\}$. We divide this case further into two cases.
  - **Case 1: $|N^+(u) \cap U_i| \leq k$:** Let $\widetilde{U_i}$ be the last $k + 1$ vertices of U_i in π. Observe that $\widetilde{U_i} \subseteq N^-(c_i) \cap U_i$. Define $\mathcal{Z}_i = (N^+(u) \cap U_i) \cup \widetilde{U_i} \cup \{u, c_i\}$. Clearly, $|\mathcal{Z}_i| \leq 2k + 3$. To prove that $\mathcal{Z}_i$ is a k-cut preserving set for (u, c_i) in D_i, consider some (u, c_i)-path P in D_i, such that $V(P) \not\subseteq \mathcal{Z}_i$. We will show a $\mathcal{Z}_i$-replacement witness for P in D_i. Consider the semi-$\mathcal{Z}_i$-based partition of P, $P = P_1 \uplus P_2$, where P_1 is the arc $(u, x) \in E(P)$, for some $x \in N^+(u) \cap U_i$ and P_2 is the (x, c_i)-subpath of P. Clearly, $V(P_1) \subseteq \mathcal{Z}_i$. We claim that there are $k + 1$ vertex-disjoint (x, c_i)-paths in $\mathcal{Z}_i$. To see this, consider the following argument. Since $V(P) \not\subseteq \mathcal{Z}_i$, there exists a vertex $y \in V(P)$ such that $y \notin \mathcal{Z}_i$. Then, $y \in V(P_2)$. Since $y \notin \mathcal{Z}_i$, it in particular holds that $y \notin \widetilde{U_i}$. Thus, all the vertices of $\widetilde{U_i}$ appear after y in π. Since there is a (x, y)-path in D_i, x appears before y in π. Thus, x appears before all the vertices of $\widetilde{U_i}$ in π. Thus, because $D - \{u, v\}$ is a tournament, $\widetilde{U_i} \subseteq N^+(x) \cap U_i$. Since $\widetilde{U_i} \subseteq N^-(c_i) \cap U_i$, there are $|\widetilde{U_i}|$ many vertex disjoint (x, c_i)-paths in $\mathcal{Z}$. This completes the proof.
  - **Case 2: $|N^+(u) \cap U_i| > k$:** First observe that all the vertices of $N^+(u) \cap U_i$ appear before c_i in π. Since π is a topological ordering of $D - \{u, v\}$, there are $|N^+(u) \cap U_i| > k$ vertex-disjoint (u, c_i)-paths in $\mathcal{Z}_i$. Thus, each (u, c_i)-vertex-cut in D_i has size at least

$k+1$. In this case, let $\mathcal{Z}_i$ be the k-cut preserving set for (u, c_i) in D_i obtained from Lemma 3.9. Observe that $|\mathcal{Z}_i| \leq k+3$.

- **Computing $\mathcal{Z}_j$, $j \in J$:** $\mathcal{Z}_j$ can be computed using arguments symmetric to the previous case.
- **Computing $\mathcal{Z}_{i,j}$, $(i,j) \in [\ell]^2$, $i \neq j$:** First observe that all the vertices of $Q_{i,j} \setminus \{c_i, c_j\}$ appear after c_i and before c_j in π. Thus, there are $|Q_{i,j} \setminus \{c_i, c_j\}|$ many vertex-disjoint (c_i, c_j)-paths in $D_{i,j}$. If $|Q_{i,j}| \leq k-2$, then set $\mathcal{Z}_i = Q_{i,j}$, otherwise let $\mathcal{Z}_i$ be the k-cut preserving set for (c_i, c_j) in $D_{i,j}$ obtained from Lemma 3.9. In either case, $|\mathcal{Z}_i| \leq k+3$.

Let $\mathcal{Z} := \bigcup_{i \in I} \mathcal{Z}_i \cup \bigcup_{j \in J} \mathcal{Z}_j \cup \bigcup_{(i,j) \in [\ell]^2, i \neq j} \mathcal{Z}_{i,j}$. Observe that $C \subseteq \mathcal{Z}$. First note that $|\mathcal{Z}| \leq |I|(2k+3) + |J|(2k+3) + \ell^2(k+3) \leq k^3 + 5k^2 + 3k^2$ (the last inequality holds because $|I| + |J| = \ell$ and $\ell \leq k$). We will now show that $\mathcal{Z}$ is a k-cut preserving set for (u, v) in D. To see this, consider some(u, v)-path P, in D. Since C is a (u, v)-vertex-cut in D there exists a vertex of C on P. Let c_i be the first vertex of C on P and c_j be the last vertex of C on P (c_i could be the same as c_j). Let P_1 be the (u, c_i)-subpath of P, P_2 be the (c_i, c_j)-subpath of P and P_3 be the (c_j, v)-subpath of P (if c_i is the same as c_j, then P_2 is empty). Thus, $P = P_1 \circ P_2 \circ P_3$ is a semi-$\mathcal{Z}$-based partition of P (as $C \subseteq \mathcal{Z}$). Since $\mathcal{Z}_i$ is a k-cut preserving set for (u, c_i) in D_i, $\mathcal{Z}_i$ is a k-cut preserving set for (c_j, v) in D_j and $\mathcal{Z}_{i,j}$ is a k-cut preserving set for (c_i, c_j) in $D_{i,j}$, and $\mathcal{Z}_i, \mathcal{Z}_j, \mathcal{Z}_{i,j} \subseteq \mathcal{Z}$, from Observation 3.4, $\mathcal{Z}$ is a k-cut preserving set for (u, v) in D. ◀

3.3 Finding a small k-cut preserving set for a pair in a $D \in \mathcal{D}_\alpha$

We are now ready to prove Lemma 3.3.

▶ **Lemma 3.3** (k-Cut Preserving Lemma). *Let D be an acyclic digraph, and $u, v \in V(D)$ be such that $N^-(u) = N^+(v) = \emptyset$. Additionally, let $D - \{u, v\} \in \mathcal{D}_\alpha$. Then there exists a k-cut preserving set for (u, v) in D of size at most $f(\alpha)$, where $f(1) = k^3 + 5k^2 + 3k$ and for $\alpha > 1$, $f(\alpha) = k^2 g(\alpha) + 2kh(\alpha)$, $g(\alpha) = (2k + (k + kf(\alpha-1))^2) f(\alpha-1)$ and $h(\alpha) = (k^2+k)g(\alpha) + kf(\alpha-1)$. Moreover, such a set can be found in time $n^{\mathcal{O}(1)}$, where $n = |V(D)|$.*

Proof. We prove this lemma using induction on α. When $\alpha = 1$, the proof follows from Lemma 3.10.

▷ Claim 3.12. Let $x, y \in V(D) \setminus \{x, y\}$. Then, a k-cut preserving set for (x, y) of size $g(\alpha)$ in any digraph D' that is a subgraph of D where $u, v \notin V(D')$, can be found in polynomial time.

Proof. Let W be a minimum (x, y)-vertex-cut in D'. If $|W| > k$, then the claim follows from Lemma 3.9. Thus, we are now in the case where $|W| \leq k$. For each $w \in W$, let $\mathcal{Z}(x, w)$ (resp. $\mathcal{Z}(w, y)$) be a k-cut preserving set for (x, w) (resp. (w, y)) in $D'[ver_{D'}(x, w; W)]$ (resp. $D'[ver_{D'}(w, y; W)]$). Let $B = N^-_{D'}(y) \cap \bigcup_{w \in W} \mathcal{Z}(w, y)$. For each $(\mathtt{a}, \mathtt{b}) \in (W \cup B)^2$, let $\mathcal{Z}(\mathtt{a}, \mathtt{b})$ be a k-cut preserving set for $(\mathtt{a}, \mathtt{b})$ in $D'[ver'_D(\mathtt{a}, \mathtt{b}; W \cup N^-(y))]$. Then, from Lemma 3.6, $\mathcal{Z}(x, y) := \bigcup_{w \in W} (\mathcal{Z}(x, w) \cup \mathcal{Z}(w, y)) \cup \bigcup_{(\mathtt{a},\mathtt{b}) \in (W \cup B)^2} \mathcal{Z}(\mathtt{a}, \mathtt{b})$ is a k-cut preserving set for (x, y) in D'.

We will now show that for any $w \in W$ and $(\mathtt{a}, \mathtt{b}) \in (W \cup B)^2$, each digraph among $D'[ver_{D'}(x, w; W)]$, $D'[ver_{D'}(w, y; W)]$ and $D'[ver_{D'}(\mathtt{a}, \mathtt{b}; W \cup N^-_{D'}(y))]$ has independence number strictly smaller than α. Then, from induction hypothesis and the expression for $\mathcal{Z}(x, y)$ written above, we will conclude that a k-cut preserving set for (x, y) in D' of size $g(\alpha)$ can be found in polynomial time. To see that the independence number of $D'[ver_{D'}(x, w; W)]$

is strictly less than α, observe that y is not adjacent to any vertex in $ver_{D'}(x, w; W)$, as W is an (x, y)-vertex cut in D'. Thus, any independent set of $D'[ver_{D'}(x, w; W)]$ together with y is an independent set of D' and hence of D. Since $y \notin \{u, v\}$, $u, v \notin V(D')$ and the independence number of $D - \{u, v\}$ is α, we have that the independence number of $D'[ver_{D'}(x, w; W)]$ is strictly smaller than α. A similar argument holds for $D'[ver_{D'}(w, y; W)]$ as in this case x is not adjacent to any vertex of $ver_{D'}(w, y; W)$. For $D'[ver_{D'}(\mathtt{a}, \mathtt{b}; W \cup N^-_{D'}(y))]$, since $ver_{D'}(\mathtt{a}, \mathtt{b}; W \cup N^-_{D'}(y)) \cap N^-_{D'}(y) = \emptyset$, $u, v \notin V(D')$ and $N^+_{D'}(y) = \emptyset$, any independent set of $D'[ver_{D'}(\mathtt{a}, \mathtt{b}; W \cup N^-_{D'}(y))]$ together with y is an independent set in $D - \{x, y\}$. Since $D - \{x, y\}$ has independence number α, $D'[ver_{D'}(\mathtt{a}, \mathtt{b}; W \cup N^-_{D'}(y))]$ has independence number strictly smaller than α. ◁

Let C be a minimum (u, v)-vertex-cut in D. If $|C| > k$, then the lemma follows from Lemma 3.9. Thus, for the remainder of the proof we assume that $|C| \leq k$. For each $c \in C$, let $U_c = ver_D(u, c; C)$, $V_c = ver_D(c, v; C)$, $\mathcal{Z}(u, c)$ be a (u, c) k-cut preserving set in $D[U_c]$, and $\mathcal{Z}(c, v)$ be a (c, v) k-cut preserving set in $D[V_c]$. For each $(c, c') \in C^2$, $c \neq c'$, let $Q_{c,c'} = ver_D(c, c'; C)$, and $\mathcal{Z}(c, c')$ be a k-cut preserving set in $D[Q_{c,c'}]$. Then from Lemma 3.5, $\mathcal{Z} := \bigcup_{c \in C} \mathcal{Z}(u, c) \cup \mathcal{Z}(c, v) \cup \bigcup_{(c,c') \in C^2, c \neq c'} \mathcal{Z}(c, c')$ is a k-cut preserving set for (u, v) in D. Since $C \cap \{u, v\} = \emptyset$, from Claim 3.12, for each $(c, c') \in C^2$, $c \neq c'$, $\mathcal{Z}(c, c')$ of size $g(\alpha)$ can be computed in polynomial time. In the remainder of the proof, we will show how to compute $\mathcal{Z}(u, c)$ and $\mathcal{Z}(c, v)$, for any $c \in C$, of the desired size. We will only give the proof of construction of $\mathcal{Z}(u, c)$ as the proof for $\mathcal{Z}(c, v)$ is symmetrical.

▷ Claim 3.13. For any $c \in C$, $\mathcal{Z}(u, c)$ of size $h(\alpha)$ can be computed in polynomial time.

Proof. For ease of notation, let $\widehat{D} = D[U_c]$. Let A be a minimum (u, c)-vertex-cut in $\widehat{D}$. First note that $A \cap \{u, v\} = \emptyset$. If $|A| > k$, then the claim follows from Lemma 3.9. Thus, for the remainder of the proof, assume that $|A| \leq k$.

For each $a \in A$, let $\widehat{U_a} = ver_{\widehat{D}}(u, a; A)$, $\widehat{V_a} = ver_{\widehat{D}}(a, c; A)$, $\widehat{\mathcal{Z}}(u, a)$ be a (u, a) k-cut preserving set in $\widehat{D}[\widehat{U_a}]$ and $\widehat{\mathcal{Z}}(a, c)$ be a (a, c) k-cut preserving set in $\widehat{D}[\widehat{V_a}]$. For each $(a, a') \in A^2$, $a \neq a'$, let $R_{a,a'} = ver_{\widehat{D}}(a, a'; A)$ and $\widehat{\mathcal{Z}}(a, a')$ be a k-cut preserving set in $\widehat{D}[R_{a,a'}]$. Then from Lemma 3.5, $\mathcal{Z}(u, c) := \bigcup_{a \in A}(\widehat{\mathcal{Z}}(u, a) \cup \widehat{\mathcal{Z}}(a, c)) \cup \bigcup_{(a,a') \in A^2, a \neq a'} \widehat{\mathcal{Z}}(a, a')$ is a k-cut preserving set for (u, c) in D. Since $A \cap \{u, v\} = \emptyset$ and $c \in \{u, v\}$, from Claim 3.12, for each $a \in A$, $(a, a') \in A^2$, $a \neq a'$, $\widehat{\mathcal{Z}}(a, c)$ and $\widehat{\mathcal{Z}}(a, a')$ of size $g(\alpha)$ can be computed in polynomial time. Moreover, the independence number of $\widehat{D}[\widehat{U_a}] - \{u, a\}$ is strictly smaller than α because $c(\neq v)$ is not adjacent to any vertex in $\widehat{U_a}$, besides possibly u and a. Thus, for each $a \in A$, a set $\widehat{\mathcal{Z}}(u, a)$ of size $f(\alpha - 1)$ can be computed in polynomial time by the induction hypothesis. This finishes the proof of the claim. ◁

Thus, from the previous arguments and Claim 3.13, we have that $\mathcal{Z}$ is a k-cut preserving set for (u, v) in D of size at most $k^2 g(\alpha) + 2kh(\alpha)$. ◀

A rough computation gives that, for any k, $g(\alpha) \leq 6k^2 f(\alpha - 1)$ and $h(\alpha) \leq 8k^4 f(\alpha - 1)$. This imply that $f(\alpha) \leq 22k^5 f(\alpha - 1)^3$. By noting that $f(1) \leq 22k^5$, we can show the following observation.

▶ **Observation 3.14.** *For any α and k, there exists a k-cut preserving set of size smaller than $f(k, \alpha) = (22k^5)^{4^\alpha}$.*

3.4 k-Cut Preserving Sets for a Set of Vertices

Below we also define a notion of k-cut preserving sets for a set of vertices. Such a notion will come handy in our applications. Given a digraph D and $X \subseteq V(D)$, for each $(u,v) \in X^2$, we define the digraph $D^X_{(u,v)}$ as follows (note that u could be equal to v). Let $R = V(D) - X$. Then, $D^X_{(u,v)}$ is the supergraph of $D[R]$ obtained by adding two new vertices u^+ and v^- together with the following set of additional arcs: $\{(u^+, x) : x \in R, (u,x) \in E(D)\} \cup \{(x, v^-) : x \in R, (x,v) \in E(D)\}$.

▶ **Definition 3.15** (k-Cut Preserving Set for a Set of Vertices). *For any digraph D, a positive integer k and $X \subseteq V(D)$, we say that $X \subseteq \mathcal{Z} \subseteq V(D)$ is a* k-cut preserving set for X, *if for all $(u,v) \in X^2$, $\mathcal{Z}$ is a k-cut preserving set for (u,v) in $D^X_{(u,v)}$.*

▶ **Lemma 3.16.** *For any digraph $D \in \mathcal{D}_\alpha$, a positive integer k, and $S \subseteq V(D)$ such that $D - S$ is a acyclic, a k-cut preserving set for S of size at most $|S|^2 f(k,\alpha)$ can be found in polynomial time, where $f(k,\alpha) \leq (22k^5)^{4^\alpha}$.*

Proof. For each pair $(u,v) \in S^2$ (u and v could be equal), let $\mathcal{Z}_{(u,v)}$ be the a k-cut preserving set for (u^+, v^-) in $D^S_{(u,v)}$ obtained from Lemma 3.3. From the definition of k-cut preserving set for S, $\mathcal{Z} = \bigcup_{(u,v)\in S^2} \mathcal{Z}_{(u,v)}$ is a k-cut preserving set for S. From Observation 3.14, for any $(u,v) \in S^2$, $|\mathcal{Z}_{(u,v)}| \leq f(k,\alpha)$. Thus, we conclude the correctness of the lemma. ◀

4 Fault-Tolerant (S,S)-Reachability

In this section, we prove Lemma 1.1. Recall that (D, S, ℓ, k) is an instance of FTR(S,S) where $D \in \mathcal{D}_\alpha$, $S \subseteq V(D)$ and ℓ, k are positive integers such that each strongly connected component of $D - S$ has size at most ℓ. The goal is to compute a subgraph H of D of size $k^{2^{\mathcal{O}(\alpha)}}$ such that, for any $A \subseteq E(D)$ of size at most k, for any $s,t \in S$, if $D - A$ has an (s,t)-path, then so does $H - A$. It is not difficult to see from Lemma 3.2 that if $\mathcal{Z}$ is a k-cut preserving set for S in D, then $H = D[\mathcal{Z}]$ is a solution for (D, S, ℓ, k) (for any ℓ). When $\ell = 1$, $D - S$ is acyclic and hence a k-cut preserving set for S can be computed using Lemma 3.16. When $\ell > 1$, in order to use Lemma 3.16 we modify the digraph D to turn $D - S$ acyclic. We now describe the operation, which we call dagify, that is used to turn $D - S$ acyclic. Informally, for each strongly connected component SC of D we turn it into an independent set while preserving the paths in D that use the vertices of SC. This is achieved by creating a new vertex for every ordered pair of vertices (say, (u,v)) in SC. Such a vertex represents the existence of a (u,v)-path in the strongly connected component SC. In fact, in the path in the modified graph, each new vertex corresponding to some pair (u,v) can be replaced by some (u,v)-path from the strongly connected component SC to yield a path in the original graph. Then, arcs between two vertices in this newly constructed vertex set are put in such a way that the concatenation of the paths corresponding to these new vertices gives a path in D. This idea is formalized below.

▶ **Definition 4.1** (dagify(D,R)). *Let D be a digraph, $R \subseteq V(D)$ and $S = V(D) \setminus R$. Let $SC_1, \ldots, SC_d$ be the strongly connected components of $D[R]$. For $a \in [d]$, let $V(SC_a) = \{v^a_1, \ldots, v^a_{n_a}\}$, where $n_a = |V(SC_a)|$. Then, $D^\dagger_R :=$ dagify(D,R) is the digraph defined as:*

Vertex set of $D^\dagger_R$: *For each $a \in [d]$, let $SC^\dagger_a = \{\mathbf{v}^a_{ij} \mid (v^a_i, v^a_j) \in \{SC_a\}^2, i,j \in [n_a]\}$. Let $R^\dagger = \cup_{a\in[d]} SC^\dagger_a$ and $V(D^\dagger_R) = R^\dagger \cup S$.*

Arc set of $D^\dagger_R$: *It contains all the arcs of D with both end-points in S. For each $a \in [d]$, ${SC^\dagger}_a$ is an independent set in $D^\dagger_R$. For any $a \in [d]$, $s \in S$ and $i,j \in [n_a]$, $(s, \mathbf{v}^a_{ij}) \in E(D^\dagger_R)$ if and only if $(s, v^a_i) \in E(D)$. Similarly, $(\mathbf{v}^a_{ij}, s) \in E(D^\dagger_R)$ if and only if $(v^a_j, s) \in E(D)$. We put the arcs between $SC^\dagger_a$ and $SC^\dagger_b$, for distinct $a,b \in [d]$ as follows. For any $i,j \in [n_a]$ and $i',j' \in [n_b]$, $(\mathbf{v}^a_{ij}, \mathbf{v}^b_{i'j'}) \in E(D^\dagger_R)$ if and only if $(v^a_j, v^b_{i'}) \in E(D)$.*

For a set of vertices of $X^\dagger \subseteq D_R^\dagger$, $full\text{-}comp(X^\dagger)$ denotes the set of vertices of $V(D)$ such that, for each $\mathbf{v}_{i,j}^a \in X^\dagger$, all the vertices of SC_a belong to full-comp$(X^\dagger)$. Also all the vertices of S that belong to $X^\dagger$, belong to $full\text{-}comp(X^\dagger)$. Observe that $|full\text{-}comp(X^\dagger)| \leq \ell^2 \cdot |X^\dagger|$, where ℓ is the upper bound on the size of each SC_a. Note from the construction above that, for any $s,t \in S$ and an (s,t)-path $P^\dagger$ in $D_R^\dagger$, there exists an (s,t)-path P in D such that $V(P) \subseteq full\text{-}comp(P^\dagger)$. The following observations state a few properties of the digraph $D_R^\dagger$ that would be useful when we want to find a k-cut preserving set for $D_R^\dagger$ using Lemma 3.16.

▶ **Observation 4.2.** *$D_R^\dagger[R^\dagger]$ is acyclic.*

Proof. Recall, from the construction of $D_R^\dagger$, that $R^\dagger = \bigcup_{a\in[d]} SC_a^\dagger$ and each $SC_a^\dagger$ is an independent set in $D_R^\dagger$. Without loss of generality, let $SC_1, \ldots, SC_d$ be the strongly connected components of $D[R]$ ordered as in their topological ordering. Then, there is no arc from a vertex of SC_b to a vertex of SC_a, for any $b > a$, in D. Thus, from the construction of $D_R^\dagger$, there is no arc from any $\mathbf{v}_{ij}^b$ to any $\mathbf{v}_{i'j'}^a$ $(b > a)$. This shows that $D_R^\dagger[R^\dagger]$ is acyclic. ◀

▶ **Observation 4.3.** *If $D \in \mathcal{D}_\alpha$ and every strongly connected component of $D[R]$ has size at most ℓ, then $D_R^\dagger \in \mathcal{D}_{\ell^2\alpha}$.*

Proof. Recall that $R^\dagger = \bigcup_{a\in[d]} {SC^\dagger}_a$ and $D_R^\dagger[SC_a^\dagger]$ has no arc. From the construction of $D_R^\dagger$, for each $a \in [d]$, $|SC_a^\dagger| \leq \ell^2$. Finally, since $D \in \mathcal{D}_\alpha$, from the construction of $D_R^\dagger$, the size of any maximum independent set in $D_R^\dagger$ is at most $\max_{a\in[d]} |SC_a^\dagger| \cdot \alpha \leq \ell^2\alpha$. ◀

We define some terminology that would come handy later. For any $A \subseteq E(D)$, we say that a vertex $v \in V(D)$ *is affected by* A if there exists some arc of A that is incident on v. The *set affected by A in $D_R^\dagger$* is the set of vertices of $D_R^\dagger$ containing the union of the vertices in $SC_a^\dagger$, for each $a \in [d]$ such that a vertex in SC_a is affected by A in D.

▶ **Observation 4.4.** *Let D be a digraph, $R \subseteq V(D)$ and $S = V(D) \setminus R$. Let $A \subseteq E(D)$ of size at most k. Let $\mathcal{A}^\dagger$ be the set affected by A in $D_R^\dagger$. Recall the construction of $D_R^\dagger$ from Definition 4.1. For some $\mathbf{v}_{ij}^a, \mathbf{v}_{i'j'}^b \in R^\dagger$, let $P^\dagger$ be an $\mathcal{A}^\dagger$-free $(\mathbf{v}_{ij}^a, \mathbf{v}_{i'j'}^b)$-path in $D_R^\dagger$. Then there exists a $(v_i^a, v_{j'}^b)$-path P in D such that: $V(P) \subseteq full\text{-}comp(P^\dagger)$ and, P does not use any arc of A.*

Proof. Recall the construction of dagify(D,R). Consider any path P obtained from $P^\dagger$ by replacing all the vertices of $R^\dagger$ as follows. If for any $c \in [d]$, $i^*, j^* \in [n_c]$, $\mathbf{v}_{i^*j^*}^c \in V(P^\dagger)$, then replace $\mathbf{v}_{i^*j^*}^c$ in $P^\dagger$ by any $(v_{i^*}^c, v_{j^*}^c)$-path in the strongly connected component SC_c. Clearly, the path P obtained is a $(v_i^a, v_{j'}^b)$-path in D and $V(P) \subseteq full\text{-}comp(P^\dagger)$. Also from the definition of $\mathcal{A}^\dagger$ and the fact that $P^\dagger$ is $\mathcal{A}^\dagger$-free, we get that P cannot use an arc of A. ◀

From the construction in Definition 4.1, for any $s,t \in S$, for an (s,t)-path P in D, we can associate a unique (s,t)-path $P^\dagger$ in $D_R^\dagger$. This is elaborated below. Consider the digraph $D_R^\dagger$ obtained by dagify(D,R). $(v_i^a, v_j^a) \in SC_a^2$ for some component SC_a of $D[R]$. Let $s,t \in S$. Let P be an S-free (s,t)-path in D. For any such path P, we define the notion of a *reduced path* of P in $D_R^\dagger$ as follows. Consider the unique partition $P = P_s \circ P_{i_1} \circ \ldots \circ P_{i_q} \circ P_t$ such that P_s is an arc (s,u) where $u \in V(SC_{i_1})$, P_t is an arc (v,t) where $v \in V(SC_{i_q})$ and for each $j \in [q]$, $V(P_{i_j}) \subseteq V(SC_{i_j})$, where $i_1, \ldots, i_j \in [d]$ and $i_1 < \ldots < i_q$. For each $j \in [q]$, let P_{i_j} be a $(v_{p_j}^{i_j}, v_{r_j}^{i_j})$-path. Consider the vertex $\mathbf{v}_{p_j,r_j}^{i_j}$ in $V_{i_j} \subseteq R^\dagger \subseteq V(D_R^\dagger)$. From the construction of $D_R^\dagger$, we get the (s,t)-path $P^\dagger = s \circ \mathbf{v}_{p_1,r_1}^{i_1} \circ \mathbf{v}_{p_2,r_2}^{i_2} \circ \ldots \circ \mathbf{v}_{p_q,r_q}^{i_q} \circ t$ in $D_R^\dagger$. This (s,t)-path $P^\dagger$ in $D_R^\dagger$ is called the *reduced path of P in $D_R^\dagger$*.

Proof of Lemma 1.1. Recall (D, S, ℓ, k) is an instance of FTR(S, S). Let $R = V(D) \setminus S$. Let $D^\dagger_R$ be obtained by dagify(D, R). From Observations 4.2 and 4.3, Lemma 3.16 can be used to compute a $(2k\ell^2 + 1)$-cut preserving set for S in $D^\dagger_R$. Let $\mathcal{Z}^\dagger$ be such a set. Let $\mathcal{Z} = \mathit{full\text{-}comp}(\mathcal{Z}^\dagger)$. We claim that $H = D[\mathcal{Z}]$ is a solution to the instance (D, S, ℓ, k). (First note that the size bound on H follows from Lemma 3.16 and the fact that each strongly connected component of R has size at most ℓ.)

Towards this let $A \subseteq E(D)$ of size at most k, $s, t \in S$ and P be an (s, t)-path in $D - A$. We need to show that there is some (s, t)-path in $H - A$ too. Let $P = P_1 \circ \ldots \circ P_q$ be the S-based partition of P such that each P_i is an (s_i, t_i)-path. Then it suffices to show that for each fixed $i \in [q]$, there is some (s_i, t_i) path in $H - A$ (these paths would yield a closed walk from s to t in $H - A$ and hence an (s, t)-path in $H - A$). In the remaining part of the proof, we focus on proving this. Note that each P_i is S-free. Fix any $i \in [q]$. For the ease of notation, let us call the path P_i as P, vertices s_i, t_i as s, t respectively.

Let $P^\dagger$ be the reduced path corresponding of P in $D^\dagger_R$. Since $\mathcal{Z}^\dagger$ is a $(2k\ell^2 + 1)$-cut preserving set for $P^\dagger$ in $D^\dagger_R$, consider a $\mathcal{Z}^\dagger$-witnessing replacement $P^\dagger = P^\dagger_1 \circ \ldots \circ P^\dagger_r$. Recall the notation from the construction in Definition 4.1.

For an arbitrary $c \in [r]$, let $P^\dagger_c$ be a $(\mathbf{v}^a_{ij}, \mathbf{v}^b_{i',j'})$-path (or $(s, \mathbf{v}^a_{ij})$-path or $(\mathbf{v}^a_{ij}, s)$-path). Observe that, since $P^\dagger$ is the reduced path of P, to finish the proof of the lemma, it is enough to show a (v^a_i, v^b_j)-path (or (s, v^a_i)-path or (v^a_i, s)-path) exists in $H - A$. Without loss of generality, let $P^\dagger_c$ be a $(\mathbf{v}^a_{ij}, \mathbf{v}^b_{i',j'})$-path, the other cases hold due to similar arguments.

As $P^\dagger = P^\dagger_1 \circ \ldots \circ P^\dagger_d$ is a $\mathcal{Z}^\dagger$-witnessing replacement, one of the following cases arises.

1. $V(P^\dagger_c) \subseteq \mathcal{Z}^\dagger$. Since $P^\dagger$ is the reduced path of P, consider the (v^a_i, v^b_j)-subpath, say P'_c, of P. Then, $V(P'_c) \subseteq \mathit{full\text{-}comp}(P^\dagger_c) \subseteq \mathcal{Z}$ (because $V(P^\dagger_c) \subseteq \mathcal{Z}^\dagger$). Also since P does not have an arc in A, so does P'_c. Thus, by the construction of H, P'_c is a path in $H - A$.
2. There is a list $\mathcal{L}_i$ of $2k\ell^2 + 1$ internally vertex-disjoint $(\mathbf{v}^a_{ij}, \mathbf{v}^b_{i'j'})$-paths in $D^\dagger_R[\mathcal{Z}^\dagger]$. Let $\mathcal{A}^\dagger$ be the set of affected vertices of A in $D^\dagger_R$. Clearly, $|\mathcal{A}^\dagger| \leq 2k\ell^2$. Then there exists a path in $\mathcal{L}_i$ that is $\mathcal{A}^\dagger$-free. Then from Observation 4.4, there exists a (v^a_i, v^b_j)-path, say P'_c, such that $V(P'_C) \subseteq \mathit{full\text{-}comp}(P^\dagger_c) \subseteq \mathcal{Z}$ and, that does not use an arc of A. From the construction of H, P'_c is a path in $H - A$.

This finishes the proof of the lemma. ◀

5 Conclusion

In this paper, we presented a sparsification procedure for the class of acyclic digraphs (or more generally, "almost" acyclic) of bounded independence, to preserve the (both normal and parity-based) reachability from a given terminal set S to a given terminal set T under the failure of any set of at most k arcs. In particular, it outputs a digraph whose size is completely *independent* of n and polynomial in k, while even the simple classes of directed paths and tournaments admit no sparsifier whose output is a digraph of less than $n - 1$ arcs already when $k = 1$. Apart from being interesting on its own from the perspective of fault tolerance, we also showed that our sparsification procedure finds applications in Kernelization. Specifically, we proved that the classic Directed Feedback Arc Set problem as well as Directed Edge Odd Cycle Transversal (which, in sharp contract, is W[1]-hard on general digraphs) admit polynomial kernels on bounded independence number digraphs. In fact, for any $p \in \mathbb{N}$, we designed a polynomial kernel for hitting all cycles of length ℓ where (ℓ mod $p = 1$). Additionally, we derived complementary results that assert the NP-hardness of DEOCT on tournaments, as well as its admittance of a sub-exponential time parameterized algorithm on digraphs of bounded independence.

We conclude the paper with a few directions for further research. Our result, currently, holds when the input digraph D is "almost acyclic" and has bounded independence number. From the example of the tournament described in the introduction (the one that is obtained by taking a transitive tournament and reversing the arcs along the Hamiltonian path defined by its topological ordering), it seems that some notion of "almost acyclic" might be necessary to have fault tolerant subgraphs whose size avoid the dependence on n. On the other hand, it might be possible to ask for something weaker than bounded independence number. For example, forbidding the existence of an induced P_α, the directed path on α vertices.

Question 1. Does $\mathrm{FTR}(S, S)$ admit a subgraph of size independent of n on digraphs that are "almost acyclic" and have no induced P_α, for some fixed positive integer α?

It is not very difficult to observe that our results (Lemmas 1.1 and 1.2) also hold when the input graph is undirected and has bounded independence number. It would be interesting (because of the arguments discussed earlier) if one could obtain similar results when the input undirected graph has no induced P_α.

Question 2. Does $\mathrm{FTR}(S, S)$ admit a subgraph of size independent of n when the input graph is undirected and has no induced P_α, for some fixed positive integer α?

It would also be interesting to discover other (di)graph classes where the dependence on n of the size of the output subgraph can be sublinear, for example, $\log n$, for $\mathrm{FTR}(S, S)$ and also for other fault tolerant graph properties.

References

1. Open Problems in Parameterized Complexity. `http://fpt.wikidot.com/open-problems`. Accessed: 2019-02-15.
2. Nir Ailon, Moses Charikar, and Alantha Newman. Aggregating inconsistent information: Ranking and clustering. *J. ACM*, 55(5):23:1–23:27, 2008.
3. Noga Alon. Ranking Tournaments. *SIAM J. Discrete Math.*, 20(1):137–142, 2006.
4. Noga Alon, Daniel Lokshtanov, and Saket Saurabh. Fast FAST. In *Automata, Languages and Programming, 36th International Colloquium, ICALP 2009, Rhodes, Greece, July 5-12, 2009, Proceedings, Part I*, pages 49–58, 2009.
5. Alexandr Andoni, Anupam Gupta, and Robert Krauthgamer. Towards (1+ ε)-approximate flow sparsifiers. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 279–293. Society for Industrial and Applied Mathematics, 2014.
6. Jørgen Bang-Jensen and Carsten Thomassen. A Polynomial Algorithm for the 2-Path Problem for Semicomplete Digraphs. *SIAM J. Discrete Math.*, 5(3):366–376, 1992.
7. Surender Baswana, Keerti Choudhary, Moazzam Hussain, and Liam Roditty. Approximate single source fault tolerant shortest path. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1901–1915. SIAM, 2018.
8. Surender Baswana, Keerti Choudhary, and Liam Roditty. Fault tolerant reachability for directed graphs. In *International Symposium on Distributed Computing*, pages 528–543. Springer, 2015.
9. Surender Baswana, Keerti Choudhary, and Liam Roditty. Fault-Tolerant Subgraph for Single-Source Reachability: General and Optimal. *SIAM Journal on Computing*, 47(1):80–95, 2018.
10. Stéphane Bessy, Fedor V Fomin, Serge Gaspers, Christophe Paul, Anthony Perez, Saket Saurabh, and Stéphan Thomassé. Kernels for feedback arc set in tournaments. *Journal of Computer and System Sciences*, 77(6):1071–1078, 2011.
11. Davide Bilò, Fabrizio Grandoni, Luciano Gualà, Stefano Leucci, and Guido Proietti. Improved purely additive fault-tolerant spanners. In *Algorithms-ESA 2015*, pages 167–178. Springer, 2015.

12 Davide Bilò, Luciano Gualà, Stefano Leucci, and Guido Proietti. Fault-tolerant approximate shortest-path trees. In *European Symposium on Algorithms*, pages 137–148. Springer, 2014.

13 Diptarka Chakraborty and Debarati Das. Sparse Weight Tolerant Subgraph for Single Source Shortest Path. In *16th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

14 Pierre Charbit, Stéphan Thomassé, and Anders Yeo. The Minimum Feedback Arc Set Problem is NP-Hard for Tournaments. *Combinatorics, Probability & Computing*, 16(1):1–4, 2007.

15 Moses Charikar, Tom Leighton, Shi Li, and Ankur Moitra. Vertex sparsifiers and abstract rounding algorithms. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pages 265–274. IEEE, 2010.

16 Shiri Chechik. Fault-tolerant compact routing schemes for general graphs. In *International Colloquium on Automata, Languages, and Programming*, pages 101–112. Springer, 2011.

17 Shiri Chechik, Michael Langberg, David Peleg, and Liam Roditty. Fault tolerant spanners for general graphs. *SIAM Journal on Computing*, 39(7):3403–3423, 2010.

18 Shiri Chechik, Michael Langberg, David Peleg, and Liam Roditty. F-sensitivity distance oracles and routing schemes. *Algorithmica*, 63(4):861–882, 2012.

19 Jianer Chen, Yang Liu, Songjian Lu, Barry O'sullivan, and Igor Razgon. A fixed-parameter algorithm for the directed feedback vertex set problem. *Journal of the ACM (JACM)*, 55(5):21, 2008.

20 Rajesh Chitnis. *Directed Graphs: Fixed-Parameter Tractability & Beyond*. PhD thesis, University of Maryland, 2014.

21 Rajesh Chitnis and Mohammad Taghi Hajiaghayi. Shadowless solutions for fixed-parameter tractability of directed graphs. *Encyclopedia of Algorithms*, pages 1–5, 2008.

22 Julia Chuzhoy. On vertex sparsifiers with Steiner nodes. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 673–688. ACM, 2012.

23 Marek Cygan, Fedor V Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized algorithms*, volume 4. Springer, 2015.

24 Erik Demaine, Gregory Z Gutin, Dániel Marx, and Ulrike Stege. 07281 Open Problems–Structure Theory and FPT Algorithmcs for Graphs, Digraphs and Hypergraphs. In *Dagstuhl Seminar Proceedings*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2007.

25 Reinhard Diestel. Graph Theory, volume 173 of. *Graduate texts in mathematics*, page 7, 2012.

26 Michael Dinitz and Robert Krauthgamer. Fault-tolerant spanners: better and simpler. In *Proceedings of the 30th annual ACM SIGACT-SIGOPS symposium on Principles of distributed computing*, pages 169–178. ACM, 2011.

27 Michael Dom, Jiong Guo, Falk Hüffner, Rolf Niedermeier, and Anke Truß. Fixed-parameter tractability results for feedback set problems in tournaments. In *Italian Conference on Algorithms and Complexity*, pages 320–331. Springer, 2006.

28 Rodney G Downey and Michael R Fellows. *Fundamentals of parameterized complexity*, volume 4. Springer, 2013.

29 Matthias Englert, Anupam Gupta, Robert Krauthgamer, Harald Racke, Inbal Talgam-Cohen, and Kunal Talwar. Vertex sparsifiers: New results from old techniques. *SIAM Journal on Computing*, 43(4):1239–1262, 2014.

30 Uriel Feige. Faster FAST(Feedback Arc Set in Tournaments). *CoRR*, abs/0911.5094, 2009. `arXiv:0911.5094`.

31 Paola Festa, Panos M Pardalos, and Mauricio GC Resende. Feedback set problems. In *Handbook of combinatorial optimization*, pages 209–258. Springer, 1999.

32 Fedor V Fomin and Michał Pilipczuk. Subexponential parameterized algorithm for computing the cutwidth of a semi-complete digraph. In *European Symposium on Algorithms*, pages 505–516. Springer, 2013.

33 Alexandra Fradkin and Paul Seymour. Edge-disjoint paths in digraphs with bounded independence number. *Journal of Combinatorial Theory, Series B*, 110:19–46, 2015.

34 Georges Gardarin and Wesley W. Chu. Integrity of databases: A general lockout algorithm with deadlock avoidance. In *Proceedings of the IFIP Working Conference on Modelling in Data Base Management Systems, [Freudenstadt, Germany, 5-8 January 1976]*, pages 395–411. North-Holland Publishing Company, 1976.

35 Richard M Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Springer, 1972.

36 Marek Karpinski and Warren Schudy. Faster Algorithms for Feedback Arc Set Tournament, Kemeny Rank Aggregation and Betweenness Tournament. In *Algorithms and Computation - 21st International Symposium, ISAAC 2010, Jeju Island, Korea, December 15-17, 2010, Proceedings, Part I*, pages 3–14, 2010.

37 Claire Kenyon-Mathieu and Warren Schudy. How to rank with few errors. In *STOC*, volume 7, pages 95–103, 2007.

38 F Thomson Leighton and Ankur Moitra. Extensions and limits to vertex sparsification. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 47–56. ACM, 2010.

39 Daniel Lokshtanov, MS Ramanujan, and Saket Saurabh. When recursion is better than iteration: A linear-time algorithm for acyclicity with few error vertices. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1916–1933. SIAM, 2018.

40 Daniel Lokshtanov, MS Ramanujan, Saket Saurabh, and Meirav Zehavi. Parameterized complexity and approximability of directed odd cycle transversal. *arXiv preprint*, 2017. `arXiv:1704.04249`.

41 Dániel Marx. Some open problems in parameterized complexity y. `http://www.cs.bme.hu/dmarx/papers/marx-dagstuhl2017-open.pdf`, 2-17.

42 Dániel Marx. What's next? Future directions in parameterized complexity. In *The Multivariate Algorithmic Revolution and Beyond*, pages 469–496. Springer, 2012.

43 Pranabendu Misra, Saket Saurabh, Roohani Sharma, and Meirav Zehavi. Sub-Exponential Time Parameterized Algorithms for Graph Layout Problems on Digraphs with Bounded Independence Number. In *38th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

44 Merav Parter. Dual failure resilient BFS structure. In *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing*, pages 481–490. ACM, 2015.

45 Merav Parter. Vertex fault tolerant additive spanners. *Distributed Computing*, 30(5):357–372, 2017.

46 Merav Parter et al. Fault-tolerant logical network structures. *Bulletin of EATCS*, 1(118), 2016.

47 Merav Parter and David Peleg. Sparse fault-tolerant BFS trees. In *European Symposium on Algorithms*, pages 779–790. Springer, 2013.

48 Abraham Silberschatz, Greg Gagne, and Peter B Galvin. *Operating system concepts (9th edition)*. Wiley, 2012.

The Computational Cost of Asynchronous Neural Communication

Yael Hitron
Weizmann Institute of Science, Rehovot, Israel
yael.hitron@weizmann.ac.il

Merav Parter
Weizmann Institute of Science, Rehovot, Israel
merav.parter@weizmann.ac.il

Gur Perri
Weizmann Institute of Science, Rehovot, Israel
gur.perri@weizmann.ac.il

Abstract

Biological neural computation is inherently asynchronous due to large variations in neuronal spike timing and transmission delays. So-far, most theoretical work on neural networks assumes the *synchronous* setting where neurons fire simultaneously in discrete rounds. In this work we aim at understanding the barriers of asynchronous neural computation from an algorithmic perspective. We consider an extension of the widely studied model of synchronized spiking neurons [Maass, Neural Networks 97] to the asynchronous setting by taking into account edge and node delays.

- **Edge Delays:** We define an asynchronous model for spiking neurons in which the latency values (i.e., transmission delays) of non self-loop edges *vary* adversarially over time. This extends the recent work of [Hitron and Parter, ESA'19] in which the latency values are restricted to be fixed over time. Our first contribution is an impossibility result that implies that the assumption that self-loop edges have no delays (as assumed in Hitron and Parter) is indeed necessary. Interestingly, in real biological networks self-loop edges (a.k.a. autapse) are indeed free of delays, and the latter has been noted by neuroscientists to be crucial for network synchronization.

  To capture the computational challenges in this setting, we first consider the implementation of a single NOT gate. This simple function already captures the fundamental difficulties in the asynchronous setting. Our key technical results are space and time upper and lower bounds for the NOT function, our time bounds are *tight*. In the spirit of the distributed synchronizers [Awerbuch and Peleg, FOCS'90] and following [Hitron and Parter, ESA'19], we then provide a general synchronizer machinery. Our construction is very modular and it is based on efficient circuit implementation of threshold gates. The complexity of our scheme is measured by the overhead in the number of neurons and the computation time, both are shown to be polynomial in the largest latency value, and the largest incoming degree Δ of the original network.
- **Node Delays:** We introduce the study of asynchronous communication due to variations in the response rates of the neurons in the network. In real brain networks, the *round duration* varies between different neurons in the network. Our key result is a simulation methodology that allows one to transform the above mentioned synchronized solution under edge delays into a synchronized under node delays while incurring a small overhead w.r.t space and time.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

Keywords and phrases asynchronous communication, asynchronous computation, spiking neurons, synchronizers

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.48

Acknowledgements We are thankful to Roei Tell and Gil Cohen for helpful discussions on Boolean Circuits. We also thank Yoram Moses for referring us to related work on asynchronous digital circuits and discussing the connections between the two settings.

1 Introduction

Understanding how the brain works, as a computational device, is a central challenge of modern neuroscience, artificial intelligence, and lately also in theoretical computer science and distributed computing [18, 19, 20, 17, 16, 32, 6, 39, 37]. This line of work usually assumes a simple synchronized model [23, 24] in which neurons fire simultaneously in discrete rounds in response to their neighboring neurons that fired in the previous round. This model, while being very convenient for algorithm design, does not take into account the inherent *asynchronous* nature of neural communication. In the neuroscience literature it has been noted that the asynchronous nature of these networks mostly stems from two independent sources [29]: *edge delays* (known as response latency[1].) [35, 5] and *node delays* (known as refractory period) [34, 3]. In this paper, we aim at understanding the computational cost incurred by such asynchronous communication. The overhead is measured by the overhead in the number of neurons and computation time required to compute a certain function. We believe that understanding the computational power, limitations and the connections between these models go beyond the setting of spiking neurons, and might also be relevant for the theory of digital logic design and circuit computation in general.

The Standard Synchronous Model [23, 24]. Before describing our asynchronous models, we first revise the standard synchronous model formally defined by Maass. In this model, the network evolves in discrete, synchronous *rounds* as a Markov chain where each neuron u in the network is a probabilistic threshold gate with a threshold (or bias) value $b(u)$. In every round t, the firing probability of neuron u only depends on the firing status of its incoming neighbors in the preceding round $t-1$. Formally, the potential $\text{pot}_t(u)$ of neuron u in round t is defined by the weighted sum over its incoming neighbors that fired in round $t-1$. The neuron u fires in round t with probability that depends on the quantity $\text{pot}_t(u) - b(u)$.

1.1 Asynchronous Computation with Bounded Edge Delays

We define an asynchronous model with edge delays bounded[2] by some given integer L. The dynamic of the network $\mathcal{N}$ is specified by a latency function $\ell : V \times V \times \mathbb{N} \to \mathbb{N}_{\leq L}$ interpreted as follows: For every neuron u firing in round τ, its spike reaches its outgoing neighbor v within $\ell(u, v, \tau)$ rounds where $\ell(u, v, \tau) \in [1, L]$ might be chosen adversarially for every $u \neq v$, and every round τ. The network solution $\mathcal{N}$ should then output the desired solution for *any* adversarial choice of the latency function ℓ. Setting $L = 1$ yields the standard synchronous model.

Asynchronous computation with edge delays was recently introduced by Hitron and Parter [12]. Their model is similar to ours only that in their model, the edge latencies are required to be *fixed* over time, whereas in our model the adversary is allowed to change it from round to round. The model of Hitron and Parter includes an additional restriction on the adversary (that sets the latency values) by requiring that self-spikes, i.e., of the form $\langle u, u, \tau \rangle$ have no latency and arrive within a single round to their destination. This assumption is justified in [12] by the experimental evidence that self-spikes in brain networks have almost no delays [14]. It is commonly believed in the neuroscience community that this no-delay property of self-edges is in fact essential for network synchronization [33, 21, 40, 8].

[1] Throughout, we use the terms edge-delay and latency interchangeably.

[2] This bound is crucial as will be later implied by our lower bound results that depend on L. E.g., both the computation time and the size of the network in this model *must* depend on L.

In this work, we provide a theoretical support for this hypothesis by showing that without such an assumption, one cannot even implement a single AND gate in this model. This impossibility result holds already in a setting where $L = 2$, the edge latencies are fixed over time (as assumed in [12]), and the computation time and network size are allowed to be unbounded. For this reason, we will mostly consider in this paper *nice* latency functions in which all self-spikes have a latency value of one round.

1.2 Asynchronous Computation with Bounded Node Delays

We next turn to consider an alternative source for asynchronous communication due to variations in the response timing of the individual neurons in the network. In real brain networks, for every neuron u there is a predefined time interval between two consecutive spikes of u. The length of this interval, which we call *round*, varies considerably among different neurons in the network [34, 3]. This poses the challenge of creating a synchronized response at the network level. To account this behavior, we consider a model in which network's evolution proceeds in *seconds*. A second in this context is simply the smallest measurable time unit. For a given integer $T \geq 1$, the dynamics is specified by a node-delay function $t : V \to \mathbb{N}_{\leq T}$ interpreted as follows: the round duration of each neuron v consists of $t(v)$ seconds. Specifically, the i^{th} round of v is defined by the time interval $R_i(v) = [(i-1)t(v)+1, i \cdot t(v)]$ for every $i \geq 1$. The neuron u fires in the second $i \cdot t(v)$ (i.e., at the end of its i^{th} round) only if the total potential due to spikes arriving in the interval $R_i(v)$ is sufficiently large. The network solution $\mathcal{N}$ should then output the desired solution for *any* adversarial choice of the node-delay function t. The input parameter T sets a bound on the differences between the round duration over all neurons in the network. Setting $T = 1$ yields the standard synchronous model.

Observe that the edge and node delay models do not imply one another. In the edge latency model, even though the spikes arrive in adversarially chosen rounds, all neurons in round τ still depend only on the spikes arriving in round τ. Thus, the duration of a round for all the neurons in the network is the same: a single tick (or a second) of the global clock. In contrast, in the node delay model, the adversary selects the round duration for each neuron which has two physical interpretations. First, it determines the time duration over which the potential due to arriving spikes is accumulated. In addition, it also determines the time interval between two consecutive spikes by the given neuron. In one of our most technically involved results, we show a non-trivial reduction between the edge delay and the node delay models, provided that the input network satisfies certain properties.

Finally, we note that this model has several variations which are in fact supported by our simulation results (from edge delays to node delays). In particular, one can consider a more elaborated setting in which the duration of each round per node varies in an adversarial manner *over time*, i.e., the node-delay function in such a model is of the form $t : V \times \mathbb{N}_{\geq 1} \to \mathbb{N}_{\leq T}$ interpreted as follows: the i^{th} round duration on each neuron v consists of $t(v, i)$ seconds. For example, in such a model the i^{th} round of node u can consists of 2 seconds while its $(i+1)^{th}$ round might consist of 100 seconds. In another variation, both edge and node delays are combined and the dynamic is specified by an L-bounded edge latency function $\ell : V \times V \times \mathbb{N}_{\geq 1} \to \mathbb{N}_{\leq L}$ and a T-bounded node-delay function $t : V \times \mathbb{N}_{\geq 1} \to \mathbb{N}_{\leq T}$.

1.3 Synchronizers

In the spirit of Awerbuch and Peleg's synchronizers for distributed networks [2] and the recent work of [12], our primary goal with respect to upper bound results is to provide a general simulation methodology that takes any n-neuron network $\mathcal{N}$ that *solves* the problem

in the standard synchronized setting (i.e., in which all spikes arrive within a single round) and transforms it into an "analogous" network $\mathsf{Sync}(\mathcal{N})$ in the edge delay setting while incurring a small overhead in the number of neurons and the computation time (w.r.t the base network $\mathcal{N}$).

For the setting in which the edge latencies are fixed over time and bounded by some integer L, Hitron and Parter [12] showed such a simulation using their efficient constructions for neural timers and counters. Their synchronized network solution $\mathsf{Sync}(\mathcal{N})$ has $O(n + L \log L)$ neurons and $O(rL^3)$ rounds where r is the computation time of the base network $\mathcal{N}$. While being quite efficient in terms of space and time, the synchronizers of [12] are heavily built up on the strong assumption that the latency values of the edges are fixed over time. In this paper, we aimed at understanding the edge latency setting in its most general form, and ask:

> *Can one provide a general synchronization scheme in a setting that allows the latency of the network edges to vary in an adversarial manner in each round?*

A priori it is not so clear if one can compute even basic Boolean functions without assuming that the latency values are fixed. We answer this question in the affirmative by presenting a modular syncronization scheme using a different approach than that taken in [12]. The benefit of this approach is in its modularity. We start by understanding the implementation of a single NOT gate in this model in terms of upper and lower bounds on the space and the time of the computation. We then use this synchronized NOT solution as a building block in the final synchronized network solution. Specifically, the synchronized NOT gates are used to build synchronized circuits (and hence threshold gates) which in turn combined into a whole synchronized network solution. The space and time overheads incurred by our solution are polynomial in L (the bound on the latency) and Δ, the maximal incoming degree in the base network $\mathcal{N}$.

We next turn to consider synchronizers for the node delay model. Our approach is based on showing a *simulation* result that takes any synchronized solution $\mathsf{sync}_E(\mathcal{N}, L)$ obtained by the synchronizer in the L-bounded *edge* latency model, and transforms it into a synchronized solution $\mathsf{sync}_V(\mathcal{N}, T)$ that works in the T-bounded *node* delay model for $L = \Theta(T^2)$.

▶ Remark. It is noteworthy that in contrast to the *distributed* setting of Awerbuch and Peleg [2] where the network size does not depend on the latencies, in the neural setting it is not the case. As our lower bound constructions, both the computation time and the network size must depend (in fact, polynomially) on the largest edge latency. For this reason, for any practical purposes, the study of asynchronous communication, in general, must assume bounded delays.

1.4 Our Results

We study the cost and limitations of asynchronous neural computation in a biologically plausible yet simple model of spiking neural networks. Our main focus is in the edge latency model where the dynamic is specified by an L-bounded function $\ell : V \times V \times \mathbb{N} \to \mathbb{N}_{\leq L}$. The node delay model is concerned only towards the end of the paper (Appendix C), as it is handled via reduction to the edge delay setting. In the first part of the paper, we show several negative results for the L-bounded model. This includes an impossibility results for delay on self-loop edges, as well as size and time lower bound on an implementation of a NOT gate in this model. In the second part, we consider the construction of synchronizers in this generalized setting. We note that these constructions are self-contained and are technically different from [13].

Negative Results. We first show that without assuming a minimal latency value on the self-loop edges, one cannot compute $AND(x, y)$ given two Boolean inputs x and y, even when the edge latency values are fixed over time and the largest latency is $L = 2$.

▶ **Theorem 1** (Impossibility for Arbitrary Latency Functions). *There exists no network that computes $AND(x, y)$ in a setting that allows the adversary to pick latencies in $\{1, 2\}$ for* all *edges in the network.*

The proof goes by showing that for any given candidate network solution $\mathcal{N}$, there exists a bad latency function ℓ under which $\mathcal{N}$ fails to compute $AND(x, y)$. This holds even when the latencies are fixed over time. From that point on, we restrict attention to *nice* latency functions, that assign latency value 1 to the self-loop edges in the network.

▶ **Definition 2.** *A latency function is* nice *if $\ell(u, u, \tau) = 1$ for every $u \in V$ and round τ.*

Our key technical contributions are lower bounds on the network size and the computation time for computing the $NOT(x)$ function in the L-bounded setting. Informally speaking, the $NOT(x)$ function appears to be a "complete" function for the purpose of synchronization in this asynchronous model. Indeed, our $NOT(x)$ implementation captures most of the essence of the L-bounded model. To obtain the final synchronization scheme we mainly glue together synchronized NOT units. For this reason, we spend much attention into understanding the tightness of our constructions by providing nearly matching lower bound results.

▶ **Lemma 3** (Size and Time Lower Bounds for Async. Computation of $NOT(x)$). *Any network that computes $NOT(x)$ in the L-bounded asynchronous setting must use $\Omega(L)$ neurons and $\Omega(L^3)$ time (the time lower bound is tight).*

This should be compared with the size lower bound of $\Omega(\log L)$ shown by [13] for their simplified asynchronous setting.

Positive Results. Our end result is a synchronizer that given any network $\mathcal{N}$ in the standard synchronous setting and an integer L, computes a network $\mathsf{sync}_E(\mathcal{N}, L)$ that performs the "same" computation as $\mathcal{N}$ in the L-bounded edge delay setting.

▶ **Theorem 4** (Synchronizers for Edge Delays). *There exists a synchronizer that given a network $\mathcal{N}$ with n neurons, maximum in-degree Δ, and maximum edge latency L, constructs a network $\mathsf{sync}_E(\mathcal{N}, L)$ that has an "analogous" execution in the L-bounded edge-delay setting with a total number of $\widetilde{O}(L^4 \cdot \mathrm{poly}(\Delta) \cdot n)$ neurons and a time overhead[3] of $\widetilde{O}(L^5 \cdot \log \Delta)$.*

Although the construction is inspired by the work of Awerbuch and Peleg [2], the implementation is very different as our neurons, unlike processors in a distributed network, are memoryless. Thus, they cannot aggregate the incoming messages as in [2]. Our construction is also different than that of [13], as the latter crucially depends on having fixed latencies over time.

For the node delay model, in Appendix C we show that given a network $\mathcal{N}$ in the standard synchronous setting and an integer T, one can compute an analogous network $\mathsf{sync}_V(\mathcal{N}, T)$ in the node-delay model with bounded node delay T by taking the following approach. Apply the algorithm of Theorem 4 with $\mathcal{N}$ and $L = \Theta(T^2)$. This results with a network $\mathsf{sync}_E(\mathcal{N}, L)$

[3] The $\widetilde{O}$ hides a factor of $\mathrm{poly}(\log(n \cdot r))$, where r is the number of simulation rounds.

that performs the same computation as $\mathcal{N}$ in the L-bounded edge delay model. The desired network $\mathsf{sync}_V(\mathcal{N}, T)$ is then obtained by multiplying the edge weights of a carefully defined edge *subset* in $\mathsf{sync}_E(\mathcal{N}, L)$ by a factor of T. The quite delicate analysis then shows that the network $\mathsf{sync}_V(\mathcal{N}, T)$ indeed simulates the original network $\mathcal{N}$ upon any selection of the node-delay function $t : V \to \mathbb{N}_{\leq T}$. By setting $L = O(T^2)$ in Theorem 4, we show the following for the node delay model:

▶ **Theorem 5** (Synchronizers for Node Delays)**.** *There exists a synchronizer that given a network $\mathcal{N}$ with n neurons, maximum in-degree Δ, and maximum node delay T, constructs a network $\mathsf{sync}_V(\mathcal{N}, T)$ that has an "analogous" execution in the T-bounded node-delay setting with a total number of $\widetilde{O}(T^8 \cdot \mathrm{poly}(\Delta) \cdot n)$ neurons and a time overhead of $\widetilde{O}(T^{10} \cdot \log \Delta)$.*

We note that our preference to take a modular approach rather then an optimized one inevitably leads to suboptimal space and time bounds in both Theorems 4 and 5. For example, Theorem 5 is shown via a simulation result, which further deepens our understanding of the connections between these models. We believe that by employing a more direct approach for building synchronizers in the node-delay model, one should get a considerably improved dependency in the delay bound T.

1.5 Our Approach in a Nutshell

We next provide the high level ideas for the key contributions. Throughout, unless stated otherwise we consider the edge delay model where the dynamics is specified by an arbitrary latency function.

Size and Time Lower Bound for Computing $NOT(x)$. A network $\mathcal{N}$ with input neuron x and an output neuron z computes the function $NOT(x)$ in the asynchronous setting if the following holds: when $x = 0$, the output z fires in at *least* one round regardless of the latency function; and when $x = 1$ the output z never fires for any latency function. To show a size lower bound of $\Omega(L)$ we take the following approach. First, we reduce any network $\mathcal{N}$ that computes $NOT(x)$ into a simpler and yet not larger network $\mathcal{N}_{simple}$. In the latter network the only inhibitor is the input x which also has a self-loop of large positive weight, and outgoing edges of very large negative weight to all the excitatory neurons in $\mathcal{N}$. The second part of the proof shows a lower bound for $\mathcal{N}_{simple}$ using its specialized structure. We will assume towards a contradiction that the in-degree of each neuron in $\mathcal{N}_{simple}$ is less than L and exhibit two conflicting latency functions ℓ_0, ℓ_1 that satisfy the following. If $\mathcal{N}_{simple}$ computes $NOT(x)$ with ℓ_0 and with $x = 0$, then it must fail to compute $NOT(x)$ with the function ℓ_1 and $x = 1$. To compute these latency functions, we partition the simulation into blocks $T_0, \ldots,$ each containing L rounds. In each phase i, we set the latency values for all the edges and all the rounds in block T_i. Throughout, we will keep the invariant that there exists *no* neuron that fires when $x = 0$ and with ℓ_0, but does not fire when $x = 1$ and with ℓ_1. By the correctness of the network, the output neuron must fire at least once when $x = 0$, thus leading to the contradiction. In the very high level, the fact that the in-degree of each vertex is small is used in order to spread over the at most L incoming spikes of each neuron u in a balanced manner over the L rounds of the block. This will prevent the firing of a neuron z when $x = 0$. Our lower bound is complemented by an upper bound of $O(L^2)$ as described next.

The Generalized Synchronization Scheme. The scheme is based on gradual steps.

Step I: Synchronization of NOT and OR Gates. We start by considering the asynchronous computation of simple Boolean functions $NOT(x)$ and $OR(x_1, \ldots, x_\ell)$ with a small number of neurons. The key challenge is in implementing the NOT gate. When $x = 1$, the output gate is required not to fire (i.e., output 0) throughout the *entire* execution. In contrast, when $x = 0$ the output gate should fire at least once during the execution. The construction is combinatorial and uses a similar logic to the lower bound arguments. It contains a collection of $L + 1$ neurons with outgoing edges to the output z. The above mentioned lower bound result shows that the incoming degree of z must be at least $L - 2$.

Step II: Synchronization of a Boolean Circuit. Any Boolean circuit $\mathcal{A}$ can be implemented by NOT and OR gates. To simulate the computation of $\mathcal{A}$ in the asynchronous setting, we replace each gate g_i in $\mathcal{A}$ by its synchronized implementation $\mathsf{Sync}(g_i)$ constructed in Step (I). For a gate g_i in layer j with incoming gates $g_{i,1}, \ldots, g_{i,k}$, the input to the sub-network $\mathsf{Sync}(g_i)$ are the output neurons of the sub-networks $\mathsf{Sync}(g_{i,1}), \ldots, \mathsf{Sync}(g_{i,k})$. The synchronization between the layers of the circuit is governed by a directed chain of $O(dL^3)$ neurons. The head of the chain fires in the first round of the simulation and activates the network. The sub-networks $\mathsf{Sync}(g_i)$ of gates g_i in layer j are activated by the $\Theta(j \cdot L^3)$ neuron in this chain. These parameters are set so that we can be sure that the modules of layer j are activated, only *after* the spikes from the output neurons of the previous layer have reached the input of this layer. Overall the synchronized transformed network $\mathsf{Sync}(\mathcal{A})$ has $O(d \cdot L^3 + m \cdot L^2)$ neurons, where d is the depth of $\mathcal{A}$ and m is the number of gates. The overtime in the computation is $O(d \cdot L^4)$ rounds.

Step III: Synchronization of a Single (Probabilistic) Threshold Gate. To synchronize a single deterministic threshold gate, we use the fact that a threshold gate with incoming degree Δ can be implemented by a Boolean circuit with $\text{poly}(\Delta)$ neurons and depth $O(\log \Delta)$. This allows us to use the synchronized construction of the previous step. Turning to probabilistic threshold gates, here it is much less clear how to implement such a gate by a Boolean circuit. We take the following approach. First, we use the fact from [20] that a spiking neuron[4] u with bias $b(u)$ is equivalent to a *deterministic* neuron u' whose bias is sampled from the Logistic distribution with mean $b(u)$. Therefore our key challenge is in sampling a value from a given Logistic distribution. To do that, we use a collection of k (input-less) spiking neurons each fires independently with probability half. These neurons provide us the random bits for this process of sampling. In fact, these fair coins tosses allows one to sample a value *almost* uniformly at random in the range $[0, 2^k]$. We will then use the method of inverse transform sampling to convert this almost-uniform sampled value to a value that is sampled from the Logistic distribution up to a small error in the sampling. Using Taylor expansion of the natural log function, we implement this Uniform to Logistic transformation by a collection of simple arithmetic operations applied on a collection of Boolean neurons. The total error in our sampling is set to be small enough so that the output distribution of the Boolean circuit is almost indistinguishable from that of the probabilistic threshold gate.

Grand Finale: Synchronization of a Spiking Neural Network. Finally, given an SNN network $\mathcal{N}$ of (probabilistic) threshold gates the synchronized network $\mathsf{sync}(\mathcal{N})$ is obtained as follows. Each threshold gate g_i in $\mathcal{N}$ is replaced by its synchronized implementation $\mathsf{Sync}(g_i)$.

[4] The probabilistic threshold gates of SNN.

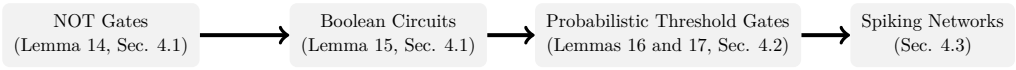


Figure 1 A road-map for synchronizing spiking neural networks.

The key challenge is in synchronizing these modules so that every neuron v in $\mathcal{N}$ (i.e., not only the output neuron) has an equivalent neuron v' in $\mathsf{sync}(\mathcal{N})$ that simulates v for any possible latency function throughout the entire simulation. See Fig. 1 for an illustration.

From Edge Delays to Node Delays. Given a network $\mathcal{N}$ to be simulated and an integer T, our goal is to build a synchronizer $\mathsf{sync}_V(\mathcal{N}, T)$ that *simulates* $\mathcal{N}$ in the T-bounded node-delay model. To do that, we first compute a network $\mathsf{sync}_E(\mathcal{N}, L)$ that simulates $\mathcal{N}$ in the L-bounded edge-delay model for $L = \Theta(T^2)$. Then the output network $\mathsf{sync}_V(\mathcal{N}, T)$ is obtained by dividing some of the edge weights in $\mathsf{sync}_E(\mathcal{N}, L)$ by a factor of T. The correctness argument is based on showing that for every node-delay function $t : V \to \mathbb{N}_{\leq T}$, there exists an edge-delay function $\ell : V \times V \times \mathbb{N}_{\geq 0} \to \mathbb{N}_{\leq L}$ such that the execution of the network $\mathsf{sync}_E(\mathcal{N}, L)$ with the edge-delay function ℓ (in the edge-delay model) is *similar* to the execution of the network $\mathsf{sync}_V(\mathcal{N}, T)$ with the node-delay function t (in the node-delay model). Since the network $\mathsf{sync}_E(\mathcal{N}, L)$ simulates the original network $\mathcal{N}$ for any edge-delay function, it will imply that the network $\mathsf{sync}_V(\mathcal{N}, T)$ simulates the original network $\mathcal{N}$ for any node-delay function as desired.

Additional Related Work. Asynchronized communication in spiking neural networks has been studied in several settings. Maass [22, 25] considered a quite elaborated model for deterministic neural networks with *arbitrary* response *functions* for the edges, and a vector firing times for all neuron. The approach of [22] mostly concerned the computational power of this model upon choosing the best parameters for the network. I.e., showing feasibility results for various functions. In contrast, in this work our goal is to *bound* the computation time and the network size under this asynchronous setting. Khun et al. [15] studied the asynchronous dynamics under the stochastic model of DeVille and Peskin [7].

Turning to the setting of logical circuits, there is a long line of work on the asynchronous setting under various model assumptions [1, 11, 36, 4] that do not quite fit the memory-less setting of spiking neurons. A more related work to our setting is by Martin, Manohar and Moses [28, 26, 27] who studied the computational power of asynchronous digital circuits. In particular, they characterize the necessary and sufficient conditions for a valid operation of a given circuit in the asynchronous setting. For example, they showed that if all edges and nodes suffer from an unbounded delay then the computational power of the circuit must be very limited. The focus in our work is quite different. Instead of studying the computational power of the asynchronous setting, we bound the computational overhead for solving concrete problems.

2 The Synchronous and Asynchronous SNN Models

A deterministic neuron u is modeled by a *deterministic* threshold gate. Letting $b(u)$ to be the threshold value of u, then u outputs 1 if the weighted sum of its incoming neighbors exceeds $b(u)$. A *spiking neuron* is modeled by a probabilistic threshold gate which fires with a sigmoidal probability that depends on the difference between its weighted incoming sum and $b(u)$.

Neural Network Definition. A *Neural Network* (NN) $\mathcal{N} = \langle X, Z, Y, w, b\rangle$ consists of n input neurons $X = \{x_1, \ldots, x_n\}$, m output neurons $Y = \{y_1, \ldots, y_m\}$, and k auxiliary neurons $Z = \{z_1, ..., z_k\}$. In spiking neural network (SNN), the neurons can be either deterministic threshold gates or probabilistic threshold gates. The directed weighted synaptic connections between $V = X \cup Z \cup Y$ are described by the weight function $w : V \times V \to \mathbb{R}$. A weight $w(u, v) = 0$ indicates that a connection is not present between neurons u and v. Finally, for any neuron v, the value $b(v) \in \mathbb{R}$ is the threshold value (activation bias). The in-degree of every input neuron x_i is zero, i.e., $w(u, x) = 0$ for all $u \in V$ and $x \in X$. Additionally, each neuron is either inhibitory or excitatory: if v is inhibitory, then $w(v, u) \leq 0$ and if v is excitatory, then $w(v, u) \geq 0$ for every u. This restriction arises from the biological structure of the neurons.

Network Dynamics in the Synchronous Setting. The network evolves in discrete, synchronous rounds as a Markov chain. The firing probability of every neuron in round τ depends on the firing status of its neighbors in round $\tau - 1$, via a standard sigmoid function, with details given below. For each neuron u, and each round $\tau \geq 0$, let $\sigma_\tau(u) = 1$ if u fires (i.e., generates a spike) in round τ. Let $\sigma_0(u)$ denote the initial firing state of the neuron. The firing state of each input neuron x_j in each round is the input to the network. For each non-input neuron u and every round $\tau \geq 1$, let $\text{pot}(u, \tau)$ denote the membrane potential at round τ and $p(u, \tau)$ denote the firing probability ($\Pr[\sigma_\tau(u) = 1]$), calculated as $\text{pot}(u, \tau) = \sum_{v \in V} w(v, u) \cdot \sigma_{\tau-1}(v) - b(u)$ and $p(u, \tau) = \frac{1}{1+e^{-\text{pot}(u,\tau)/\lambda}}$ where $\lambda > 0$ is a *temperature parameter* which determines the steepness of the sigmoid. Clearly, λ does not affect the computational power of the network, thus we set $\lambda = 1$.

2.1 Network Dynamics in the Edge Delay Setting

The dynamic of the network is governed by a latency function $\ell : V \times V \times \mathbb{N} \to \mathbb{N}$ interpreted as follows. For every directed edge $e = (u, v)$ and round τ, a spike generated by u in round τ arrived at v after $\ell(u, v, \tau)$ rounds. In the synchronous setting, $\ell(u, v, \tau) = 1$ for every u, v, τ.

For every neuron v and round τ, let $A(u, \tau) = \{(v, \tau') |\ v \in V,\ \tau' + \ell(v, u, \tau') = \tau\}$ denote all the spike events that if occur, arrive to u at round τ. The state of u in round τ is given by:

$$\text{pot}(u, \tau) = \sum_{(v,\tau') \in A(v,\tau)} w(v, u) \cdot \sigma_{\tau'}(v) \ - \ b(u) \ \text{ and } \ \sigma_\tau(u) = 1 \text{ iff } \text{pot}(v, \tau) \geq 0 \ . \qquad (1)$$

If u is a probabilistic threshold gate then it fires with probability $p(u, \tau) = \frac{1}{1+e^{-\text{pot}(u,\tau)}}$. When $\ell(u, v, \tau) = \ell(u, v, \tau')$ for every u, v and $\tau' \neq \tau$, we may omit τ and write $\ell(u, v)$.

▶ **Definition 6** (The L-bounded Edge-Delay Setting)**.** *Given is a network $\mathcal{N}$ and an integer L. It is assumed the network contains a special neuron, the* starter*, that fires in the first round of the simulation. The dynamic is determined by a latency function ℓ. This function ℓ can be chosen arbitrarily among all L-bounded nice functions.*

▶ **Definition 7** (Computation of a Boolean Function in the L-bounded Edge-Delay Setting)**.** *Let $f : \{0, 1\}^n \to \{0, 1\}^k$ be a Boolean function. A network $\mathcal{N}$ with n input neurons $x_1, \ldots, x_n$ and k output neurons $z_1, \ldots, z_k$* computes *f in this setting if for every nice L-bounded function ℓ and for every fixed possible assignment to the input neurons $b_1, \ldots, b_n$ the following holds: (i) If $f_i(b_1, \ldots, b_n) = 1$, then there exists a round in which z_i fires, where $f_i(\cdot)$ is the i^{th} bit in the output of f. (ii) If $f_i(b_1, \ldots, b_n) = 0$ then z_i does not fire throughout the entire execution. Furthermore, the network $\mathcal{N}$ computes the function f* in r rounds *if $\mathcal{N}$ computes f, and for every nice L-bounded function ℓ, input bits $b_1, \ldots, b_n$ and index i such that $f_i(b_1, \ldots, b_n) = 1$, z_i fires in some round $\tau \leq r$.*

Note that by this definition, for a network $\mathcal{N}$ that computes a Boolean function f within r rounds, to evaluate the output of the function it is sufficient to inspect the state of the output bits over the first r rounds of the network's simulation. Furthermore, as edge delays are allowed to be chosen in an adversarial manner, one cannot hope for having all output neurons to fire in the exact same around. One mechanism that we use to keep the output neurons fire simultaneously is by using self-loop edges whose latency values is fixed to be 1.

Synchronizers. A synchronizer ν is an algorithm that gets as input a network $\mathcal{N}$ and outputs a network $\mathcal{N}' = \mathsf{sync}(\mathcal{N})$ that contains all the neurons of $\mathcal{N}$, plus additional auxiliary neurons. One of the auxiliary neurons in $\mathcal{N}'$ is a *starter* neuron that fires in the *first* round of the simulation. The network $\mathcal{N}'$ works in the asynchronous setting and should have *similar execution* to $\mathcal{N}$ in the sense that for every neuron $v \in V(\mathcal{N})$, the firing pattern of v in the asynchronous network should be similar to the one in the synchronous network. The output network $\mathcal{N}'$ simulates each round of the network $\mathcal{N}$ by a *phase*.

▶ **Definition 8** (Phases). *We partition the execution of $\mathcal{N}'$ into phases $1, 2, \ldots$, using a function $r : V(\mathcal{N}) \times \mathbb{N} \to \mathbb{N}$ that defines the beginning of phase p, i.e. the p^{th} phase is the round interval $[r(v,p), r(v,p+1))$.*

▶ **Definition 9** (Similar Executions (Deterministic Networks)). *The* synchronous *execution Π of a deterministic network $\mathcal{N}$ is specified by a list of states $\Pi = \{\sigma_1, \ldots, \}$ where each σ_i is a binary vector describing the firing status of the neurons in round i. The* asynchronous *execution of the network $\mathcal{N}' = \mathsf{sync}_E(\mathcal{N}, L)$ with a latency function ℓ denoted by $\Pi'(\ell)$ is defined analogously only when applying the asynchronous dynamic of Eq. (1). The execution $\Pi'(\ell)$ is divided into phases according the a function $r : V(\mathcal{N}) \times \mathbb{N} \to \mathbb{N}$.*

*The network $\mathcal{N}$ and the pair $\langle \mathcal{N}', \ell \rangle$ have a **similar execution** if $V(\mathcal{N}) \subseteq V(\mathcal{N}')$, and in addition, a neuron $v \in V(\mathcal{N})$ fires in round p in the execution Π iff v fires during phase p in $\Pi'(\ell)$. The networks $\mathcal{N}$ and $\mathcal{N}'$ are **similar** if $\mathcal{N}$ and $\langle \mathcal{N}', \ell \rangle$ have a similar execution for every nice latency function ℓ.*

Note that specifically, if a synchronous network $\mathcal{N}$ computes a Boolean function f by round r and $\mathcal{N}$ and $\mathcal{N}'$ are similar, then $\mathcal{N}'$ computes f by phase r. Therefore, if we know that each phase is of at most q rounds, we get that $\mathcal{N}'$ computes f in $r \cdot q$ rounds.

Finally, we note that the extension for randomized networks with probabilistic gates is quite straightforward if one simply fixed the random coins used by the neurons over the simulation. That is, to be able to faithfully compare the simulation of two random networks, one has to fix the random coins of both of the simulations to be the same. For this reason, given an input randomized network $\mathcal{N}$ in the synchronized model, we maintain all the random coins generated by the neurons of the network over the simulation. These random coins are then fed to the network $\mathcal{N}'$ (i.e., obtained by applying our synchronizers). Since we compare two randomized networks that use the same set of random coins, we can treat these networks as deterministic. In Appendix C we provide the analogous definitions for the T-bounded node-delay model. Throughout the main paper, we consider only the edge-delay model and to avoid cumbersome notation that synchronized network solutions for this model are denoted by $\mathsf{sync}(\mathcal{N})$ (rather than $\mathsf{sync}_E(\mathcal{N}, L)$).

3 Negative Results

Impossibility Result for Arbitrary Latency Functions. We start by considering Theorem 1, and show that if the latency values are allowed to be set in an adversarial manner in $\{1, 2\}$, then there exists no network that computes the AND of two Boolean inputs. In Appendix A, we show:

▶ **Lemma 10.** *Given input neurons x, y and an output neuron z, there is no network computing $AND(x, y)$ under every latency function $\ell : V \times V \to \{1, 2\}$.*

In the high level, we show that one can set the latency values such that all the spikes that *depend* on the value of x (resp., y) arrive at odd (resp., even) rounds. Therefore, at any round, there is no neuron that fires as a function of *both* x and y.

Size and Time Lower Bound

In this section we show the proof for Lemma 3. Here we focus on the size lower bound although the high level proof strategy for the time lower bound is quite similar. The time lower bound is presented in Appendix A.1. Our proof strategy is as follows. First we reduce any network $\mathcal{N}$ that computes $NOT(x)$ in the asynchronous setting, to a network $\mathcal{N}_{simple}$ with a simpler structure that makes it easier to make arguments on it. The second part of the argument shows the lower bound for simple networks. All missing proofs of this section are in Appendix A.

▶ **Definition 11** (Strong Neurons and Simple Networks)**.** *A neuron u is* strong *in a given network if $w(u, u) \geq b(u)$, and otherwise it is* weak*. Note that specifically, an excitatory neuron u with $b(u) \leq 0$ is strong. Given a single input neuron x, we say that a network $\mathcal{N}$ is* simple *if the following hold: (i) x is a strong neuron and has an outgoing edge of infinite* negative *weights to all other neurons in the network; and (ii) all other neurons are excitatory.*

We note that the simple network is not a *legally defined* neural network: the input neuron has an incoming edge (self-loop), and it is an inhibitor with a positive self-loop. However, this network definition is only for the sake of the analysis and as such, it is not restricted to follow any rule.

Reduction to Simple Networks. Given a network $\mathcal{N}$ with an input neuron x, define $\mathcal{N}_{simple}$ as follows. Exclude all the inhibitory neurons from $\mathcal{N}_{simple}$ and take all edges between excitatory neurons to be as in $\mathcal{N}$. Then, add a self-loop of infinite weight to the input neuron x, and connect it to every neuron with infinite[5] negative weight.

▶ **Lemma 12.** *If $\mathcal{N}$ computes $NOT(x)$ within r rounds starting with the initial state $\bar{\sigma}$, then also $\mathcal{N}_{simple}$ computes it within r rounds, when starting with the initial states as in $\bar{\sigma}$ restricted to the vertices of $\mathcal{N}_{simple}$.*

The proof goes by claiming that for any latency function ℓ_{simple} for $\mathcal{N}_{simple}$, we can show the existence of a latency function ℓ for $\mathcal{N}$ whose performance is only *worse* than that of $\mathcal{N}_{simple}$ with ℓ_{simple}. That is, when $x = 0$ (resp., $x = 1$) then the potential of all neurons in $\mathcal{N}_{simple}, \ell_{simple}$ is not decreased (resp. increased) when compared to $\mathcal{N}, \ell$. Since the network $\mathcal{N}$ computes $NOT(x)$ with the latency function ℓ within r rounds, we conclude that also $\mathcal{N}_{simple}$ computes it with the latency function ℓ_{simple} within at most r rounds.

Fix a $NOT(x)$ network $\mathcal{N}$. For an integer r, a latency function ℓ for $\mathcal{N}$ is *r-good* with the initial configuration $\bar{\sigma}$, if the network computes $NOT(x)$ within r rounds. I.e., when $x = 0$, the output of $\mathcal{N}$ fires in some round $\tau \leq r$, and when $x = 1$ it never fires when all latencies are given based on ℓ. If ℓ is r-good for some integer r we say it is *good*, and otherwise the

[5] By infinite we mean large enough so that when the spike by x arrives at some neuron v, v would not fire.

latency function is *bad* (the network fails to compute $NOT(x)$). Note that in order for a network to compute $NOT(x)$ within r rounds, it is required that any latency function is r-good for a fixed initial configuration.

Lower Bound for Simple Networks. Assume towards contradiction that there exists a *simple* network $\mathcal{N} = \mathcal{N}_{simple}$ with maximum in-degree $\Delta_{in} < L - 2$ that computes $NOT(x)$. I.e., there exists an initial configuration $\bar{\sigma}$ for all neurons but x such that every latency function ℓ is good for $\langle \mathcal{N}, \bar{\sigma} \rangle$. In what follows, we define two *conflicting* latency functions ℓ_0 and ℓ_1, such that if ℓ_0 is good when the initial state of x is 0, then it implies that ℓ_1 is bad when the initial state of x is 1.

Defining the Latency Functions ℓ_0 and ℓ_1. Recall that for every $b \in \{0,1\}$, $\bar{\sigma}_b = [b, \bar{\sigma}]$ is the initial state vector where x has the initial state b and the initial states of all other neurons is specified by the vector $\bar{\sigma}$. The construction of ℓ_0, ℓ_1 is inductive. To avoid cumbersome notation, we start the simulation in round -1 rather than in round 0. For this first round -1, let $\ell_b(v, u, -1) = 1$ for $v \neq x$, and $\ell_b(x, u, -1) = L$ for every u and $b \in \{0,1\}$. Thus, the positive spikes (by any $v \neq x$) fired in round -1 arrive in round 0, and the negative spikes of x arrive in round $L - 1$.

To define the latency of the edges in the remaining rounds $\tau \geq 0$, we partition them into blocks, each of size L rounds where the i^{th} block is $T_i = [iL, iL + (L-1)]$ for every $i \geq 0$. We continue in steps $i = 1, \ldots$ where in step i, the latency values of $\ell_0(e, \tau), \ell_1(e, \tau)$ are defined for every edge e in the network $\mathcal{N}$, and for every round $\tau \in T_i$. For every $b \in \{0,1\}$ and a block T_i, define $A_{i,b}$ as the set of neurons that fire (hence *active*) in the first round of T_i when executing $\mathcal{N}$ with the initial configuration $\bar{\sigma}_b$, and the latency function ℓ_b. Throughout the process of defining the latency functions, we maintain these invariants at the beginning of step i:

- (I1) All the positive spikes generated at any round before the interval T_i arrive to their destination by the first round of T_i. Furthermore, all negative spikes generated at any round before the interval T_i arrive to their destination either by the first round of T_i or on the last round of T_i, namely, round $iL + (L-1)$.
- (I2) $A_{i,0} \setminus \bigcup_{i' \leq i} A_{i',1} = \emptyset$.

We define the latency for the rounds in T_i, and then show that the invariants are maintained.

Defining the latency function ℓ_1 for T_i. For every self-loop edge e and every $\tau \in T_i$, let $\ell_1(e, \tau) = 1$. For every edge $e = (x, u)$ where $u \neq x$, and $\tau \in T_i \setminus \{iL + (L-1)\}$, let $\ell_1(e, \tau) = (iL + (L-1)) - \tau$, i.e., the spike of x arrives u in the last round of the interval T_i. For $e = (x, u)$ and $\tau = iL + (L-1)$, let $\ell_1(e, \tau) = L$ so that the spike arrives in the last round of the next block, i.e., round $(i+1)L + (L-1)$. For every other edge $e = (v, u)$ with $v \neq x$, let $\ell_1(e, \tau) = (i+1)L - \tau$, i.e., the spike arrives at the first *first* round of the next block T_{i+1}.

Defining the latency function ℓ_0 for T_i. As for ℓ_1, for a self-loop edge e, we set $\ell_0(e, \tau) = 1$. For an edge $e = (x, u)$ we set $\ell_0(e, \tau)$ arbitrarily (since $x = 0$, those values are meaningless). We now fix a neuron u, and set the latency values of all its incoming edges (v, u). Since we have already defined the latency values of all edges up to block T_i, at the beginning of step i, the sets $A_{i,0}, A_{i,1}$ can be computed. Let $g_1, \ldots, g_\omega$ be the weak incoming neighbors of u in $A_{i,0}$, and $h_1, \ldots, h_s$ be the strong incoming neighbors of u in $A_{i,0}$. We Consider two cases. The neuron u is said to have a *dominant* neighbor if it has a neighbor with a *sufficiently*

large incoming weight, where the precise weight threshold depend on whether the incoming neighbor is weak or strong. Specifically, it has a dominant neighbor if it has either a weak neighbor g_j with $w(g_j, u) \geq b(u)$, or a strong neighbor h_j with $w(h_j, u) \geq b(u)/(L-1)$.

- **Case 1: u has a dominant neighbor.** Let $\ell_0(e, \tau) = (i+1)L - \tau$ for every incoming edge $e = (v, u)$. That is, we schedule all the incoming spikes of u in this block to arrive at u in the *first* round of the next block T_{i+1}.
- **Case 2: u has no dominant neighbor.** Since $\deg(u) < L-2$, we have that $\omega + s < L-2$, and in particular $\omega \leq L-2$. For each weak neuron g_j, set $\ell_0(g_j, u, iL) = j+1$. That is, the spike from g_j in round iL is scheduled to arrive at u in round $iL + (j+1)$. For each strong neighbor h_j, we split all the spikes generated by h_j during the $\omega+1$ rounds $iL, \ldots, iL+\omega$ in a balanced manner over $L-(\omega+1)$ rounds. Specifically, we set the latency values of the at most $\omega+1$ spikes by h_j during the rounds $iL, \ldots, iL+\omega$ such that in each round $\tau \in [iL+(\omega+2), (i+1)L]$, u receives at most $(\omega+1)/(L-(\omega+1))$ spikes from h_j[6]. For every $\tau \in [iL+(\omega+1), iL+(L-1)]$, let $\ell_0(h_j, u, \tau) = 1$, i.e., the spike arrives one round later. The latency of all the remaining edges e and rounds τ in T_i is set to $\ell_0(e, \tau) = (i+1)L - \tau$, so that it arrives in round $(i+1)L$.

In AppendixA.1.2, we prove that the invariants hold by induction on the number of rounds. Since the output z is required to fire when $x = 0$ but must not fire when $x = 1$, we get the desired contradiction. In Appendix A.1, we show the time lower bound of $\Omega(L^3)$ rounds. This bound is tight, and the construction while having a similar high-level ideas is slightly more involved than the size lower bound.

4 Upper Bounds

4.1 Synchronization of Logic Gates and Boolean Circuits

First observe that the simple implementation of an OR-gate works also in the asynchronous setting.

▶ **Observation 13** (OR gate). *Given input neurons $x_1, \ldots, x_n$ and output neuron z, there exists a deterministic network* $\mathsf{OR}_{\mathsf{sync}}$ *with* no *auxiliary neurons, that computes the OR gate of $x_1, \ldots, x_n$ using L rounds. I.e, it holds that: (i) If $\sigma_0(x_1) \vee \ldots \vee \sigma_0(x_n) = 0$, then $\sigma_t(z) = 0$ for every t, and (ii) If $\sigma_0(x_1) \vee \ldots \vee \sigma_0(x_n) = 1$, then there exists a round $t \in [1, L]$ such that $\sigma_t(z) = 1$. Moreover, if an input neuron fires in round τ, the output neuron z fires in some round $t \in [\tau+1, \tau+L]$.*

We next consider the more technically involved setting of synchronizing a NOT gate.

▶ **Lemma 14** (NOT gate). *There is a network* $\mathsf{NOT}_{\mathsf{sync}}$ *of size $O(L^2)$ with input neuron x and output z, that computes $NOT(x)$ within $O(L^3)$ rounds. I.e, it holds that: (i) If $\sigma_0(x) = 1$, then $\sigma_t(z) = 0$ for every t, and (ii) If $\sigma_0(x) = 0$, then there exists a round $t \in [L, \Theta(L^3)]$ such that $\sigma_t(z) = 1$.*

The following synchronous implementation assumes that the network contains a special starter neuron v^* that fires at the beginning of the simulation, regardless of the input value of x. Later on in Section 4.3, when presenting the complete synchronization scheme, this neuron v^* will receive the starting firing signal from the *global* pulse generator.

[6] For simplicity, we assume that $(\omega+1)$ divides $(L-(\omega+1))$

Network Description. The network consists of the following components, see Figure 2.

1. A *chain* $C = [c_0 = v^*, \ldots, c_{5L^2}]$ containing $5L^2 + 1$ neurons. The head of the chain is the starter neuron that fires in the first round. For every $i \geq 0$, the neuron c_i has bias $b(c_i) = 1$. Moreover, for every $i \geq 1$ the neuron c_i has an incoming edge from c_{i-1} with weight 1.
2. A *memory neuron* m that remembers the initial state of x. The memory neuron has a positive incoming edge from x, as well as a self-loop both with weight 1 and bias $b(m) = 1$.
3. A *reset* inhibitory neuron r with an edge from m of weight $w(m, r) = 1$, and bias $b(r) = 1$.
4. A collection of $L + 1$ intermediate neurons $v_0, \ldots, v_L$ that are connected to the output neuron z, where each v_i has an incoming edge from the neuron $c_{5 \cdot iL} \in C$ with weight $w(c_{i \cdot 5L}, v_i) = 1$, a self-loop of weight 1 and bias $b(v_i) = 1$. In addition, each v_i has a negative incoming edge from the reset neuron r with weight $w(r, v_i) = -\infty$. Finally, each v_i has an edge to z with weight $w(v_i, z) = 1$ and bias $b(z) = L + 1$.

The correctness of the construction and the proof of Lemma 14 are deferred to Appendix B.1.

Synchronization of a Boolean Circuit. Given the synchronized sub-networks of Observation 13 and Lemma 14, we now show how to synchronize a Boolean circuit that contains OR and NOT gates.

▶ **Lemma 15.** *Given a Boolean circuit $\mathcal{A}$ of OR and NOT gates with n inputs, k outputs, m gates and depth d, there exists a deterministic network $\mathcal{N} = \mathsf{sync}(\mathcal{A})$ with input neurons $x_1, \ldots, x_n$, output neurons $z_1, \ldots, z_k$ and $O(dL^3 + mL^2)$ auxiliary neurons, that computes $\mathcal{A}$ in $O(dL^4)$ rounds. I.e., it holds that (i) If $[\mathcal{A}(\sigma_0(x_1), \ldots, \sigma_0(x_n))]_i = 0$ then $\sigma_t(z_i, \mathcal{N}) = 0$ for every t; and (ii) If $[\mathcal{A}(\sigma_0(x_1), \ldots, \sigma_0(x_n))]_i = 1$, then[7] there exists $t \in [1, O(dL^4)]$ such that $\sigma_t(z_i, \mathcal{N}) = 1$.*

In the high-level, the network $\mathcal{N} = \mathsf{sync}(\mathcal{A})$ is obtained by replacing each gate g_i with its synchronized module $\mathsf{Sync}(g_i)$. The input neurons to the gate modules in layer j of $\mathcal{A}$ are the output neurons of the gate modules of layer $j - 1$ in $\mathcal{A}$. The network then contains a chain of length $O(d \cdot L^3)$ to control the synchronization between layers: the modules of layer j are activated only after the modules of the previous layer have completed their computation. See Appendix B.2.

4.2 Synchronization of a Single Threshold Gate

Deterministic Threshold Gate. Given a deterministic threshold gate g with Δ inputs, one can implement g using a Boolean Circuit with $\mathrm{poly}(\Delta)$ gates and depth $O(\log \Delta)$ (see Appendix B.3). Combining with the construction described in Lemma 15 we show the following:

▶ **Lemma 16.** *Given a weighted threshold gate $g = f(x_1, \ldots, x_\Delta)$, there exists a network $\mathcal{N} = \mathsf{Sync}(g)$ with Δ input neurons $x_1, \ldots, x_\Delta$, an output neuron z, and $O(\log \Delta \cdot L^3 + \mathrm{poly}(\Delta) \cdot L^2)$ auxiliary neurons that computes f within $O(\log \Delta \cdot L^4)$ rounds. I.e. the output z fires in round $\tau \in [2, O(\log \Delta \cdot L^4)]$ if and only if $f(\sigma_0(x_1), \ldots, \sigma_0(x_\Delta)) = 1$.*

[7] For a vector of n bits $x \in \{0,1\}^n$, let $[x]_i$ denote the i^{th} bit of x. I.e., if $x = (x_1, \ldots, x_n)$, then $[x]_i = x_i$.

Probabilistic Threshold Gate. We next turn to consider the more challenging setting of probabilistic threshold gates. To synchronize such gates, we first describe how to implement them by using a Boolean *circuit* $\mathcal{A}$ that contains two types of gates: deterministic threshold gates, and input-less gates which outputs 1 with probability 1/2. We hereafter denote the latter gates by uniformly random gates[8]. The output distributions of the probabilistic threshold gate and the output gate of $\mathcal{A}$ will be very close up to a small additive error of $\epsilon \in (0,1)$. The synchronized probabilistic gate will be obtained by applying the synchronization scheme of Lemma 15 on the circuit $\mathcal{A}$.

Our key result might be of independent interest in the context of Boolean circuits:

▶ **Lemma 17.** *Given a* probabilistic *threshold gate g with Δ inputs, and an error parameter $\epsilon \in (0,1)$, there exists a Boolean circuit with depth* $\text{poly}(\log\Delta, \log(1/\epsilon))$ *and a total* $\text{poly}(\Delta, \log(1/\epsilon))$ deterministic *gates. In addition, there is a collection of $O(\log(1/\epsilon))$ uniformly random gates (each outputs 1 independently w.p. 1/2), and an output gate g' that approximates g in the following sense. Letting $p(\bar{x}), p'(\bar{x})$ be the probability that g, g' output 1 given input $\bar{x}$, it holds that $|p(\bar{x}) - p'(\bar{x})| \leq \theta(\epsilon)$ for any fixed assignment of input $\bar{x}$.*

Our starting point is the following useful fact from [20]:

▶ **Observation 18.** *Let g_1 be a* probabilistic *gate with an incoming weighted sum W and bias b_1. Let g_2 be a* deterministic *threshold gate with incoming weighted sum W and bias b_2, where b_2 is sampled from the Logistic distribution with mean b_1 and scale 1. Then* $\Pr[g_1 = 1] = \Pr[g_2 = 1] = 1/(1 + e^{-(W - b_1)})$.

The observation holds as the cumulative density function of the Logistic distribution is a Sigmoidal function. Since we already know how to implement a deterministic threshold gate using a Boolean circuit, the key challenge is in sampling a value from the Logistic distribution using a small number of uniformly random gates (i.e., fair coins). This is done in two key steps. First, using $O(\log 1/\epsilon)$ uniformly random gates, we sample a value from an ϵ^4-discretization of the uniform distribution [9]. Then, we use the method of inverse transform sampling to sample from a distribution that is $\Theta(\epsilon)$-close (in L_1 norm) to the Sigmoidal distribution. For a value r sampled u.a.r in $[0,1]$, a sample from the Logistic distribution with mean b and scale 1 is given by $b + \ln(r/(1-r))$. To compute the expression $b + \ln(r/(1-r))$ using a Boolean circuit, we approximate the $\ln(x)$ function using the first $O(\log 1/\epsilon)$ terms of the Taylor expansion. The almost-Logistic sample will serve as the bias of a deterministic threshold gate and will be fed to the Boolean circuit of Lemma 16. The full description is given in Appendix B.4.

We can then synchronize the Boolean Circuit as described in Lemma 17.

▶ **Corollary 19.** *Given a probabilistic threshold gate g with Δ inputs, and an error parameter $\epsilon \in (0,1)$, there exists a network $\mathcal{N} = \mathsf{Sync}(g)$ with Δ input neurons $x_1, \ldots, x_\Delta$, an output neuron z, and* $\text{poly}(\Delta, \log 1/\epsilon) \cdot L^3$ *auxiliary neurons such that z* approximates *the gate g within* $\text{poly}(\log\Delta, \log 1/\epsilon) \cdot L^4$ *rounds in the following sense. For any fixed input $\bar{x}$, with probability at least $1 - \Theta(\epsilon)$, it holds that g outputs 1 iff z fires in some round in* $[1, \text{poly}(\log\Delta, \log 1/\epsilon) \cdot L^4]$.

[8] A uniformly random gate is a fair coin, in contrast to probabilistic threshold gate that outputs 1 based on a Sigmoidal distribution.

[9] Our sample is equivalent to sampling a value from the uniform distribution and then rounding it to the closest value of the form $i \cdot \epsilon^4$ for some integer i.

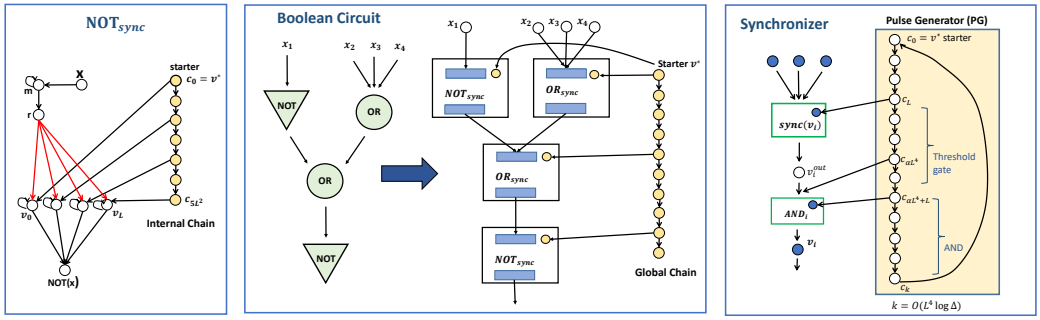


Figure 2 Left: synchronized network of a single NOT gate. Middle: A synchronized network for a Boolean circuit. Right: The transformation of a single neuron v_i in the synchronized network for the given SNN.

4.3 The Complete Synchronization Scheme

The complete synchronization scheme and the proof of Theorem 4 are given in Appendix B.5. In the high level, the construction has two parts: a global pulse generator, and a specific adaptation of the given network $\mathcal{N}$ into a network $\mathsf{sync}(\mathcal{N})$, see Figure 2.

The *pulse generator* is implemented by a directed cycle of length $k = \widetilde{O}(L^4 \log \Delta)$. The input layer and output layer in $\mathsf{sync}(\mathcal{N})$ are *exactly* as in $\mathcal{N}$. Let V be the neurons of $\mathcal{N}$. For each auxiliary neuron $v_i \in V$, we add its synchronized sub-network $\mathsf{Sync}(v_i)$ from Lemma 16 and Cor. 19. Recall that each neuron in $\mathcal{N}$ implements either a threshold gate or a probabilistic threshold gate. For each such $v_i \in V$, we also add an AND module AND_i, which receives input from the sub-network $\mathsf{Sync}(v_i)$ and the pulse generator. The neuron v_i is set to be the output neuron of this AND_i module.

References

1 Douglas B Armstrong, Arthur D Friedman, and Premachandran R Menon. Design of asynchronous circuits assuming unbounded gate delays. *IEEE Transactions on Computers*, 100(12):1110–1120, 1969.

2 Baruch Awerbuch and David Peleg. Network Synchronization with Polylogarithmic Overhead. In *31st Annual Symposium on Foundations of Computer Science, St. Louis, Missouri, USA, October 22-24, 1990, Volume II*, pages 514–522, 1990.

3 Michael J Berry II and Markus Meister. Refractoriness and neural precision. In *Advances in Neural Information Processing Systems*, pages 110–116, 1998.

4 Tobias Bjerregaard and Shankar Mahadevan. A survey of research and practices of network-on-chip. *ACM Computing Surveys (CSUR)*, 38(1):1, 2006.

5 Sami Boudkkazi, Edmond Carlier, Norbert Ankri, Olivier Caillard, Pierre Giraud, Laure Fronzaroli-Molinieres, and Dominique Debanne. Release-dependent variations in synaptic latency: a putative code for short-and long-term synaptic dynamics. *Neuron*, 56(6):1048–1060, 2007.

6 Chi-Ning Chou, Kai-Min Chung, and Chi-Jen Lu. On the Algorithmic Power of Spiking Neural Networks. In *10th Innovations in Theoretical Computer Science Conference, ITCS 2019, January 10-12, 2019, San Diego, California, USA*, pages 26:1–26:20, 2019.

7 RE Lee DeVille and Charles S Peskin. Synchrony and asynchrony in a fully stochastic neural network. *Bulletin of mathematical biology*, 70(6):1608–1633, 2008.

8 Huawei Fan, Yafeng Wang, Hengtong Wang, Ying-Cheng Lai, and Xingang Wang. Autapses promote synchronization in neuronal networks. *Scientific reports*, 8(1):580, 2018.

9 Martin Fürer. Faster integer multiplication. *SIAM Journal on Computing*, 39(3):979–1005, 2009.

10 Johan Håstad. On the Size of Weights for Threshold Gates. *SIAM J. Discrete Math.*, 7(3):484–492, 1994. doi:10.1137/S0895480192235878.

11 Scott Hauck. Asynchronous design methodologies: An overview. *Proceedings of the IEEE*, 83(1):69–93, 1995.

12 Yael Hitron and Merav Parter. Counting to Ten with Two Fingers: Compressed Counting with Spiking Neurons. *ESA*, 2019.

13 Yael Hitron and Merav Parter. Counting to Ten with Two Fingers: Compressed Counting with Spiking Neurons. *CoRR*, abs/1902.10369, 2019. arXiv:1902.10369.

14 Kaori Ikeda and John M Bekkers. Autapses. *Current Biology*, 16(9):R308, 2006.

15 Fabian Kuhn, Joel Spencer, Konstantinos Panagiotou, and Angelika Steger. Synchrony and asynchrony in neural networks. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete algorithms*, pages 949–964. SIAM, 2010.

16 Robert A. Legenstein, Wolfgang Maass, Christos H. Papadimitriou, and Santosh Srinivas Vempala. Long Term Memory and the Densest K-Subgraph Problem. In *9th Innovations in Theoretical Computer Science Conference, ITCS 2018, January 11-14, 2018, Cambridge, MA, USA*, pages 57:1–57:15, 2018.

17 Nancy Lynch and Cameron Musco. A Basic Compositional Model for Spiking Neural Networks. *arXiv preprint*, 2018. arXiv:1808.03884.

18 Nancy Lynch, Cameron Musco, and Merav Parter. Computational Tradeoffs in Biological Neural Networks: Self-Stabilizing Winner-Take-All Networks. In *Proceedings of the 8th Conference on Innovations in Theoretical Computer Science (ITCS)*, 2017.

19 Nancy Lynch, Cameron Musco, and Merav Parter. Spiking Neural Networks: An Algorithmic Perspective. In *5th Workshop on Biological Distributed Algorithms (BDA 2017)*, July 2017.

20 Nancy A. Lynch, Cameron Musco, and Merav Parter. Neuro-RAM Unit with Applications to Similarity Testing and Compression in Spiking Neural Networks. In *31st International Symposium on Distributed Computing, DISC 2017, October 16-20, 2017, Vienna, Austria*, pages 33:1–33:16, 2017.

21 Jun Ma, Xinlin Song, Wuyin Jin, and Chuni Wang. Autapse-induced synchronization in a coupled neuronal network. *Chaos, Solitons & Fractals*, 80:31–38, 2015.

22 Wolfgang Maass. Lower Bounds for the Computational Power of Networks of Spiking Neurons. *Electronic Colloquium on Computational Complexity (ECCC)*, 1(19), 1994. URL: http://eccc.hpi-web.de/eccc-reports/1994/TR94-019/index.html.

23 Wolfgang Maass. On the computational power of noisy spiking neurons. In *Advances in Neural Information Processing Systems 8 (NIPS)*, 1996.

24 Wolfgang Maass. Networks of spiking neurons: the third generation of neural network models. *Neural Networks*, 10(9):1659–1671, 1997.

25 Wolfgang Maass. Paradigms for computing with spiking neurons. In *Models of Neural Networks IV*, pages 373–402. Springer, 2002.

26 Rajit Manohar and Yoram Moses. Analyzing Isochronic Forks with Potential Causality. In *21st IEEE International Symposium on Asynchronous Circuits and Systems, ASYNC 2015, Mountain View, CA, USA, May 4-6, 2015*, pages 69–76, 2015. doi:10.1109/ASYNC.2015.19.

27 Rajit Manohar and Yoram Moses. The eventual C-element theorem for delay-insensitive asynchronous circuits. In *2017 23rd IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC)*, pages 102–109. IEEE, 2017.

28 Alain J Martin. The limitations to delay-insensitivity in asynchronous circuits. In *Beauty is our business*, pages 302–311. Springer, 1990.

29 Robert Miller. Time and the brain. *CRC Press*, 2000.

30 Saburo Muroga. *Threshold logic and its applications*. Wiley, 1971.

31 Yu P Ofman. On the algorithmic complexity of discrete functions. In *Doklady Akademii Nauk*, volume 145, pages 48–51. Russian Academy of Sciences, 1962.

32 Christos H. Papadimitriou and Santosh S. Vempala. Random Projection in the Brain and Computation with Assemblies of Neurons. In *10th Innovations in Theoretical Computer Science Conference, ITCS 2019, January 10-12, 2019, San Diego, California, USA*, pages 57:1–57:19, 2019. doi:10.4230/LIPIcs.ITCS.2019.57.

33 Huixin Qin, Jun Ma, Chunni Wang, and Ying Wu. Autapse-induced spiral wave in network of neurons under noise. *PloS one*, 9(6):e100849, 2014.

34 Alexa Riehle, Sonja Grün, Markus Diesmann, and Ad Aertsen. Spike synchronization and rate modulation differentially involved in motor cortical function. *Science*, 278(5345):1950–1953, 1997.

35 BL Sabatini and WG Regehr. Timing of synaptic transmission. *Annual review of physiology*, 61(1):521–542, 1999.

36 Jens Sparsø. Asynchronous circuit design-a tutorial. In *Chapters 1-8 in "Principles of asynchronous circuit design-A systems Perspective"*. Kluwer Academic Publishers, 2001.

37 Lili Su, Chia-Jung Chang, and Nancy Lynch. Spike-Based Winner-Take-All Computation: Fundamental Limits and Order-Optimal Circuits. *Neural Computation*, 2019.

38 Christopher S. Wallace. A Suggestion for a Fast Multiplier. *IEEE Trans. Electronic Computers*, 13(1):14–17, 1964. doi:10.1109/PGEC.1964.263830.

39 Barbeeba Wang and Nancy Lynch. Integrating Temporal Information to Spatial Information in a Neural Circuit. *arXiv preprint*, 2019. arXiv:1903.01217.

40 Ergin Yilmaz, Mahmut Ozer, Veli Baysal, and Matjaž Perc. Autapse-induced multiple coherence resonance in single neurons and neuronal networks. *Scientific Reports*, 6:30914, 2016.

A Missing Proofs for the Negative Results

Impossibility Result, Proof of Lemma 10. Assume towards contradiction there exists a network $\mathcal{N} = (V, \{x, y\}, \{z\}, w, b)$ that computes the AND gate of the initial states x_0, y_0 of the inputs x and y. It is then required that if $x_0 \wedge y_0 = 1$, then there exists a round in which z fires, and if $x_0 \wedge y_0 = 0$ then z is idle throughout the execution. Our goal is to show the existence of a bad assignment of latency values to the edges of $\mathcal{N}$. Such bad assignment exists even if we fix the latency of each edge to the same value throughout the entire execution. We begin with some quick observations.

- The state of a neuron v in round τ, namely, $\sigma_\tau(v)$, is fully determined by the network, the latency function, the input and the initial state σ_0, that is $\sigma_\tau(v) = H(\mathcal{N}, \ell, \sigma_0, x_0, y_0, v, \tau)$ for some function H.
- Given the network $\mathcal{N}$ and a latency function ℓ, the state of a neuron v in round τ is a function of the *previous* states of its incoming neighbors denoted as $u_1 \dots u_k$:

$$\sigma_\tau(v) = F_v(\sigma_{\tau-\ell(u_1,v)}(u_1), \dots, \sigma_{\tau-\ell(u_k,v)}(u_k)).$$

▶ **Definition 20.** *For a neuron v and round τ, we say that the state $\sigma_\tau(v)$ is* x-independent *(equivalently y-independent) if its value does not depend on the initial state of x, i.e if*

$$H(\mathcal{N}, \ell, \sigma_0, x_0 = 0, y_0, v, \tau) = H(\mathcal{N}, \ell, \sigma_0, x_0 = 1, y_0, v, \tau) \ .$$

▶ **Observation 21.** *A concatenation of x-independent functions is also x-independent. Specifically, for round τ and neuron v with incoming neighbors $u_1, \dots u_k$, it holds that if $\sigma_{\tau-\ell(u_1,v)}(u_1), \dots, \sigma_{\tau-\ell(u_k,v)}(u_k)$ are x-independent then $\sigma_\tau(v)$ is also x-independent.*

Given the network $\mathcal{N}$ we set the edge latencies as follows:

$$\ell(u, v) = \begin{cases} 1 & \text{if either } u = x \text{ or } v = x, \text{ but not both.} \\ 2 & \text{otherwise.} \end{cases}$$

We next show that for every neuron $v \in V$, its firing state in each round if either x-independent or y-independent. Specifically, the firing state of z in each round does not depend on both x_0 and y_0. This will contradict the assumption that z computes an AND gate of x_0 and y_0.

▷ Claim 22. For every round $\tau \geq 1$ it holds that: (1) For every $v \in V \setminus \{x\}$, the firing state $\sigma_\tau(v)$ is x-independent if τ is even and y-independent if τ is odd. (2) $\sigma_\tau(x)$ is x-independent if τ is odd and y-independent if τ is even.

Proof. By induction on the round τ. For $\tau = 1$, since all outgoing edges from y have latency 2 (except for the edge (y, x), if exists), in round 1 no neuron $v \in V \setminus \{x\}$ received a spike from y and therefore $\sigma_1(v)$ is y-independent. Because the edge from x to itself has latency 2, and the edge from y to x has latency 1, in round 1 the neuron x can receive a signal from y but not from x and therefore also $\sigma_1(x)$ is x-independent. For $\tau = 2$ and $v \neq x$, since the edges from x have latencies 1, and all other edges have latency 2, it holds that

$$\sigma_2(v) = F_v(\sigma_1(x), \sigma_0(y), \sigma_0(u_1), \ldots, \sigma_0(u_k)),$$

where $u_1, \ldots, u_k$ are the neighbors of v in $V \setminus \{x\}$. Because $\sigma_1(x)$ is x-independent, and $\sigma_0(u_i)$ are the initial states, by Observation 21 we can conclude that $\sigma_2(v)$ is x-independent. Next, for the input neuron x, since all its incoming edges (except for the self-loop) have latency 1 it holds that

$$\sigma_2(x) = F_x(\sigma_0(x), \sigma_1(u_1), \ldots, \sigma_1(u_k)).$$

Because $\sigma_1(v)$ is y-independent for all $v \neq x$, we conclude that $\sigma_2(x)$ is y-independent as well.

Assume the claim holds for every round $\tau' < \tau$ and we will show the claim holds for round τ as well. For $v \neq x$ with incoming neighbors $u_1, \ldots u_k$ in $V \setminus \{x\}$, by the definition of the latencies it holds that

$$\sigma_\tau(v) = F_v(\sigma_{\tau-1}(x), \sigma_{\tau-2}(u_1), \ldots, \sigma_{\tau-2}(u_k)).$$

If τ is even, so is $\tau - 2$ and by the induction assumption $\sigma_{\tau-2}(u_i)$ are x-independent. Because $\tau - 1$ is odd, $\sigma_{\tau-1}(x)$ is x-independent. Hence, by Observation 21 we conclude that $\sigma_\tau(v)$ is also x-independent. Similarly, if τ is odd, then by the induction assumption $\sigma_{\tau-2}(u_i)$ and $\sigma_{\tau-1}(x)$ are y-independent, and therefore $\sigma_\tau(v)$ is also y-independent. Then, for the neuron x, it holds that

$$\sigma_\tau(x) = F_x(\sigma_{\tau-2}(x), \sigma_{\tau-1}(u_1), \ldots, \sigma_{\tau-1}(u_k)).$$

If τ is odd, by the induction assumption $\sigma_{\tau-2}(x)$ as well as $\sigma_{\tau-1}(u_1), \ldots, \sigma_{\tau-1}(u_k)$ are x-independent and therefore $\sigma_\tau(x)$ is x-independent. On the other hand, if x is even, then by the induction assumption $\sigma_{\tau-2}(x)$ and $\sigma_{\tau-1}(u_1), \ldots, \sigma_{\tau-1}(u_k)$ are y-independent and therefore $\sigma_\tau(x)$ is also y-independent. ◁

Since in each round the output neuron z is either x-independent or y-independent, this contradicts the assumption and Lemma 10 follows.

A.1 Size Lower Bound for Computing $NOT(x)$

A.1.1 Reduction to Simple Networks, Proof of Lemma 12

The reduction is based on the following notion of domination between two configurations.

Domination. Given a network $\mathcal{N}$, a latency function ℓ, and a vector of starting states $\bar{\sigma}$ for all neurons but the input x, for $b \in \{0,1\}$ define $\text{pot}_b(u,\tau,\mathcal{N},\ell,\bar{\sigma})$ as the potential of neuron u in round τ in the simulation of $\mathcal{N}$ with the initial vector state $[b,\bar{\sigma}]$, i.e., with the initial state of x is being b and all other initial states are as in $\bar{\sigma}$. When $\bar{\sigma}$ is clear from the context, we may omit it, and simply write $\text{pot}_b(u,\tau,\mathcal{N},\ell)$. Given networks $\mathcal{N}_1,\mathcal{N}_2$ with vertices V_1,V_2 and latency functions ℓ_1,ℓ_2 respectively, we say that $\langle\mathcal{N}_1,\bar{\sigma}_1\rangle$ and $\langle\mathcal{N}_2,\bar{\sigma}_2\rangle$ are *compatible* if $V_1 \subseteq V_2$ and σ_1 and σ_2 agree on the mutual vertices of V_1, i.e., $\sigma_1(u) = \sigma_2(u)$ for every $u \in V_1$.

In our arguments, we consider a pair of compatible configurations $\langle\mathcal{N}_1,\bar{\sigma}_1\rangle$ and $\langle\mathcal{N}_2,\bar{\sigma}_2\rangle$ along with latency functions ℓ_1,ℓ_2 for these configurations. We say that $\langle\mathcal{N}_1,\bar{\sigma}_1,\ell_1\rangle$ *dominates* $\langle\mathcal{N}_2,\bar{\sigma}_2,\ell_2\rangle$ if $\langle\mathcal{N}_1,\bar{\sigma}_1\rangle$ and $\langle\mathcal{N}_2,\bar{\sigma}_2\rangle$ are compatible and in addition:

- $\text{pot}_1(u,\tau,\mathcal{N}_1,\ell_1,\bar{\sigma}_1) \leq \text{pot}_1(u,\tau,\mathcal{N}_2,\ell_2,\bar{\sigma}_2)$ for every $u \in V_1 \setminus \{x\}$ and $\tau \geq 0$.
- $\text{pot}_0(u,\tau,\mathcal{N}_1,\ell_1,\bar{\sigma}_1) \geq \text{pot}_0(u,\tau,\mathcal{N}_2,\ell_2,\bar{\sigma}_2)$ for every $u \in V_1 \setminus \{x\}$ and $\tau \geq 0$.

Let V, V_{simple} be the vertex sets of $\mathcal{N}$ and $\mathcal{N}_{simple}$ respectively. Let $\bar{\sigma}_{simple}$ be the initial state vector that agrees with $\bar{\sigma}$ on all vertices in V_{simple}. Thus, $\langle\mathcal{N}_{simple},\bar{\sigma}_{simple}\rangle$ and $\langle\mathcal{N},\bar{\sigma}\rangle$ are compatible. For a number of rounds r, a latency function ℓ is *r-good* for $\langle\mathcal{N},\bar{\sigma}\rangle$ if $\mathcal{N}$ computes $NOT(x)$ within r rounds under ℓ when starting from the initial state vector $\bar{\sigma}$. Our proof strategy is as follows. We will show that every latency function ℓ_{simple} is r-good for $\langle\mathcal{N}_{simple},\bar{\sigma}_{simple}\rangle$, by showing that there exists a function ℓ such that

$$\langle\mathcal{N}_{simple},\bar{\sigma}_{simple},\ell_{simple}\rangle \text{ dominates } \langle\mathcal{N},\bar{\sigma},\ell\rangle.$$

Given a latency function ℓ_{simple} for the network $\mathcal{N}_{simple}$, let ℓ be a latency function for $\mathcal{N}$ which is similar on excitatory neurons and gives inhibitory neurons the latency value of the neuron x. I.e., $\ell(v,u,\tau) = \ell_{simple}(v,u,\tau)$ for every pair of excitatory neurons. In addition, $\ell(v',u,\tau) = \ell_{simple}(x,u,\tau)$ for every inhibitory neuron v', and a neuron $u \in V_{simple}$. All remaining latency values (i.e., the incoming edges to the inhibitors of $\mathcal{N}$) can be chosen arbitrarily.

We show by induction on the round τ, that (i) $\text{pot}_1(u,\tau,\mathcal{N}_{simple},\ell_{simple}) \leq \text{pot}_1(u,\tau,\mathcal{N},\ell)$ for every $u \in V_{simple} \setminus \{x\}$, $\tau \geq 0$, and (ii) $\text{pot}_0(u,\tau,\mathcal{N}_{simple},\ell_{simple}) \geq \text{pot}_0(u,\tau,\mathcal{N},\ell)$ for every $u \in V_{simple} \setminus \{x\}$ and $\tau \geq 0$. For $\tau = 0$, this is true as the potential values in $\tau = 0$ are simply the initial states, and the vector of initial states of $\bar{\sigma}$ and $\bar{\sigma}_{simple}$ are compatible. For the induction step, let $\tau \geq 1$, and assume correctness for all $\tau' \leq \tau - 1$. We will prove the claims for round τ. Let u be a neuron in $V_{simple} \setminus \{x\}$, and let $v_1,\ldots,v_k$ be its incoming excitatory neighbors.

The initial state of x is 0. By the induction assumption, it holds that $\text{pot}_0(u,\tau',\mathcal{N}_{simple},\ell_{simple}) \geq \text{pot}_0(u,\tau',\mathcal{N},\ell)$ for every round $\tau' \leq \tau - 1$ and every neuron $v \in \{v_1,\ldots,v_k\}$. Thus every excitatory neuron v_i that fires in round τ' in the simulation of $\mathcal{N}$, also fires in round τ' in the simulation of $\mathcal{N}_{simple}$ for every $\tau' \leq \tau - 1$. Combining with the definition of the latency function ℓ, we get that each spike from v_i that arrives to u at round τ of the simulation of $\mathcal{N}$ also arrives u in round τ in the simulation of $\mathcal{N}_{simple}$. Let ω be an inhibitory incoming neighbor of u in $\mathcal{N}$, then ω does not exist in $\mathcal{N}_{simple}$. Also note that since $\sigma_0(x) = 0$, a negative spike from x never arrives at u in the simulation of network

$\mathcal{N}_{simple}$. Therefore, no negative spikes arrive at u in $\mathcal{N}_{simple}$. Summing over the positive and negative spike weights, we get that $\text{pot}_0(u, \tau, \mathcal{N}_{simple}, \ell_{simple}) \geq \text{pot}_0(u, \tau, \mathcal{N}, \ell)$ for every $u \in V_1 \setminus \{x\}$.

The initial state of x is 1. The initial state of x is 1: By the induction assumption it holds that $\text{pot}_1(u, \tau', \mathcal{N}_{simple}, \ell_{simple}) \leq \text{pot}_1(u, \tau', \mathcal{N}, \ell)$ for every round $\tau' \leq \tau - 1$ and every $u \in \{v_1, \ldots, v_k\}$. Thus every v_i that fires in round τ' in the simulation of $\mathcal{N}_{simple}$, also fires in round τ' in the simulation of $\mathcal{N}$. Combining with the definition of the latency function ℓ, we get that each spike from v_i that arrives at u in round τ of the simulation of $\mathcal{N}_{simple}$ also arrives at u in round τ in the simulation of $\mathcal{N}$.

Let ω be an inhibitory incoming neighbor of u in $\mathcal{N}$. By the definition of the latency function ℓ, and the fact that x fires in every round, for each spike that ω fires and arrives at u in round τ in $\mathcal{N}$, there is a spike from x that arrives at u in round τ in $\mathcal{N}_{simple}$. Since the edges from x have weight $-\infty$, we get that the negative spikes weight arriving at u in round τ in $\mathcal{N}_{simple}$ are larger (in absolute value) than the negative spikes in $\mathcal{N}$. Thus, summing up the both positive and negative spike weights, we get that $\text{pot}_1(u, \tau, \mathcal{N}_{simple}, \ell_{simple}) \leq \text{pot}_1(u, \tau, \mathcal{N}, \ell)$ for every $u \in V_{simple} \setminus \{x\}$. This proves the induction step for round τ. We get that $\langle \mathcal{N}_{simple}, \bar{\sigma}_{simple}, \ell_{simple} \rangle$ dominates $\langle \mathcal{N}, \bar{\sigma}, \ell \rangle$.

Finally, we show that if $\langle \mathcal{N}_1, \bar{\sigma}_1, \ell_1 \rangle$ dominates $\langle \mathcal{N}_2, \bar{\sigma}_2, \ell_2 \rangle$ and ℓ_2 is r-good for $\langle \mathcal{N}_2, \bar{\sigma}_2 \rangle$, then also ℓ_1 is r-good for $\langle \mathcal{N}_1, \bar{\sigma}_1 \rangle$.

Consider the simulation of $\langle \mathcal{N}_2, \bar{\sigma}_2, \ell_2 \rangle$, and assume that the initial state of x is 0. Since ℓ_2 is r-good for $\mathcal{N}_2$, there is a round $\tau \leq r$ in which z fires in $\mathcal{N}_2$. Since $\langle \mathcal{N}_1, \bar{\sigma}_1, \ell_1 \rangle$ dominates $\langle \mathcal{N}_2, \bar{\sigma}_2, \ell_2 \rangle$, we can apply the condition of domination for z, τ and get that z also fires in round τ in $\mathcal{N}_1$. Now, assume that the initial state of x is 1. Since ℓ_2 is r-good for $\mathcal{N}_2$, there is *no* round τ in which z fires in $\mathcal{N}_2$. Since $\langle \mathcal{N}_1, \bar{\sigma}_1, \ell_1 \rangle$ dominates $\langle \mathcal{N}_2, \bar{\sigma}_2, \ell_2 \rangle$, we can apply the condition of domination for z and every τ, and get that z also never fires in $\mathcal{N}_1$. Hence, ℓ_1 is r-good for $\mathcal{N}_1$. This completes the proof of the lemma.

A.1.2 Size Lower Bound for Simple Networks

We show that the latency values defined in the step i satisfy the invariant in the beginning of step $i + 1$.

Proving that the Invariants Hold. For round $i = 0$, the correctness of invariant (I1) hold since in the first round $\tau = -1$ all positive spikes are set to arrive in round 0 in both ℓ_0, ℓ_1, while a spike from x arrives at round $\tau = L - 1$. As for the correctness of invariant (I2), note that both simulations are similar for all neurons $V \setminus \{x\}$, and, again, a spike from x arrives only in round $L - 1$. Therefore the same neurons are active in round $\tau = 0$. Hence $A_{0,0} = A_{0,1}$ and we get correctness for (I2). We now show that the invariants are preserved after each step. Assume that the correctness holds at the beginning of each step $j \leq i$, and consider now the beginning of step $i + 1$.

- (I1) By the construction, in all of the cases, the values we set for ℓ_0, ℓ_1 in step i are such that all spikes except the spike from x generated at round $iL + (L - 1)$ which are generated in T_i arrive at some round $\tau \leq (i + 1)L$, i.e. by the first round of T_{i+1}. Furthermore, spikes from x generated at round $iL + (L - 1)$ is set to arrive in round $(i + 1)L + (L - 1)$. The invariant holds by combining with the correctness for all steps $i' \leq i$.
- (I2) We start by proving the following auxiliary claim.

▷ Claim 23. Consider the simulation of $\mathcal{N}$ with initial state $\bar{\sigma}_b$ and latency function ℓ_b, and let τ be round in T_i. Then:

1. For a strong neuron $u \in A_{i,1}$, u fires iff $\tau \in [iL, iL+(L-2)] \subseteq T_i$. For a strong neuron $u \in A_{i,0}$, u fires for every $\tau \in T_i$.
2. For a weak neuron $u \in A_{i,b}$, u fires iff $\tau = iL$.
3. For $u \notin A_{i,b}$, u is not active in round τ.

Proof.

Case $b = 1$: We start by showing that all three claims hold for $b = 1$. By the definition of ℓ_1, the only positive spikes received by any neuron in some round $\tau \in [iL+1, iL+(L-1)]$ are self-loop spikes. Since a strong active neuron $u \in A_{i,1}$ receives an inhibiting spike from x in round $iL + (L-1)$, it is not active in this last round. For a weak neuron u', its spike from the self-loop is not strong enough to make it active. Lastly, for $u \notin A_{i,1}$ since no negative spikes arrive at u in round iL we have that $b(u) > 0$. Due to the fact that no spikes arrive at u in round τ, we get that u stays inactive.

Case $b = 0$: claim (1) holds since a strong $u \in A_{i,0}$ never gets inhibited as x never fires. We will now consider claim (2) and (3) for a neuron u that is either a *weak* neuron, or a strong neuron that is *not* in $A_{i,0}$. By Invariant (I1), all the positive spikes from the previous blocks arrived by the first round of T_i. Thus if u fires in any round $\tau \in [iL+1, iL+(L-1)]$ (i.e., any round which is not the first one in T_i), this must be due to the incoming spikes generated in the first round of T_i. We will know prove by induction on the round τ that u does not fire in any round in $[iL+1, iL+(L-1)]$.

Induction Base, Round $\tau = iL + 1$: By the definition of the latency function ℓ_0, no spike arrives at u from an incoming neighbor in that round. Therefore, if u is weak, then a spike from itself will not make it active in round τ. In addition, if $u \notin A_{i,0}$ then u did not fire in round iL, and thus receives no self-spike in round τ. Since no negative spikes arrive at u in round iL, for every $u \notin A_{i,0}$, it must hold that $b(u) > 0$.

Induction Step $\tau \geq iL + 2$. Assume that the claims (2,3) hold up to round $\tau - 1$ and consider round $\tau \geq iL + 2$. Consider first the case that u has either a weak neighbor g_j with $w(g_j, u) \geq b(u)$, or alternatively a strong neighbor h_j with $w(h_j, u) \geq b(u)/(L-1)$. Then by the definition of ℓ_0 (Case I in our definition), all the spikes fired by the incoming neighbors of u are scheduled to arrive in the first round of T_{i+1}. Therefore, u does not receive any spike in round τ, and remains inactive.

Next, consider the complimentary case where all the weak neighbors g_j satisfy that $w(g_j, u) < b(u)$, and all the strong neighbors h_j satisfy $w(h_j, u) < b(u)/(L-1)$.

Case (1): $\tau \in [iL + 2, iL + \omega + 1]$. By the induction assumption on $\tau - 1$, u did not fire in round $\tau - 1$. By the definition of ℓ_0, u receives a spike from at most one weak neighbor g_j in round τ, and since $w(g_j, u) < b(u)$, it does not fire in this round.

Case (2): $\tau \in [iL + (\omega + 2), iL + (L-1)]$. Let h_j be a strong active neighbor of u. By the definition of ℓ_0, in round τ, u receives at most $(\omega+1)/(L-(\omega+1))$ of the spikes that fired by h_j during the interval $[iL, iL + \omega]$. Furthermore, there is one additional spike that h_j fired at round $\tau - 1$, that arrives at u in round τ. Note that u does not receive spikes from weak neighbors in round τ since all spikes from weak neighbors arrive in an earlier round $\tau'' \in [iL + 2, iL + (\omega + 1)]$. In addition, since u did not fire in round $\tau - 1$, it also does not get any self spikes in round τ. Overall, u receives at most $((\omega+1)/(L-(\omega+1))) + 1$ spikes by strong neighbors in round τ, and no other spikes (by a weak neighbor or by u). Since the spikes from the strong

neighbors have weight of at most $b(u)/(L-1)$, and there are s strong active neighbors, the overall weighted sum of the received spikes at round τ is at most

$$s \cdot ((\omega+1)/(L-(\omega+1))+1) \cdot \frac{b(u)}{L-1} <$$

$$s \cdot ((\omega+1)/s+1)\frac{b(u)}{L-1} = (s+\omega+1)\frac{b(u)}{L-1} < \quad b(u) \ , \tag{2}$$

where both inequalities follow as $s+\omega < L-2$. Therefore, u does not get activated at round τ, claims (2)+(3) follow. ◁

We are now ready to prove the induction step for (I2). Assume towards contradiction that there exists a neuron $u \in A_{i+1,0} \setminus \bigcup_{i' \leq i+1} A_{i',1}$. Since u is active in the first round of T_{i+1}, using Claim 23(3), it must have an incoming neighbor that fires in the *first round* of the previous block. Let $A_{i,0}(u) = \Gamma_{in}(u) \cap A_{i,0}$ be those neighbors.

▷ Claim 24. For every strong neuron $v \in A_{i,0}(u)$, it holds that $w(v,u) < b(u)/(L-1)$.

Proof. Consider such strong $v \in A_{i,0}(u)$. By Invariant (I2) for the beginning of step i, there exists a round $j \leq i$ such that $v \in A_{j,1}$. When running $\mathcal{N}$ with initial state $\bar{\sigma}_1$ and the latency function ℓ_1, by Claim 23(1), v fires in all of the $L-1$ rounds of $[jL, jL+(L-2)]$. By the construction of ℓ_1, all these $L-1$ spikes arrive in round $(j+1)L$. Therefore, if $(L-1) \cdot w(v,u) \geq b(u)$, then u is activated in round $(j+1)L$, i.e., $u \in A_{j+1,1}$, in contradiction to the definition of u. Therefore $(L-1) \cdot w(v,u) < b(u)$ for every strong neuron v. ◁

▷ Claim 25. For every weak neuron $v \in A_{i,0}(u)$, it holds that $w(v,u) < b(u)$.

Proof. Consider such weak neuron $v \in A_{i,0}(u)$. When running $\mathcal{N}$ with initial state $\bar{\sigma}_1$ and latency function ℓ_1, by Claim 23(2), v fires once in the interval T_j, i.e., in the first round jL. By the construction of ℓ_1 this spike arrives in round $(j+1)L$. Therefore, if $w(v,u) \geq b(u)$, then u is activated in round $(j+1)L$, implying that $u \in A_{j+1,1}$, contradiction to the definition of u. We therefore conclude that $w(v,u) < b(u)$ for every weak neuron $v \in A_{i,0}(u)$. ◁

Let ω, s be the number of weak (resp., strong) neurons in $A_{i,0}(u)$. By Claims 25 and 23(2), all active weak neurons in T_i fire only in the first round of that block. By the definition of the latency function, all these spikes are scheduled to the first ω rounds in T_i, and therefore none of them is scheduled to arrive on the first round of T_{i+1}. This implies that u fires in that round due to the spikes generated by its strong neighbors. By Claim 24, $w(v,u) < b(u)/(L-1)$ for each such strong neighbor v of u. This in particular implies that u is a weak neuron, and by Claim. 23 it did not fire in the last round of T_i. By the definition of the latency function, the spikes generated by such strong neighbors are divided almost evenly among $L-\omega$ rounds, up to the first round of T_{i+1}. Each round gets at most $s \cdot (\omega/(L-\omega)+1)$, which is strictly less than $b(u)$ by Eq. (2). Leading to contradiction for the assumption that $u \in A_{i+1,0}$.

Since ℓ_1 is a good latency function when starting with $x=1$, we have that z never fires and thus $z \notin \bigcup A_{i,1}$. By applying invariant (I2) on the output neuron z for every round $\tau \geq 0$, we get that $z \notin \bigcup A_{i,0}$. By using Claim 23, we get that z never fires with ℓ_0 and $x=0$. Contradiction to the fact that $\mathcal{N}$ solves $NOT(x)$.

A.2 Time Lower Bound for Computing $NOT(x)$

In this section we show the following.

▶ **Lemma 26.** *Every network $\mathcal{N}$ that computes $NOT(x)$ in the L-bounded asynchronous setting requires $\Omega(L^3)$ rounds.*

By Lemma 12, we restrict attention to a simple network $\mathcal{N} = \mathcal{N}_{simple}$ with one input neuron x that computes $NOT(x)$. Similarly to the size lower bound, we define two *conflicting* latency functions ℓ_0 and ℓ_1, such that if ℓ_1 is good when $x_0 = 1$, then the output neuron z of $\mathcal{N}$ fires after $\Omega(L^3)$ rounds in the simulation with the latency function ℓ_0 and $x_0 = 0$.

- The simulation with the latency function ℓ_0 is partitioned into consecutive blocks of L rounds, $T_i = [iL, iL + (L-1)]$ for every $i \in \mathbb{N}$.
- The simulation with the latency function ℓ_1 is based on the notion of *important* and *unimportant* rounds. Consider the L-round interval $T_k = [k \cdot L, k \cdot L + (L-1)]$ for $k \in \mathbb{N}$. Among the first $L/2$ rounds, there is an important round once every 16 rounds, and the rest are unimportant. Furthermore, each of the last $L/2$ rounds of the interval are unimportant. I.e., the important rounds in the interval are $\{kL + 16j \mid 16j < L/2, j \in \mathbb{N}\}$. Denote by τ_i the i^{th} important round in the simulation. Note that by definition $\tau_{i+1} - \tau_i \leq L/2$. In our arguments, the configuration of the network in the i^{th} important round τ_i of the simulation with ℓ_1 and $x_0 = 1$ will be compared against the configuration in round iL (i.e., the first round of the block T_i) in the simulation with ℓ_0 and $x_0 = 0$.
- Active subsets of neurons: For every $i \in \mathbb{N}$, let $A_{0,i}$ be the firing neurons (hence *active*) of round $i \cdot L$ (the first round of the block T_i) in the simulation of $\langle \mathcal{N}, \sigma_0, \ell_0 \rangle$. Similarly, let $A_{1,i}$ be the firing neurons in round τ_i of the simulation of $\langle \mathcal{N}, \sigma_1, \ell_1 \rangle$. Also define $A'_{b,i} = A_{b,i} \setminus \bigcup_{j \leq i-1} A_{b,j}$, the neurons that fire for the first time in "round" i.
- For every neuron u, $b \in \{0,1\}$ and $i \in \mathbb{N}$, let $A_{b,i}(u) = A_{b,i} \cap N_{in}(u)$, $A'_{b,i}(u) = A'_{b,i} \cap N_{in}(u)$ where $N_{in}(u)$ is the set of incoming neighbors of u.
- For a subset of neurons $V' \subseteq V$ and a neuron u, let $w(V', u) = \sum_{v \in V'} w(v, u)$. Moreover, let $\mathbf{S}(V')$ and $\mathbf{W}(V')$ be the strong[10] and weak (respectively) neurons subsets of V'.

A.2.1 Defining the latency functions ℓ_0 and ℓ_1

Throughout, a spike event is represented by a triplet $\langle v, u, \tau \rangle$ where $v \in N_{in}(u)$ fires in round τ. Since the functions are nice, the latency values for the self spikes $\langle u, u, \tau \rangle$ for every u and τ are set to 1. For technical reasons, it is more convenient to start the simulations in round -1, rather than in round 0. For this first round -1, let $\ell_b((v,u), -1) = 1$ for every u and every $v \neq x$, and $\ell_b((x,u), -1) = L$ for every u and $b \in \{0,1\}$. As a result, the positive spikes (by any $v \neq x$) fired in round -1 arrive to their destination in round 0, and the negative spikes of x arrive in round $L-1$. We now define the latency values for the remaining spikes.

Defining the function ℓ_0. Note that when $x_0 = 0$, x never fires and thus there is no need to define ℓ_0 values for the spikes of x. We define ℓ_0 iteratively in a block by block manner. Here we do not accumulate spikes and spikes generated in the i^{th} block T_i will arrive by the first round of the $(i+1)^{th}$ block T_{i+1}. Fix a block $T_i = [iL, iL + (L-1)]$ for $i \geq 0$ and assume that the latency values ℓ_0 for all prior spikes in rounds $\tau < iL$ have already been

[10] Recall that a neuron u is strong if $w(u,u) \geq b(u)$ and it is weak otherwise.

fixed. Thus the active set $A_{0,i}$ can be determined. First, the algorithm checks if there is a way to spread all the spikes generated in rounds of T_i among the interval $[iL+2,(i+1)L]$, in a way that guarantees that u will not fire in *any* of the rounds this interval. In particular, *no* spike is scheduled to arrive in round $iL+1$. Otherwise, all spikes generated in this block are scheduled to arrive in round $(i+1)L$ (the first round of the $(i+1)^{th}$ block).

Defining the function ℓ_1. The definition of the function ℓ_1 is more involved. Unlike the function ℓ_0 in which all spikes generated in block T_i are scheduled by round $(i+1)L$, here the setting is slightly more sensitive. Specifically, the scheduling algorithm of ℓ_1 will make sure that non-self spikes arrive to their destination only in important rounds.

Spikes by the input (inhibitory neuron) x: All spikes from x are scheduled to arrive in the last round of the blocks T_i, namely, in rounds of the form $i \cdot L + (L-1)$. Formally, for every spike $\langle x,u,\tau\rangle$ where $\tau = i\cdot L+(L-1)$ for some $i\in\mathbb{N}$, let $\ell_1((x,u),\tau) = L$ thus arriving in round $\tau + L = (i+1)L + L - 1$. For every $\tau \in [iL, iL+(L-2)]$, let $\ell_1((x,u),\tau) = (iL+(L-1)) - \tau$, thus arriving in round $iL+(L-1)$ as desired.

Spikes by $v \neq x$: The latency values are defined in a round by round fashion, such that for every important round τ_i, every neuron u gets activated if possible. Otherwise the arrival of the spikes towards u are postponed (when possible) to the next important round τ_{i+1}. The spikes generated at non-important rounds will be always delayed to the next important round. This is always possible as the distance to the next important round is at most $L/2$. For a subset of spikes S, let $w(S) = \sum_{\langle v,u,\tau\rangle\in S} w(v,u)$ be the total weight of the spikes in S. For every important round τ_i, we will maintain a list of pending spikes $\mathcal{R}_{\tau_i}(u)$ towards u that were not yet scheduled. In every step $\tau \geq 0$, the algorithm will schedule the spikes generated in this round. If the round τ is important, then the algorithm will also make decisions regarding the set of pending spikes $\mathcal{R}_{\tau_i}(u)$.

We will keep the invariant that at the beginning of step τ, the latency value of all spikes scheduled to arrive by round τ has already been determined. As we will see, the non-self spikes will always be scheduled to arrive in important rounds. As a result, a neuron u fires in an unimportant round τ iff u is strong and it fired in round $\tau-1$. Initially, for every neuron u, the algorithm adds every non-self spike $\langle v,u,-1\rangle$ to $\mathcal{R}_{\tau_i}(u)$. For every $\tau \geq 0$, we consider the following algorithm.

- All self-spikes $\langle u,u,\tau'\rangle$ are given a latency value of $\ell(u,u,\tau') = 1$.
- **Handling important rounds τ_i.** Consider a neuron u. If u fired in round $\tau_i - 1$, add the self-spike $\langle u,u,\tau_i-1\rangle$ to the pending spike set $\mathcal{R}_{\tau_i}(u)$. If the total weight of its pending spikes (towards u) is sufficiently large to make u fire, all the non-self spikes are scheduled to arrive in τ_i. Formally, if $w(\mathcal{R}_{\tau_i}(u)) \geq b(u)$, schedule all these spikes to round τ_i by setting $\ell(v,u,\tau') = \tau_i - \tau'$ for every spike $\langle v,u,\tau'\rangle \in \mathcal{R}_{\tau_i}(u)$.
  Otherwise, if the total weight of pending spikes is small, i.e., $w(\mathcal{R}_{\tau_i}(u)) < b(u)$, the non-self spikes are deferred to the next important round τ_{i+1} if possible (i.e., if the latency does not exceed its upper bound L). Formally, for every non-self pending spike $\langle v,u,\tau'\rangle \in \mathcal{R}_{\tau_i}(u)$, if $\tau_{i+1} - \tau' > L$ then let $\ell((v,u),\tau') = L$ (i.e., $\langle v,u,\tau'\rangle$ cannot be further deferred). Otherwise, add $\langle v,u,\tau'\rangle$ to the pending spike set $\mathcal{R}_{\tau_{i+1}}(u)$ of the next important round τ_{i+1}.
  Finally, all spikes generated in round τ_i are also (safely) added to the pending list $\mathcal{R}_{\tau_{i+1}}(u)$.
- **Handling unimportant rounds.** The non-self spikes towards u generated in round τ are added to the pending spike set $\mathcal{R}_{\tau_{i+1}}(u)$ of the next important round τ_{i+1} (after round τ).

First observe that the function ℓ_1 is valid: All self-spikes have a latency value of 1. Moreover, the non-self spikes have a latency value in $[1, L]$. To see this observe that for unimportant round τ, a non-self spike $\langle v, u, \tau \rangle$ is added to the pending list $\mathcal{R}_{\tau_{i+1}}(u)$ where τ_{i+1} is the next important round after τ. Due to the fact that $\tau_{i+1} - \tau_i \leq L/2$, this assignment is valid. In addition, the pending spikes $\langle v, u, \tau \rangle \in \mathcal{R}_{\tau_i}(u)$ are deferred to τ_{i+1} only if $\tau_{i+1} - \tau \leq L$.

A.2.2 Proof of Lemma 26

The key lemma that establishes Lemma 26 is the following:

▶ **Lemma 27.** *For every neuron $u \neq x$ with $u \in A'_{0,i}$ for $i < L^2/1024$, there exists some $i' \leq i$ such that $u \in A'_{1,i'}$.*

By the correctness of the simple network $\mathcal{N}$, the output neuron z should not fire when $x_0 = 1$ and with the latency function ℓ_1. In other words, $z \notin A_{1,i'}$ for any i'. By Lemma 27, we get that z can only be in $A'_{0,j}$ for some $j \geq L^2/1024$, hence firing when $x_0 = 0$ only after $\Omega(L^3)$ rounds. We start with the following simple observation.

▶ **Observation 28.** *In the simulation of $\langle \mathcal{N}, \bar{\sigma}_0 \rangle$ with ℓ_0, it holds for every i that: (i) each strong neuron $s \in A_{0,i}$ fires in every round of block T_i; (ii) each weak neuron $\omega \in A_{0,i}$ which is not x fires only in the first round of block T_i; and (iii) every neuron $v \notin A_{0,i}$ does not fire in any round of block T_i.*

Proof. (i). In the simulation with ℓ_0 with $x_0 = 0$ there are no inhibiting spikes, and if a strong neuron s fires in some round, it will keep on firing for the rest of the simulation.

(ii). By the definition of the latency function ℓ_0, no spikes from incoming neighbors of the weak neuron ω arrive in round $iL + 1$, the second round of block T_i. We will prove by induction on $\tau \in [iL + 1, iL + (L - 1)]$ that ω does not fire in round τ. For the base of the induction, since $\omega \neq x$ is excitatory and weak, it holds that $0 \leq w(\omega, \omega) < b(\omega)$, thus ω does not fire in round $iL + 1$. Assume that the claim holds up to round $\tau \geq iL + 1$ and consider round $\tau + 1$. Since ω did not fire in round τ by the induction assumption, it does not receive a self spike in round $\tau + 1$. By the definition of the function ℓ_0, the non-self spikes that arrive in round $\tau + 1 < (i + 1)L$ cannot make ω fire. Thus ω does not fire in $\tau + 1$ and (ii) holds.

(iii). Let $v \notin A_{0,i}$, i.e., v did not fire in round iL. Since v does not receive negative spikes in round iL (as the spikes of x are always scheduled to the last round of the blocks). We can then conclude that $b(v) > 0$. Since in round $iL + 1$, it receives no self-spike and no other spike, it also did not fire in round $iL + 1$. The argument then follows inductively in the same manner as in (ii). ◀

We next state the following claim which is crucial to complete the key lemma.

▷ Claim 29. Fix a neuron $u \in A'_{0,i}$ such that for every $v \in A_{0,i-1}$ it holds that $w(v, u) < b(u)$. Then the total weight of spikes fired towards u in block T_{i-1} is at least $L \cdot b(u)/8$.

We first complete the proof of Lemma 27 and only then prove Claim 29.

Proof of Lemma 27. The proof is shown by induction on the block i. For the base case of $i = 0$, note that the initial states and the latency functions for the neurons $V \setminus \{x\}$ in both simulations are the same, and that spikes from x (that exist only in the simulation with ℓ_1) arrive only in round $L - 1$. This implies that in both simulations the same neurons (except for x) are active in round $\tau = 0$, hence $A_{0,0} = A_{1,0} \setminus \{x\}$. Now consider the block T_i for $1 \leq i < L^2/1024$. Let $u \in A'_{0,i}$, i.e. u fires for the first time in round iL in the simulation with ℓ_0 and $x_0 = 0$.

Case 1: There exists a previously firing dominant incoming neighbor: First assume that u has some incoming neighbor $v \in A_{0,i-1}$ with $w(v,u) \geq b(u)$. By definition $v \in A'_{0,j}(u)$ for some $j \leq i-1$, and then by the induction assumption $v \in A'_{1,i'}$ for some $i' \leq j \leq i-1$. By definition of the latency function ℓ_1, since $w(v,u) \geq b(u)$, the total weight of spikes from incoming neighbors will be sufficient to activate u in the next important round, $\tau_{i'+1}$. Therefore, $u \in A_{1,i'+1}$, which implies $u \in A'_{1,i''+1}$ for some $i'' \leq i'+1$. Since $i'' \leq i'+1 \leq i$ the condition holds.

Case 2: All previously firing incoming neighbors are not dominant: By applying Claim 29 on u and block T_i, we get that the total weight of spikes fired towards u in block T_{i-1} is at least $L \cdot b(u)/8$. Due to Observation 28, we get

$$L \cdot w(\mathbf{S}(A_{0,i-1}(u)),u) + w(\mathbf{W}(A_{0,i-1}(u)),u) \geq \frac{L}{8} \cdot b(u).$$

By the definition of $A'_{0,j}$ and the induction assumption, it holds that

$$A_{0,i-1} \subseteq \bigcup_{j \leq i-1} A'_{0,j} \subseteq \bigcup_{i' \leq i-1} A'_{1,i'}.$$

We now consider the simulation with $x_0 = 1$ and the latency function ℓ_1, and partition all the rounds until τ_i into k blocks of L rounds (expect perhaps the last one). Formally, for every $j \leq k-2$, let $B_j = [jL, jL+(L-1)]$ and let $B_{k-1} = [(k-1)L, \tau_{i-1}+15]$. Denote by $\mathbf{S}(B_j)$ and $\mathbf{W}(B_j)$ the strong and weak (respectively) incoming neighbors of u that fire in some round of B_j. Using these notations, we can write

$$\sum_{j=0}^{k-1} L \cdot w(\mathbf{S}(B_j),u) + w(\mathbf{W}(B_j),u) \geq \frac{L}{8} \cdot b(u).$$

Case 2.1: Most of the weight is in the last block. We first assume that

$$L \cdot w(\mathbf{S}(B_{k-1}),u) + w(\mathbf{W}(B_{k-1}),u) \geq \frac{L}{16} \cdot b(u).$$

Consider the algorithm that defines ℓ_1, and recall that $\mathcal{R}_{\tau_{i'}}(u)$ is the set of pending spikes that were not yet scheduled when the algorithm considered the important round τ_i. The interesting case is when u did not fire in any round of B_{k-1}. In such a case, all the spikes generated towards u in the rounds of B_{k-1} were added to the pending list of $\mathcal{R}_{\tau_i}(u)$. Note that each strong neuron $v \in \mathbf{S}(B_{k-1})$ fires at least 16 spikes in B_{k-1}, since $\tau_i - \tau_{i-1} = 16$. Furthermore, each $v \in \mathbf{W}(B_{k-1})$ fires at least one spike in B_{k-1}. Moreover, the gap between any $\tau_{i'} \in B_{k-1}$ and τ_i is at most L rounds, so they do not exceed the maximal latency in τ_i. Altogether, we get that

$$\begin{aligned} &w(\mathcal{R}_{\tau_i}(u)) \geq 16 \cdot w(\mathbf{S}(B_{k-1}),u) + w(\mathbf{W}(B_{k-1}),u) \geq \\ &\frac{16}{L} \cdot (L \cdot w(\mathbf{S}(B_{k-1}),u) + w(\mathbf{W}(B_{k-1}),u)) \geq b(u)\ . \end{aligned} \tag{3}$$

Therefore u fires in τ_i and $u \in A_{1,i'}$ for some $i' \leq i$ as desired.

Case 2.2: Most of the weight is in the first $k-1$ blocks. It remains to consider the complementary case where

$$\sum_{j=0}^{k-2} L \cdot w(\mathbf{S}(B_j),u) + w(\mathbf{W}(B_j),u) \geq \frac{L}{16} \cdot b(u).$$

Since $i < L^2/1024$ and each block B_j for $j \leq k-2$ consists of $L/32$ important rounds, we have $k \leq \frac{L^2/1024}{L/32} = \frac{L}{32}$. Therefore, by an averaging argument there exists B_j for $j \leq k-2$ satisfying that:

$$L \cdot w(\mathbf{S}(B_j), u) + w(\mathbf{W}(B_j), u) \geq 2 \cdot b(u). \tag{4}$$

First observe that every strong neuron $s \in \mathbf{S}(B_j)$ fires for at least $L/2$ rounds in this block. The reason is that there is a gap of $L/2$ rounds between the last important rounds of B_j and the round where the inhibiting spike from x arrives. During this time interval every strong neuron in $\mathbf{S}(B_j)$ keeps on firing. Now, assume that u does not fire in any round of B_j, and denote the first important round of B_{j+1} by $\tau_{i'}$. Again, consider the algorithm that defines ℓ_1. Since u did not fire in any round of the block B_j, all the spikes that are fired towards u in B_j are in the residual set $\mathcal{R}_{\tau_{i'}}(u)$. Therefore by Eq. (4), we get that $w(\mathcal{R}_{\tau_{i'}}(u)) \geq (L/2) \cdot w(\mathbf{S}(B_j), u) + w(\mathbf{W}(B_j), u) \geq b(u)$, and u fires in $\tau_{i'}$. Therefore, we get that u fires either in some important round of B_j or in $\tau_{i'}$. In both cases there is a round $\tau_{i''}$ with $i'' \leq i$ such that $u \in A_{1,i''}$. This implies $u \in A'_{1,i''}$ for $i'' \leq i$, and the condition holds. ◀

Finally, it remains to prove Claim 29.

Proof of Claim 29. Recall that $\mathbf{S}(A_{0,i-1}(u))$ and $\mathbf{W}(A_{0,i-1}(u))$ are the strong and weak (respectively) incoming neighbors of u that fire in block T_{i-1}. If $w(\mathbf{S}(A_{0,i-1}), u) \geq b(u)/8$, then by Observation 28 the total spike weight fired in block $i-1$ is at least $L \cdot b(u)/8$, and we are done. Therefore, it remains to consider the case where $w(\mathbf{S}(A_{0,i-1}), u) < b(u)/8$ and $w(\mathbf{W}(A_{0,i-1})) < L \cdot b(u)/8$. We will show that in this case, there is a way to schedule all spikes fired towards u in block T_{i-1} to arrive in rounds $[(i-1)L+2, iL]$, such that u does not get activate in any of these rounds. By the definition of ℓ_0, we get that u does not get activated in any of the rounds $[(i-1)L+2, iL]$, and in particular $u \notin A'_{0,i}$, thus a contradiction.

First observe that $b(u) > 0$ since u did not fire in round $(i-1)L$ (as $u \notin A_{0,i-1}$) and it did not receive any negative spike in that round (as all negative spikes arrive in the last rounds of the blocks). We next show that all the spikes generated in block T_{i-1} can be scheduled in rounds $[(i-1)L+2, iL]$ without making u fire in any of these rounds. Since the scheduling algorithm of ℓ_0 works in this manner, we will get a contradiction to the fact that $u \in A_{0,i}$.

Scheduling spikes from weak neighbors. Let $F_{\mathbf{W}} = \{\langle v, u, (i-1)L\rangle \mid v \in \mathbf{W}(A_{0,i-1})\}$ be the spikes of weak neighbors fired in the block T_{i-1}. Recall that by Observation 28, these weak neurons fire only in the first round. Since these spikes are fired in round $(i-1)L$, they can arrive in any of the rounds $[(i-1)L+2, iL]$. As the total weight of the weak spikes is at most $Lb(u)/8$, we show that we can schedule them in a greedy manner into at most $L/2-2$ rounds while keeping the total weight in each such round to strictly less than $b(u)$. We traverse the weak spikes one by one, and start throwing them into rounds in $[(i-1)L+2, iL-1]$. We add a spike to round τ as long as the total weight of weak spikes scheduled to it is at most $b(u)/2$. If the addition of the next weak spike raises the weight to above $b(u)$ it is deferred to the next round $\tau+1$. Let τ' be the last round to which the weak spikes are schedules. Since in each $\tau \in [(i-1)L+2, \tau'-1]$ the total weight of weak spikes is at least $b(u)/2$, we get that $\tau' \leq (i-1)L + L/4 + 3 \leq (i-1)L + L/2$ as desired.

Scheduling spikes from strong neighbors. We next turn to show that also the strong spikes can be scheduled in a balanced manner in the remaining $L/2$ slots of the block T_i without activating the neuron u. Let $F_{\mathbf{S}} = \{\langle v, u, \tau\rangle \mid v \in \mathbf{S}(A_{0,i-1}), \tau \in T_{0,i-1})\}$ be the spikes of

strong neighbors fired in block T_{i-1}. For a spike $\langle v, u, \tau \rangle \in F_{\mathbf{S}}$ with $\tau \leq (i-1)L+(L/2-1)$, schedule $\langle v, u, \tau \rangle$ to arrive in round $\tau+L/2+1$. For $\langle v, u, \tau \rangle \in F_{\mathbf{S}}$ with $\tau \geq (i-1)L+L/2$, schedule $\langle v, u, \tau \rangle$ to arrive in round $\tau+1$. In this way, due to Observation 28, u receive two spikes from each $v \in \mathbf{W}(A_{0,i-1})$ in each round $\tau \in [(i-1)L + L/2 + 1, iL]$. Since $w(\mathbf{S}(A_{0,i-1})) < b(u)/8$, we get that the total weight of spikes that u receives in each of these rounds is less than $b(u)/4$, and therefore u does not get activated. Overall, all spikes generated in the block T_{i-1} are scheduled by ℓ_0 without activating the neuron u in any of the rounds $[(i-1)L+2, iL]$, contradiction to the fact that $u \in A_{i,0}$. The claim follows. ◁

B Missing Proofs for the Positive Results

B.1 Synchronization of Boolean Gates

Proof of Observation 13. The network is as follows: connect each input neuron x_i to the output neuron z by an edge of weight $w(x_i, z) = 1$, and let the bias of z be $b(z) = 1$. First note that if all input neurons x_i did not fire in round 0, then $\text{pot}(z, \tau) = -1$ for all τ, and z will not fire. If a neuron x_i fires in round τ, since the latency of each edge is at most L, there is a round $\tau' \in [\tau+1, \tau+L]$ in which the spike from x_i arrives to z. Thus, in round τ', the weighted incoming sum to z is at least 1, therefore $\text{pot}(z, \tau') \geq 0$ and z fires in round τ'. ◀

The Complete Proof of Lemma 14. We analyze the correctness of the network $\mathsf{NOT}_{\mathsf{sync}}$. We begin by proving the following auxiliary claim.

▷ Claim 30. If all the intermediate neurons $v_0, \ldots v_L$ fire starting round τ for at least $L(L+1)$ rounds, then there exists a round $\tau' \in [\tau+1, \tau+L(L+1)]$ in which the output neuron z fires (i.e., regardless of the latencies of the edges).

Proof. For every $i \in \{0, 1 \ldots, L(L+1)\}$, denote the L-length interval $T_i = [\tau + i \cdot L, \tau + (i+1)L - 1]$. In addition, define $\widetilde{T} = T_0 \cup \ldots \cup T_{L+1}$. Let q_i be the number of spikes that were fired in the interval of T_i but received by z in the next interval T_{i+1}. Note that since the maximum edge latency is L, in the worst case the spikes of interval T_i must arrive to z by the end of the next interval T_{i+1}. We next prove by induction on i that either z fires by the end of the interval T_i, or $q_{i+1} \geq i \cdot L$. For the base of the induction, consider $i = 0$. If z did not fire in some round during T_0, we claim that $q_1 \geq L$. Since all the $L+1$ neurons fire in every round during the interval, overall $L(L+1)$ many spikes where fired. By the fact that z did not fire during T_0, we have that in each of these rounds, it received at most L spikes. This implies that z received at most L^2 many spikes during T_0, and therefore at least $q_1 \geq L$ many spikes will be received by z in the interval T_1. Assume that the claim holds up to $i-1$ and consider the i^{th} interval. If z fired by the end of the i^{th} interval T_i, we are done. Otherwise, by induction assumption for $i-1$, we have that $q_i \geq i \cdot L$. In addition, all these q_i spikes must be received at z during the interval T_i. Then, in interval T_i we again have a total of $L(L+1)$ fresh spikes by the neurons $v_0, \ldots v_L$. This creates a total of $L(L+1) + i \cdot L$ spikes. As z did not fire in T_i, it received at most L^2 many spikes, leaving at least $q_{i+1} \geq L(L+1) + i \cdot L - L^2 \geq (i+1)L$ spikes for the next interval. This completes the proof of the induction step.

Overall, for $i = L$, we have that either z fired by the end of the interval T_L, or that $q_{L+1} \geq L(L+1)$. In the latter case, since all these spikes must arrive during the last interval T_{L+1}, by the pigeonhole principle there must be a round in this interval in which z received at least $L+1$ spikes and fire. This completes the proof of the claim. ◁

Proof of Lemma 14. Due to Claim 30, it remains to show that if x did not fire in round 0, then there must be a starting round τ, in which all the neuron $v_0, \dots v_L$ fire for at least $L(L+1)$ rounds.

If x did not fire, then neither the memory neuron m nor the reset neuron r fire during the execution. Since we assume that v^* fires in round 0, it must hold that all these neurons fire starting some round $\tau \in [5L^2, 5L^3 + 2L]$. This holds due to the self-loops on the neurons $v_0, \dots v_L$, and the chain of length $5L^2$. By Claim 30, there exists a round $\tau' \in [\tau + 1, \tau + L(L+1)]$ in which z fires.

We next show that if x fired in round 0, then z would not fire in any round. The key observation is as follows:

▶ **Observation 31.** *In order for z to fire in some round τ, it must receive spikes from* at least *two different v_i neurons.*

Proof. To see this, note that since the maximum edge latency is L, in round τ, z can receive spikes only from the L previous rounds $\tau - L, \dots, \tau - 1$. In particular, a single neuron can be accounted for at most L many spikes received by z in a given round. Finally, since the bias of z is $L + 1$, and all edge weights are 1, we conclude that z must receive spikes from at least two neurons in order to fire. ◀

When x fires in round 0, the memory neuron m fires from round $\tau_m \in [1, L+1]$ ahead, due to its self-loop. Hence, starting round $\tau_r \in [2, 2L+2]$, the reset neuron r starts firing at least *once* in every interval of $2L$ rounds. Recall that each v_i gets a negative spike from the inhibitor r and positive spike from the neuron $c_{5iL} \in C$. We next show that each neuron v_i gets inhibited at least L rounds *before* the activation of the neurons v_{i+1}. As a result, at any point of time, there will be no two neurons v_i and v_j such that z received both of their spikes in the same round. By induction on i, the first intermediate neuron v_0 has an incoming edge from $c_0 = v^*$, and thus it begins to fire in some round $\tau' \in [0, L]$. Due to the negative edge from the reset neuron r, it stops firing before round $3L + 2$. Since v_1 has an incoming edge from c_{5L}, it starts firing only after round $5L + 1$, and therefore z starts receiving spikes from v_1 only starting round $5L + 2$. Assume the claim is correct for neurons $v_0, \dots, v_{i-1}$ and consider neuron v_i. If the neuron v_i starts firing in round τ_i, by round $\tau_i + 2L$ it is inhibited by r. Because v_i starts to fire after receiving a spike from $c_{5 \cdot i \cdot L}$ and v_{i+1} starts firing after receiving a spike from $c_{5(i+1)L}$, neuron v_{i+1} begins to fire only after round $\tau_i + 4L$, at least L rounds after v_i is inhibited.

Hence, z cannot receive input from two different neurons v_i, v_j at the same round, and the claim follows by combining Observation 31. Finally, the next observation plays a rule in the subsequent constructions.

▶ **Observation 32.** *The correctness still holds even if the chain starts to fire at some round $\tau > 0$. In this case the output neuron z fires in some round $t \in [\tau + 1, \tau + \Theta(L^3)]$.*

Proof of Observation 32. The correctness of the observation follows from the fact that the input neuron x activates the memory neuron m, that keeps on firing (i.e., presenting the state of x) due to its self-loop. Thus all arguments in Lemma 14 still hold in case the chain starts to fire in any later round. ◀

B.2 Synchronization of a Boolean Circuit, Proof of Lemma 15

The Construction. Given a Boolean circuit $\mathcal{A}$ of OR / NOT gates $g_1, \ldots, g_m$ of depth d, we describe a construction of an analogous neural network $\mathcal{N}$ with a similar execution. For every g_i, let $\mathsf{Sync}(g_i)$ be the synchronized sub-network of the gate g_i. Specifically, for a NOT gate (resp., OR) g_i, the sub-network $\mathsf{Sync}(g_i)$ is taken from Lemma 14 (resp., Observation 13). Recall, that for a NOT gate g_i, its syncronized sub-network $\mathsf{Sync}(g_i)$ contains a chain of neurons where the head of the chain c_0 will be denoted hereafter by v_i^*. The network $\mathcal{N}$ consists the following components:

1. Input neurons $x_1, \ldots, x_n$, and output neurons $z_1, \ldots, z_k$, that serve as the input and the output for the network $\mathcal{N}$.
2. A chain $C = [c_0, \ldots, c_q]$ containing $q + 1 = \alpha d L^3 + 1$ neurons, where α is a constant satisfying that $\alpha L^3 \geq 5L^3 + L$. For every $i \geq 0$, the neuron c_i has bias $b(c_i) = 1$. Moreover, for every $i \geq$ the neuron c_i has a positive incoming edge from c_{i-1} with weight $w(c_{i-1}, c_i) = 1$. Our simulation starts with neuron c_0 firing.
3. A $\mathsf{Sync}(g_i)$ network (using Lemma 14 and Observation 13 respectively) for every gate g_i.

The connections between these components are as follows:

1. For every gate g_i in the first layer, the input for its synchronized sub-network $\mathsf{Sync}(g_i)$ is given by $x_{i,1}, \ldots, x_{i,k_i}$, namely, the input bits of the gate g_i in the circuit $\mathcal{A}$.
2. For every gate g_i in layer $j \geq 2$, denote by $g_{i,1}, \ldots, g_{i,k_i}$ the input of the gate g_i in the circuit $\mathcal{A}$. In the network $\mathcal{N}$, the input to the sub-network $\mathsf{Sync}(g_i)$ are the output neurons of the sub-networks $\mathsf{Sync}(g_i)$, $\mathsf{Sync}(g_{i_1}), \ldots, \mathsf{Sync}(g_{i_k})$.
3. The output gates of the network $\mathcal{N}$ are the output neurons of the sub-networks $\mathsf{Sync}(o_1), \ldots, \mathsf{Sync}(o_k)$, where $o_1, \ldots, o_k$ are the output gates of the circuit $\mathcal{A}$.
4. Finally, the synchronized sub-networks of the NOT gates are connected to the chain C as follows. For each NOT gate g_i in every layer j, the $(j\alpha L^3)^{th}$ neuron $c_{j\alpha L^3}$ in the chain has an outgoing edge to v_i^* with weight 1 (where v_i^* is the head of the internal chain in $\mathsf{Sync}(g_i)$), since the bias of v_i^* is 1, a spike from $c_{j\alpha L^3}$ makes v_i^* fire.

Figure 3 illustrates the construction for a circuit with 4 NOT and OR gates of depth 3. We note that one can shave an L-factor in the size and time overhead of lemma 15, by reusing the synchronization chain for all the Boolean gates in the network.

Correctness. Let V be the total set of neurons in $\mathcal{N}$, and let $\ell : V \times V \times \mathbb{N} \to [1, L]$ be a fixed (arbitrary) nice latency function. First note that in the global chain C, each of the neurons fires once, in a sequential manner. Recall, that we assume that the starter c_0 fires in round $\tau_0 = 0$. For every $j \in \{1, \ldots, d\}$, let τ_j be the round in which $c_{j\alpha L^3}$ fires (i.e., the spike from $c_{j\alpha L^3 - 1}$ is received at $c_{j\alpha L^3}$ in round $\tau_j - 1$).

For every gate g_i in the circuit $\mathcal{A}$ in layer $j \geq 1$, denote by $out(g_i, \mathcal{A})$ the final state of g_i after receiving its inputs in the circuit $\mathcal{A}$. In addition, let q_i be the output neuron in the sub-network $\mathsf{Sync}(g_i)$, and let $\sigma_t(q_i, \mathcal{N})$ be the state of the neuron in round t when simulating the network $\mathcal{N}$. Our goal is to show that for every g_i, its corresponding output q_i in the network $\mathcal{N}$, has the same "output" as g_i in the circuit $\mathcal{A}$.

▷ Claim 33. For every layer $j \in \{1, \ldots, d\}$ and every gate g_i in layer j of circuit $\mathcal{A}$, its holds that: (i) If $out(g_i, \mathcal{A}) = 0$, then $\sigma_t(q_i, \mathcal{N}) = 0$ for every t, and (ii) If $out(g_i, \mathcal{A}) = 1$, then there exists $t \in [\tau_{j-1} + 1, \tau_j]$ such that $\sigma_t(q_i, \mathcal{N}) = 1$.

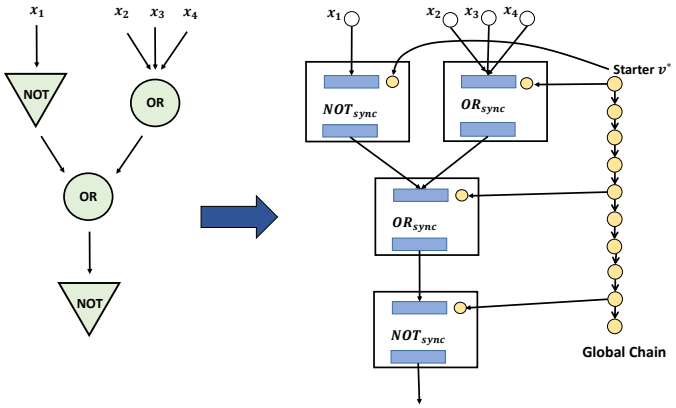


Figure 3 The transformation of the circuit on the left with 4 inputs and 3 layers. For each gate we add the corresponding synchronized sub-network, where we connect the input and output neurons of the sub-network according to the original circuit. In addition we introduce a global chain that activates the sub-networks in each layer after the previous layers have already finished the computation. The first neuron in the global chain is set to be the starter neuron which fires in the beginning of the simulation.

Proof. We prove by induction on the layer j. For $j = 1$, recall that the input neurons $x_{i,1}, \ldots, x_{i,k_i}$ of the sub-network $\mathsf{Sync}(g_i)$ are the input neurons of the network $\mathcal{N}$. Therefore, in round 0 in the simulation of $\mathcal{N}$, the sub-network $\mathsf{Sync}(g_i)$ has the same input as gate g_i in the circuit $\mathcal{A}$. Assume first that g_i is a NOT gate. Then the spike of the starter neuron c_0 arrived at the head chain v_i^* by round L. Combining with Observation 32 we get that if $out(g_i, \mathcal{A}) = 1$ then $\sigma_t(q_i, \mathcal{N}) = 1$ for some $t \in [L, L + 5L^3] \subseteq [1, \alpha L^3]$. In addition, if $out(g_i, \mathcal{A}) = 0$ then $\sigma_t(q_i, \mathcal{N}) = 0$ for every t. The case where g_i is an OR gate is even simpler and follows by Observation 13. Since the path from c_0 to $c_{\alpha L^3}$ in the chain C is of length αL^3, we have that $\tau_1 \geq \alpha L^3$. Therefore $[1, \alpha L^3] \subseteq [\tau_0 + 1, \tau_1]$, and the claim holds for $j = 1$.

For the induction step, let $j \geq 2$, and assume correctness up to layer $j - 1$. We now prove the claim for layer j. Let g_i be a gate in layer j. By Observation 32, the important thing to take care of regarding a NOT gate g is to make sure that its inputs have the correct states (i.e., as the corresponding states in $\mathcal{A}$) by the time that the head of the chain v_i^* in $\mathsf{Sync}(g)$ has received the spike from $c_{(j-1)\alpha L^3}$. Denote by $q_{i,1}, \ldots, q_{i,k_i}$ the output neurons of the sub-networks $\mathsf{Sync}(g_{i,1}), \ldots, \mathsf{Sync}(g_{i,k_i})$. By the induction assumption, for each $g_{i,h}$, if $out(g_{i,h}, \mathcal{A}) = 0$ then $\sigma_t(q_{i,h}) = 0$ for every t, and otherwise $\sigma_t(q_{i,h}) = 1$ for some $t \leq \tau_{j-1}$. Since the neurons $q_{i,1}, \ldots, q_{i,k_i}$ are the input neurons of g, it holds that the sub-network $\mathsf{Sync}(g)$ gets the same input as the input of g in $\mathcal{A}$, by round τ_{j-1} of the simulation of $\mathcal{N}$.

Now, assume that g_i is a NOT gate. Then, by round $\tau_{j-1} + L$, the head of the chain v_i^* has recieved the spike from $c_{(j-1)\cdot\alpha\cdot L^3}$. Combining with Observation 32, when $out(q_i, \mathcal{N}) = 1$, it holds that $\sigma_t(q_i, \mathcal{N}) = 1$ for some $t \in [\tau_{j-1} + 1, \tau_{j-1} + L + 5L^3]$. In addition, when $out(q_i, \mathcal{N}) = 0$ then $\sigma_t(q_i, \mathcal{N}) = 0$ for every t. Again, since the path from $c_{(j-1)\alpha L^3}$ to $c_{j\alpha L^3}$ in the chain C is of length αL^3, we have that $\tau_j \geq \tau_{j-1} + L + 5L^3$, and the claim follows. The case where g_i is an OR gate follows in a similar way by Observation 13. ◁

Lemma 15 follows by using Claim 33 with $j = d$, and noting that each output neuron z_i in $\mathcal{N}$ is the output neuron $q_{i'}$ for some sub-network $\mathsf{Sync}(g_{i'})$ where $g_{i'}$ is a gate in layer d of $\mathcal{A}$. This completes the correctness and the bound on the time overhead. We finally bound the

size of the network. The network $\mathcal{N}$ consists of a chain of $O(dL^3)$ neurons, and a $\mathsf{Sync}(g_i)$ sub-network of size $O(L^2)$ for each gate g_i in $\mathcal{A}$. Therefore, there are overall $O(dL^3 + mL^2)$ auxiliary neurons.

B.3 Synchronization of a Single Deterministic Threshold Gate

We now turn to consider the synchronized implementation of a single deterministic threshold gate and prove Lemma 16.

Thanks to a result of [30], we can assume without loss of generality that the weights and bias values can be represented using binary vectors of length $\lceil \Delta \log \Delta \rceil$. Hastad [10] also showed that this bound is tight. In addition, we can also assume without loss of generality that $b(z) \geq 0$. The key part is to implement the single threshold gate by a Boolean circuit. This requires small adaptations from existing results in the area, specifically we will use the following known facts.

▶ **Fact 34** (Iterated Addition [31, 38]). *Given two input binary vectors $\bar{x} = [x_1, \ldots, x_\Delta]$ and $\bar{y} = [y_1, \ldots, y_\Delta]$, there exists a Boolean circuit with $\mathrm{poly}(\Delta)$ gates and $O(1)$ depth that outputs the binary representation of $\mathrm{dec}(\bar{x}) + \mathrm{dec}(\bar{y})$.*

▶ **Corollary 35** (Multiple Iterated Addition). *Given Δ input binary vectors $\bar{x}_1, \ldots, \bar{x}_\Delta$ where $\bar{x}_i \in \{0,1\}^m$ for some integer $m \geq 1$, there exists a Boolean circuit with $\mathrm{poly}(\Delta, m)$ gates and $O(\log \Delta)$ depth that outputs the binary representation of $\sum_i \mathrm{dec}(\bar{x}_i)$.*

▶ **Observation 36** (Comparison). *Given two input binary vectors $\bar{x} = [x_1, \ldots, x_\Delta]$ and $\bar{y} = [y_1, \ldots, y_\Delta]$, there exists a Boolean circuit with $\mathrm{poly}(\Delta)$ gates and $O(1)$ depth that outputs 1 iff $\mathrm{dec}(\bar{x}) \geq \mathrm{dec}(\bar{y})$.*

We are now ready to implement a threshold gate by a small depth Boolean circuit of polynomial size. This lemma explains the dependency in the largest in-degree Δ of the final synchronized solution.

▶ **Lemma 37.** *Given a threshold gate g with Boolean inputs $x_1, \ldots, x_\Delta$ with weights $w_1, \ldots, w_\Delta$, and an output neuron z with bias $b(z)$, there exists a Boolean circuit that computes g (i.e., outputs 1 iff $\sum w_i \cdot x_i \geq b(z)$) using $\mathrm{poly}(\Delta)$ gates and depth $O(\log \Delta)$.*

Proof. Each input x_i is connected to $\ell = \lceil \Delta \cdot \log \Delta \rceil$ neurons $w_{i,1}, \ldots, w_{i,\ell}$, where the edge weight $w(x_i, w_{i,j})$ is 1 if the j^{th}-bit in w_i is 1 and 0 otherwise. We set the bias values to be $b(w_{i,j}) = 1$. Thus, the outgoing edge weights of x_i encode the binary representation of the weight w_i. As a result, once x_i fires in round τ, after at most L rounds, $w_{i,j}$ fires iff the j^{th} bit in the representation of w_i is 1. As we will see, those $\Delta^2 \cdot \log \Delta$ neurons $w_{1,1}, \ldots, w_{\Delta,\ell}$ will serve as the input layer to the circuit. In addition, we also represent the bias of z using ℓ neurons $b_1, \ldots, b_\ell$ that encode the binary representation of $b(z)$: the bias of $b_j = 1$ if the j^{th} bit in $b(z)$ is 0, and $b_j = -1$ otherwise. Let $\bar{x}_{pos} = \{x_i \mid w_i \geq 0\}$ and $\bar{x}_{neg} = \{x_i \mid w_i < 0\}$. In the same manner, let $W_{pos} = \sum\{w_i \mid w_i \geq 0\}$ and $W_{neg} = \sum\{|w_i| \mid w_i < 0\}$. We will use the Multiple Iterated Addition circuit of Corollary 35 to compute the binary representation of W_{pos} and $W_{neg} + b(z)$. Finally, we use the Comparison circuit of Observation 36 to compare those values, such that the output will be 1 iff $W_{pos} \geq W_{neg} + b(z)$, hence computing the function of the threshold gate. ◀

The final synchronous implementation of g is obtained by applying Lemma 15 on $\mathcal{C}$, i.e., $\mathsf{Sync}(g) \leftarrow \mathsf{Sync}(\mathcal{C})$. The construction uses a total $O(\log \Delta \cdot L^3 + \mathrm{poly}(\Delta) \cdot L^2)$ auxiliary neurons, and computation time of $O(L^4 \cdot \log \Delta)$ rounds. This completes the proof of Lemma 16.

B.4 Probabilistic Threshold Gate

B.4.1 Description of the Boolean Circuit

The construction of the boolean circuit $\mathcal{A}$ approximating a probabilistic threshold gate is achieved using two main steps. First we sample an almost uniform random variable, then we use the sampled value in order to approximate a sample from the Logistic distribution.

Step 1: Sampling from the Almost Uniform Distribution. We introduce $k = 4\log(1/\epsilon)$ uniformly random gates, denoted as $r_1, \dots r_k$. Hence, $\mathrm{dec}(\bar{r})$ encodes an integer number that is uniformly sampled between 0 and $\lceil (1/\epsilon)^4 \rceil$. In addition, we introduce k input bits (with fixed value) $a_1, \dots a_k$ such that $\mathrm{dec}(\bar{a}) = \lceil (1/\epsilon)^4 \rceil$. Thus, the value $r' = \mathrm{dec}(\bar{r})/\mathrm{dec}(\bar{a})$ is sampled uniformly at random from the set $\{0, \epsilon^4, 2\epsilon^4, 3\epsilon^4, \dots 1\}$.

Step 2: Sampling from the Almost Logistic Distribution. Next, we transform the sample r' from Step 1 into a sample from an almost Logistic distribution. This is done by using the method of inverse transform sampling. In our context, for a sample r u.a.r in $[0, 1]$, the value $b + \ln(r/(1-r))$ is a sample from the Logistic distribution with mean b and scale 1. To compute the expression $b + \ln(r'/(1-r'))$ using a Boolean circuit, we approximate the $\ln(x)$ function (up to $\pm\,\mathrm{poly}(\epsilon)$) using the first $O(\log 1/\epsilon)$ terms of the Taylor expansion around a point x_0 where $0 \le x_0 - x \le 1/2$.

▶ **Definition 38** (ϵ-Approximation of the $\ln(x)$ Function). *Given $x > 0$ and a positive integer k, let $\widehat{\ln}_k(x)$ be the* ln*-approximation of x obtained by computing the first k terms of the Taylor expansion around a point x_0, where $0 \le x_0 - x \le 1/2$. When k is clear from the content we may omit it and simply write $\widehat{\ln}(x)$.*

The task of sampling from the (almost) Logistic distribution then boils into computing $f(r') = \widehat{\ln}_k(r'/(1-r'))$ with $k = \lceil 4\log 1/\epsilon \rceil$. We first use a Boolean circuit to distinguish between the case where $r' \le 1 - r'$, and the complementary case. Using the vectors $\bar{r}$ and $\bar{a}$, this can be done using integer operations and comparison as $r'/(1-r') = \mathrm{dec}(\bar{r})/(\mathrm{dec}(\bar{a}) - \mathrm{dec}(\bar{r}))$. When $r' > 1 - r'$, we calculate $f(-r')$, and then either add or subtracts it from the bias b respectively.

In what follows, assume that $r' \le 1 - r'$, and therefore $r'/(1-r') \in [0, 1]$. To pick the point x_0 around which the Taylor approximation is expended, we let $x_0 = 1/2$ when $r'/(1-r') \le 1/2$, and $x_0 = 1$ otherwise. This latter condition can also be easily checked with a Boolean circuit.

To finally be able to compute the function $\widehat{\ln}_k(x)$ using a Boolean circuit, we must ensure that all our operations are applied on *integers*. Therefore, instead of computing $f(r')$, we will be actually computing $q \cdot f(r')$ for some large enough constant q that guarantees that $q \cdot f(r')$ is an integer. Specifically, letting $q = (\mathrm{dec}(\bar{a}) - \mathrm{dec}(\bar{r}))^k$ does the job as the function $\widehat{\ln}_k(x)$ is a polynomial of degree k. This factor of q would not affect the correctness of the computation as it will be canceled out later on. Using the circuit for iterated addition [38, 31] and fast multipliers [9], we can compute $q \cdot f(r')$ using only integer addition and multiplication. The output of the final Boolean circuit is $\bar{y}$ where $\mathrm{dec}(\bar{y}) = q \cdot b + q \cdot f(r')$. In the analysis section, we show that $\mathrm{dec}(\bar{y})/q$ is sampled from a distribution that is $\mathrm{poly}(\epsilon)$-close to the Logistic distribution with mean b.

Putting all Together: The Output Circuit. Let $w_1, \ldots, w_\Delta$ be the weights of the probabilistic threshold gate g. To cancel out the multiplication of q in the output bias value from the previous step, we multiply all the incoming weights by q as well. We can then use the construction of Lemma 16 for a deterministic threshold gate with weights $w'_1 = q \cdot w_1, \ldots, w'_\Delta = q \cdot w_\Delta$ and bias $b'' = \mathrm{dec}(\bar{y})$. This completes the description of the construction.

B.4.2 Analysis and Proof of Lemma 17

We now turn to prove Lemma 17 and start with several auxiliary claims.

▷ Claim 39. Let $r_1, r_2 \in [0,1]$ such that $|r_1 - r_2| \leq \epsilon^2$ and $\epsilon \leq r_1 \leq 1 - \epsilon$, then

$$|\ln(r_1/(1-r_1)) - \ln(r_2/(1-r_2))| \leq 2\epsilon \ .$$

Proof. By the definition of r_1 and r_2 we get the following inequalities:

$$\begin{aligned} |\ln(r_1/(1-r_1)) - \ln(r_2/(1-r_2))| &= |\ln(r_1) - \ln(1-r_1) - \ln(r_2) + \ln(1-r_2)| \\ &\leq |\ln(\frac{r_1+\epsilon^2}{r_1})| + |\ln(\frac{1-r_1+\epsilon^2}{1-r_1})| \\ &\leq |\ln(1+\epsilon^2/r_1)| + |\ln(1+\epsilon^2/(1-r_1))| \\ &\leq 2\ln(1+\epsilon) \leq 2\cdot\epsilon \ , \end{aligned}$$

where the last inequality is due to the Taylor expansion of $\ln(1+x)$ around 0. ◁

Recall that given $x > 0$ and an integer $k > 0$, $\widehat{\ln}_k(x)$ is the ln-approximation of x obtained by computing the first k terms of the Taylor expansion around a point x_0 where $0 \leq x_0 - x \leq 1/2$.

▷ Claim 40. Fix $r_1, r_2 \in [0,1]$ such that $|r_1 - r_2| \leq \epsilon^2$ and $\epsilon \leq r_1 \leq 1 - \epsilon$, denote $\widehat{b}_1 = b + \ln(r_1/(1-r_1))$ and $\widehat{b}_2 = b + \widehat{\ln}(r_2/(1-r_2))$. Then, $|\widehat{b}_1 - \widehat{b}_2| \leq 3\epsilon$.

Proof. Fix $x \in (0,1)$. Since $\widehat{\ln}(x)$ is obtained by using the first k terms in the Taylor expansion of $\ln(x)$ around x_0, we have that $|\ln(x) - \widehat{\ln}(x)| = \frac{1}{x_0^k} \cdot \frac{(x-x_0)^k}{k} \cdot \eta^k$, where $\eta \in [x, x_0]$. Since $x_0 \geq x$, also $x_0 \geq \eta$. As $|x - x_0| \leq 1/2$, we get that $|\ln(x) - \widehat{\ln}(x)| \leq (1/2)^k$. By plugging $k = \Theta(\log 1/\epsilon)$, we have that $|\ln(x) - \widehat{\ln}(x)| \leq \epsilon$ for every x.

Thus, combining with Claim 39 we conclude the following:

$$\begin{aligned} |\widehat{b}_1 - \widehat{b}_2| &= |b + \ln(r_1/(1-r_1)) - b - \widehat{\ln}(r_2/(1-r_2))| \\ &\leq |\ln(r_1/(1-r_1)) - \ln(r_2/(1-r_2)) + \epsilon| \\ &\leq \epsilon + |\ln(r_1/(1-r_1)) - \ln(r_2/(1-r_2))| \leq 3\cdot\epsilon \ . \end{aligned} \tag{5}$$

◁

▷ Claim 41. Consider two threshold gates g_1, g_2 with the same weighted sum and bias values $b_1 \leq b_2$ such that $b_2 - b_1 \leq \epsilon$. Then $|\Pr[g_1 = 1] - \Pr[g_2 = 1]| \leq \sqrt{\epsilon}$.

Proof. Let W be the weighted incoming sum to both g_1 and g_2. The probability that g_1 outputs 1 is $1/(1+e^{-(W-b_1)})$, and the probability that g_2 outputs 1 is $1/(1+e^{-(W-b_2)})$. The following holds:

$$\begin{aligned} \Pr[g_2 = 1] &= 1/(1+e^{-(W-b_2)}) \geq 1/(1+e^{-(W-b_1-\epsilon)}) = 1/(1+e^{\epsilon}e^{-(W-b_1)}) \\ &\geq \frac{1}{e^{\epsilon}\cdot(1+e^{-(W-b_1)})} \geq \frac{1}{(1+\sqrt{\epsilon})\cdot(1+e^{-(W-b_1)})} \\ &= (1-\sqrt{\epsilon})(1/(1+e^{-(W-b_1)})) \geq 1/(1+e^{-(W-b_1)}) - \sqrt{\epsilon} = \Pr[g_1 = 1] - \sqrt{\epsilon} \ . \end{aligned}$$

In the third inequality we use the fact that $e < (1+\sqrt{\epsilon})^{\frac{1}{\epsilon}}$ and thus $e^{\epsilon} < 1+\sqrt{\epsilon}$. On the other hand, since $b_2 \geq b_1$ it holds that

$$\Pr[g_2 = 1] = 1/(1+e^{-(W-b_2)}) \leq 1/(1+e^{-(W-b_1)}) = \Pr[g_1 = 1] \ .$$

Hence, we conclude that $|\Pr[g_2 = 1] - \Pr[g_1 = 1]| \leq \sqrt{\epsilon}$ as required. ◁

Analysis of Step 1. In the first step of the construction, since each uniformly random gate r_i is 1 with probability 1/2, the value $\mathrm{dec}(\bar{r})$ is a uniform sample in $\{0, 1 \ldots, (1/\epsilon)^4\}$. Therefore, $r' = \mathrm{dec}(\bar{r})/\mathrm{dec}(\bar{a}) = \epsilon^4 \cdot \mathrm{dec}(\bar{r})$ is sampled uniformly at random from $\{0, \epsilon^4, 2\epsilon^4, 3\epsilon^4, \ldots 1\}$. By a simple coupling argument, sampling r' is equivalent to the process of sampling a uniform random variable $r_1 \in [0,1]$ and rounding it to the closest value of the form $i \cdot \epsilon^4$ for some integer i. In this manner, these two samples have an additive distance of at most ϵ^4.

Analysis of Step 2. Denote the probability z outputs 1 by q, and the probability u outputs 1 by p. Recall that g is the probabilistic gate and g' is the output gate of the Boolean circuit that approximates g.

In the second step, we compute $\mathrm{dec}(\bar{y}) = q \cdot (b + f(r'))$ where $q = (\mathrm{dec}(\bar{a}) - \mathrm{dec}(\bar{r}))^k$ and $f(r') = \widehat{\ln}(r'/(1-r'))$. Then g' outputs 1 iff $\mathrm{dec}(\bar{y}) \leq W \cdot q$, or simply iff $b + f(r') \leq W$. Given that $r' \in [2\epsilon^2, 1-2\epsilon^2]$ by Claim 40, $b' = b + f(r')$ satisfies that $|b^* - b'| \leq 3\epsilon^2$ where b^* is a true sample from the Logistic distribution with mean b. Therefore, the following holds.

$$\begin{aligned}
\Pr[g' = 1 \ \mid \ r' \in [2\epsilon^2, 1-2\epsilon^2]] &= \Pr[W \geq b' \mid r' \in [2\epsilon^2, 1-2\epsilon^2]] \\
&\leq \Pr[W + 3\epsilon^2 \geq b^* \mid r' \in [2\epsilon^2, 1-2\epsilon^2]] \\
&= 1/(1+e^{-(W-b+3\epsilon^2)}),
\end{aligned}$$

and in addition

$$\Pr[g' = 1] \geq \Pr[W - 3\epsilon^2 \geq b^* \mid r' \in [2\epsilon^2, 1-2\epsilon^2]] = 1/(1+e^{-(W-b-3\epsilon^2)}).$$

Recall that $\Pr[g=1] = \frac{1}{1+e^{-(W-b)}}$. By claim 41 we conclude that $|\Pr[g'=1] - \Pr[g=1]| \leq 3\epsilon$.

We note that $r' \in [2\epsilon^2, 1-2\epsilon^2]$ with probability at least $1-4\epsilon^2$. Hence, we conclude that:

$$\Pr[g' = 1] \leq \Pr[g' = 1 \ \mid \ r' \in [2\epsilon^2, 1-2\epsilon^2]] + 4\epsilon^2 \leq \Pr[g=1] + 3\epsilon + 4\epsilon^2 = p + \Theta(\epsilon) \ ,$$

and on the other hand:

$$\begin{aligned}
\Pr[g' = 1] &\geq (1-4\epsilon^2)\Pr[g' = 1 \ \mid \ r' \in [2\epsilon^2, 1-2\epsilon^2]] \\
&\geq (1-4\epsilon^2)(\Pr[g=1] - 3\epsilon) \geq \Pr[g=1] - \Theta(\epsilon) \ .
\end{aligned}$$

Thus, $|\Pr[g=1] - \Pr[g'=1]| = O(\epsilon)$ as required.

Complexity. We assume that the bias and weights of the given probabilistic threshold gate g are polynomial in $1/\epsilon$. We first claim that with high probability of $1 - \Theta(\epsilon)$, the approximate bias sampled from the almost Logistic distribution in also bounded by $\mathrm{poly}(1/\epsilon)$.

▷ Claim 42. Given that $|\mu| = \mathrm{poly}(1/\epsilon)$, for a random variable x drown from the logistic distribution with mean μ it holds that $|x| = \mathrm{poly}(1/\epsilon)$ with probability greater than $1-\epsilon$.

Proof. By the definition of the Logistic CNF function it holds that

$$\Pr[x > 2\ln(1/\epsilon) + \mu] = 1 - \frac{1}{1+e^{-2\ln(1/\epsilon)-\mu+\mu}} = \frac{\epsilon^2}{1+\epsilon^2} < \epsilon^2 \ .$$

On the other hand

$$\Pr[x \leq -2\ln(1/\epsilon) + \mu] = \frac{1}{1 + e^{2\ln(1/\epsilon) - \mu + \mu}} = \frac{1}{1 + 1/\epsilon^2} < \epsilon^2 \ . \qquad \lhd$$

Thus we can assume from now on that all integer numbers can be representing using $O(\text{poly}(1/\epsilon))$ bits. Using circuits for fast integers multiplication as described in [9] and iterated addition [38, 31], there exists a Boolean circuit computing $W \cdot q$ as well as $b \cdot q$ using $\text{poly}(\Delta, \log(1/\epsilon))$ gates and $\text{poly}(\log \Delta, \log(1/\epsilon))$ depth. When computing the polynomial $q \cdot \widehat{\ln}(\frac{r'}{1-r'})$ (of total degree $2k$), calculating each term requires $O(\log k)$ multiplicity operations. Since we have k summands, in total we use $k \cdot \log k$ multiplicity operations, each requires $O(k \cdot \log k \cdot 2^{O(\log^* k)})$ gates (and depth), and $\log k$ addition operations. The comparison circuits uses $\text{poly}(\log 1/\epsilon)$ gates and depth, and the final threshold gate circuit requires $\text{poly}(\Delta, \log 1/\epsilon)$ gates and depth $\text{poly}(\log \Delta, \log 1/\epsilon)$. We conclude that the Boolean circuit has $\text{poly}(\Delta, \log(1/\epsilon))$ gates and depth of $\text{poly}(\log \Delta, \log(1/\epsilon))$.

B.4.3 Synchronizing a Probabilistic Threshold Gate

In order to construct a synchronized neural network computing the Boolean Circuit described in Lemma 17, we use the construction for synchronized Boolean circuits as described in Lemma 15. We are left with describing the implementation of the random bits $\bar{r}$ and the constant bits $\bar{a}$.

- In order to represent $\bar{a}$, we introduce k neurons $a_1, \ldots, a_k$. If the i^{th} bit in the binary representation of $\text{dec}(\bar{a}) = (1/\epsilon)^4$ equals 1 we set the bias of a_i to be $b(a_i) = -1$ and otherwise we set the bias to be $b(a_i) = 1$. As a result, the neurons that represent the bits that are 1 in the binary representation fire on every round, and the other neurons idle thought the execution.
- In order to represent $\bar{r}$ we introduce k *spiking* neurons $r_1, \ldots, r_k$. For the computation to succeed, we need to sample each random variable r_i only once. Therefore, each neuron r_i has a very large bias $b(r_i) = \text{poly}(1/\epsilon)$ and an incoming edge from the starter neuron s with weight $w(s, r_i) = b(r_i)$. As a result, as long as r_i did not receive a spike from s, with high probability it does not fire. On the other hand, when neuron r_i receives a spike from the starter neuron v^* it fires with probability $1/2$.

B.5 The Complete Synchronization Scheme

Finally, we describe the synchronizer for a given neural network and prove Theorem 4. We start by describing the construction for a network of deterministic threshold gates. The adaptation to a network of spiking neurons is quite straightforward as discussed in the end of the section. The construction has two parts: a global pulse generator that can be used to synchronize many networks, and an adaptation of the given network $\mathcal{N}$ into a network $\mathsf{Sync}(\mathcal{N})$, see Figure 2. The *pulse generator* is implemented by a directed cycle $PG = [c_0, \ldots, c_k]$ of length $k = O(L^4 \log \Delta)$. All neurons in PG have bias $b(c_i) = 1$. In addition, for every $i \geq 1$ neuron c_i has an incoming edge from neuron c_{i-1} with weight $w(c_{i-1}, c_i) = 1$, and the first neuron c_0 has an incoming edge from the last neuron c_k with weight $w(c_0, c_k) = 1$. The last neuron of the chain c_k will declare the end of each phase. We assume throughout that the simulation starts by a spike of the starter $v^* = c_0$.

Modifications to the Network $\mathsf{Sync}(\mathcal{N})$. The input layer and output layer in $\mathsf{Sync}(\mathcal{N})$ are *exactly* as in $\mathcal{N}$. We will now focus on the set of *auxiliary* neurons V in $\mathcal{N}$. The network $\mathsf{Sync}(\mathcal{N})$ contains the vertices V of the original network $\mathcal{N}$, and in addition, for each neuron $v_i \in V$ we add the following components to the network:

- A synchronized sub-network $\mathsf{Sync}(v_i)$ using Lemma 16 implementing the threshold gate defined by neuron v_i. The input neurons to the sub-network $\mathsf{Sync}(v_i)$ are the incoming neighbors of v_i in $\mathcal{N}$. The first neuron v_i^* in the internal chain of the sub-network $\mathsf{Sync}(v_i)$ has an incoming edge from the L^{th} neuron of PG cycle, namely c_L with weight 1 and bias $b(v_i^*) = 1$. Denote the output of the sub-network $\mathsf{Sync}(v_i)$ by v_i^{out}.
- An AND module AND_i whose output neuron is v_i. This module is implemented by a circuit of $\mathsf{OR}_{\mathsf{sync}}$ and $\mathsf{NOT}_{\mathsf{sync}}$ gates with three layers (using simple De-morgan rule). The AND_i module receives input from the neuron v_i^{out} and from the $(\alpha L^4)^{th}$ neuron in PG, $c_{\alpha L^4}$ where α is a large enough constant. The internal chains of the AND_i circuit receive input from neuron c_β in PG where $\beta = \alpha L^4 + L$, making sure the circuit begins the execution *after* receiving all its inputs[11].

Modifications to the Circuit Synchronization of Sec. 4.1. So far, we handled the synchronization of circuits. In order to handle general networks (e.g., that contains self-loops and recurrent edges), we need to apply small adaptations to the synchronized sub-networks of Sec. 4.1. Specifically, unlike circuits, in the execution of a network, certain neurons (or gates) might be activated several times. To be able to re-use the sync. sub-networks throughout the execution, we need to reset the states kept by their self-loops.

We therefore adapt the construction of the sync. sub-network presented in Section 4.1 to reset themselves at the end of their computation. For each $\mathsf{NOT}_{\mathsf{sync}}$ gate, we augment its internal chain by $3 \cdot L^2$ neurons, and the last neuron of this chain is connected to an inhibitor neuron v_r. The inhibitor v_r has outgoing edges of weight $-\infty$ to all neurons in the sub-network. Due to Claim 30, it holds that the inhibition by v_r (i.e., the round in which v_r fires) occurs *after* the output neuron has already fired. Observe that the timing of the inhibition by v_r is set in a way that guarantees that all gates in the sub-network will be idle from that point on (i.e., there will be no delayed spikes that arrive after this inhibition). For the sub-network $\mathsf{OR}_{\mathsf{sync}}$ which do not contain self-loops, no adaptation is needed.

B.6 Correctness

Throughout, we fix a synchronous execution Π_{sync} and an asynchronous execution Π_{async}. For every neuron v and phase p, define the beginning of phase p of v in the asynchronous execution $(r(v,p))$ as the round in which the p^{th} spike of c_0 is fired. I.e., the p^{th} phase of v is the time interval $[r(v,p), r(v,p+1))$. For every round p, let $V^+_{\mathsf{sync}}(p)$ be the set of neurons that fire in round p in Π_{sync} (i.e., the neurons with positive entries in σ_p). Similarly, let $V^+_{\mathsf{async}}(p)$ be the set of neurons that fire during phase p.

▶ **Lemma 43.** *The networks* $\mathsf{Sync}(\mathcal{N})$ *and* $\mathcal{N}$ *have similar executions.*

In order to show the networks $\mathsf{Sync}(\mathcal{N})$ and $\mathcal{N}$ have similar executions, we show by induction on round (resp., phase) p that $V^+_{\mathsf{sync}}(p) = V^+_{\mathsf{async}}(p)$. For $p = 1$, let $V^+_{\mathsf{sync}}(0)$ be the neurons that fired at the beginning of the simulation in round 0. We will show that every neuron $v_i \in V$ fires in phase 1 iff $v_i \in V^+_{\mathsf{sync}}(1)$. For $v_i \in V$, the spikes from its incoming neighbors in $V^+_{\mathsf{sync}}(0)$ reach the sub-network $\mathsf{Sync}(v_i)$ by round L. The global chain in $\mathsf{Sync}(v_i)$ is then activated by the neuron c_L in some round $\tau \in [L, L^2]$. Therefore, by Lemma 16 there exists a constant γ such that v_i^{out} fires in some round $\tau_i \in [2, \tau + \gamma \cdot \log \Delta \cdot L^4]$ iff the output of the

[11] We say that the circuit receives its input, if every gate in the first layer has received the signals from its incoming input.

threshold function corresponding to v_i is 1, meaning that $v_i \in V^+_{\mathsf{sync}}(1)$. We next note that the first layer of the sub-network AND_i consists of two $\mathsf{NOT}_{\mathsf{sync}}$ sub-networks with input from $c_{\alpha \cdot L^4}$ and v_i^{out}. Hence, by Observation 32 as long as AND_i receives the information from $c_{\alpha \cdot L^4}$ and v_i^{out} before the activation of the global chain of the network AND_i in some round τ^* its output neuron fires by round $\tau^* + O(L^4)$ iff both v_i^{out} and $c_{\alpha \cdot L^4}$ fired.

The global chain of AND_i is activated by neuron c_β for $\beta = \alpha \cdot L^4 + L$ and therefore is indeed activated *after* AND_i receives the spike from $c_{\alpha L^4}$. In addition, we choose α such that $\alpha L^4 > L^2 + \gamma \cdot \log \Delta \cdot L^4$. Therefore the neuron c_β fires *after* round $\tau_i + L$, i.e. after AND_i received the spike from v_i^{out} as well. We conclude that v_i fires in some round $\tau'' \in [\beta, O(L^4)]$ iff v_i^{out} fires in round τ_i. We choose k to be large enough to make sure that c_k fires after round τ'' and therefore all neurons in $V^+(1)$ fired during the first phase.

Next, we assume that $V^+_{\mathsf{sync}}(p) = V^+_{\mathsf{async}}(p)$ and consider phase $p+1$. Let τ_p be the round that c_0 fired at the beginning of phase p and let τ_{p+1} be the round in which c_0 fired at the beginning of phase $p+1$. In addition, we denote the round in which $c_{\alpha L^4}$ fired during phase p by τ_α. By the induction assumption, neuron v_i fires between round τ_p and round τ_{p+1} iff $v_i \in V^+_{\mathsf{sync}}(p)$. Moreover, since the activation of the sub-network AND_i is performed by neuron c_β, every $v_i \in V^+_{\mathsf{sync}}(p)$ fires after round τ_α. We choose α to be large enough such that by round τ_α, all sub-networks $\mathsf{Sync}(v_i)$ have been reset due to the modification in the circuit synchronization. Hence, for neuron $v_i \in V$, the spikes from its incoming neighbors in $V^+_{\mathsf{async}}(p)$ reach $\mathsf{Sync}(v_i)$ after the sub-network has already been reset. Thus, when the global chain of the sub-network $\mathsf{Sync}(v_i)$ is activated by the neuron c_L in round $\tau_L \in [\tau_{p+1} + L, \tau_{p+1} + L^2]$, the sub-network $\mathsf{Sync}(v_i)$ received spikes from the incoming neighbors of v_i in $V^+_{\mathsf{async}}(p)$. Combining with Lemma 16 we conclude that v_i^{out} fires in round $\tau_i \in [\tau_{p+1} + L, \tau_{p+1} + L^2 + \gamma \cdot \log \Delta \cdot L^4]$ iff $v_i \in V^+_{\mathsf{sync}}(p+1)$. Thus, when neuron c_β fires in phase $p+1$, the sub-network AND_i has received the spikes from both v_i^{out} and $c_{\alpha L^4}$. Since the global chain of AND_i is activated by the neuron c_β, we conclude that v_i fires in some round $\tau^* \in [\tau_{p+1} + \beta, \tau_{p+1} + \Theta(L^4)]$, iff $v_i \in V^+_{\mathsf{sync}}(p+1)$. Choosing k to be large enough, τ^* occurs before c_k fires and ends the phase.

Synchronization of a Spiking Neural Network. We next explain the adaptation of the construction given a network of spiking neurons $\mathcal{N}$. Let n be the number of auxiliary neurons in $\mathcal{N}$ and let t be the number of rounds. Each spiking neuron implemented by a probabilistic threshold gate can be made synchronized using Cor. 19 where we use an error parameter of $\epsilon = 1/\operatorname{poly}(n,t)$. Thus, The network $\mathsf{Sync}(\mathcal{N})$ consists of $\operatorname{poly}(\Delta, \log n \cdot t) \cdot L^4 \cdot n$ auxiliary neurons and uses $\operatorname{poly}(\log \Delta, \log n \cdot t) \cdot L^5$ rounds.

To compare the simulation of the given spiking neural network $\mathcal{N}$ and the synchronized network $\mathsf{Sync}(\mathcal{N})$, we fix the randomness used by $\mathcal{N}$ throughout the simulation and use these coins when simulated the network $\mathsf{Sync}(\mathcal{N})$. For neuron $v \in V$ and round $\tau \geq 1$, by Cor. 19, with probability at least $1 - 1/\operatorname{poly}(n \cdot t)$ it holds that $v \in V^+_{\mathsf{sync}}(\tau)$ iff $v \in V^+_{\mathsf{async}}(\tau)$. By applying the union bound over all n neurons and t rounds of the simulation, we conclude that with high probability $\mathcal{N}$ and $\mathsf{Sync}(\mathcal{N})$ have similar executions.

C Synchronization in the Node-Delay Model

C.1 Network Dynamics in the Node Delay Setting

Network evolution proceeds in *seconds*, namely, a sufficiently small time unit. For a given integer $T \geq 1$, the dynamics is specified by a node-delay function $t : V \to \mathbb{N}_{\leq T}$ interpreted as follows: the round duration on each neuron v consists of $t(v)$ seconds. Specifically, the i^{th}

round of v is defined by the time interval $R_i(v) = [(i-1)t(v)+1, i \cdot t(v)]$ for every $i \geq 1$. All spikes are assumed to arrive with a delay of a single second[12]. For the neuron v and integer i, the set of spikes received at v during its i^{th} round is given by

$$A(v,i) = \{(u, j \cdot t(u)) \mid j \cdot t(u) + 1 \in R_i(v)\}.$$

The state of v in its i-round (i.e., at the second $i \cdot t(v)$) is given by:

$$\text{pot}(v,i) = \sum_{(u,j\cdot t(u)) \in A(v,i)} w(u,v) \cdot \sigma_j(v) \; - \; b(v) \;\text{ and }\; \sigma_i(v) = 1 \text{ iff } \text{pot}(v,i) \geq 0 \; . \tag{6}$$

If v is a probabilistic threshold gate then it fires in second $i \cdot t(v)$ with probability $p(v,i) = \frac{1}{1+e^{-\text{pot}(v,i)}}$.

▶ **Definition 44** (The T-bounded Node-Delay Setting). *We are given a network $\mathcal{N}$ and an integer T. It is assumed the network contains a special neuron, the* starter*, that fires in the first round of the simulation. The dynamic is determined by a node-delay function $t : V \to \mathbb{N}_{\leq T}$. This function t can be chosen arbitrarily.*

▶ **Definition 45** (Computation of a Boolean Function in the T-bounded Node-Delay Setting). *Let $f : \{0,1\}^n \to \{0,1\}^k$ be a Boolean function. A network $\mathcal{N}$ with n input neurons $x_1, \ldots, x_n$ and k output neurons $z_1, \ldots, z_k$* computes *f in this setting if for every T-bounded function $t : V \to \mathbb{N}_{\leq T}$ and for every fixed possible assignment to the input neurons $b_1, \ldots, b_n$ the following holds: (i) If $f_i(b_1, \ldots, b_n) = 1$, then there exists a round in which z_i fires, where $f_i(\cdot)$ is the i^{th} bit in the output of f. (ii) If $f_i(b_1, \ldots, b_n) = 0$ then z_i does not fire throughout the entire execution.*

Synchronizers for the Node-Delay. A synchronizer ν is an algorithm that gets as input a network $\mathcal{N}$ and integer T, and outputs a network $\mathcal{N}' = \mathsf{sync}_V(\mathcal{N}, T)$ that contains all the neurons of $\mathcal{N}$, plus additional auxiliary neurons. One of the auxiliary neurons in $\mathcal{N}'$ is a *starter* neuron that fires in the *first* round of the simulation. The network $\mathcal{N}'$ works in the asynchronous setting and should have *similar execution* to $\mathcal{N}$ in the sense that for every neuron $v \in V(\mathcal{N})$, the firing pattern of v in the asynchronous network should be similar to the one in the synchronous network. The output network $\mathcal{N}'$ simulates each round of the network $\mathcal{N}$ by a phase.

▶ **Definition 46** (Phases). *We partition the execution of $\mathcal{N}'$ into phases $1, 2, \ldots$, using a function $r : V(\mathcal{N}) \times \mathbb{N} \to \mathbb{N}$ that defines the beginning of phase p. Hence, the p^{th} phase is the round interval $[r(v,p), r(v,p+1))$.*

▶ **Definition 47** (Similar Executions (Deterministic Networks)). *The* synchronous *execution Π of a deterministic network $\mathcal{N}$ is specified by a list of states $\Pi = \{\sigma_1, \ldots, \}$ where each σ_i is a binary vector describing the firing status of the neurons in round i. The* asynchronous *execution of the network $\mathcal{N}' = \mathsf{sync}_V(\mathcal{N}, T)$ with a node-delay function $t : V \to \mathbb{N}_{\leq T}$ denoted by $\Pi'(t)$ is defined analogously only when applying the asynchronous dynamic. The execution $\Pi'(t)$ is divided into phases according the a function $r : V(\mathcal{N}) \times \mathbb{N} \to \mathbb{N}$.*

*The network $\mathcal{N}$ and the pair $\langle \mathcal{N}', t \rangle$ have a **similar execution** if $V(\mathcal{N}) \subseteq V(\mathcal{N}')$, and in addition, a neuron $v \in V(\mathcal{N})$ fires in round p in the execution Π iff v fires during phase p in $\Pi'(t)$.*

[12] As discussed in the introduction, this model can be generalized to support both edge-delays and node-delays, to isolate the node-delay effect we assume that all edges have latency of 1.

The networks $\mathcal{N}$ and $\mathcal{N}'$ are ***similar*** *if $\mathcal{N}$ and $\langle \mathcal{N}', t\rangle$ have a similar execution for every node-delay function t.*

As for the edge-delay model, the extension for randomized networks is made by fixing the random bits in the simulation of the input network.

C.2 Reduction to the Edge-Delay Model: A Simulation Result

Given a neural network $\mathcal{N}$ and an integer parameter T, our goal is to construct a network $\mathcal{N}_R = \mathsf{sync}_V(\mathcal{N}, T)$ in the T-bounded node-delay model that behaves similarly to $\mathcal{N}$, i.e., that $\mathcal{N}$ and $\mathcal{N}_R$ are similar according to Definition 47.

Given the network $\mathcal{N}$ and the delay bound T, we start by computing the network $\mathcal{N}_\mathcal{L} = \mathsf{sync}_E(\mathcal{N}, L)$ with $L = 5T^2$. The desired $\mathcal{N}_R = \mathsf{sync}_V(\mathcal{N}_T)$ is obtain by changing some of the edge weights in $\mathcal{N}_\mathcal{L}$. Our proof of correctness is based on similarity between a network in the node-delay model and a network in the edge-delay model.

Similarity between the networks $\mathcal{N}_R$ and $\mathcal{N}_\mathcal{L}$. Fix integer parameters T, L. Given an edge-delay network $\mathcal{N}_\mathcal{L}$, a latency function $\ell : E(\mathcal{N}_\mathcal{L}) \times \mathbb{N} \to \mathbb{N}_{\leq L}$, a node-delay network $\mathcal{N}_R$ on the same neuron set and a node-delay function $t : V(\mathcal{N}_R) \to \mathbb{N}_{\leq T}$, we want to define similarity between the simulations $\langle \mathcal{N}_\mathcal{L}, \ell\rangle$ and $\langle \mathcal{N}_R, t\rangle$, where both simulations use the same initial configuration.

This notion of similarity is based on defining different time scales in each of the simulations. Specifically, for every $i \geq 1$ and neuron $u \in V$ the time window $R_i(u)$ will be the time that u collects spikes for its round i in the simulation of $\langle \mathcal{N}_R, t\rangle$. Moreover, for every $i \geq 0$ the time window $L_i(u)$ correspond to the firing period of round i of u in the simulation of $\langle \mathcal{N}_R, t\rangle$, where

$$R_i(u) = [(i-1)\cdot t(u) + 1, i \cdot t(u)] \text{ and } L_i(u) = [i \cdot T \cdot t(u), i \cdot T \cdot t(u) + (T \cdot t(u) - 1)].$$

Furthermore, for every second τ_R in the simulation of $\langle \mathcal{N}_R, t\rangle$ we will have the corresponding block $B_{\tau_R} = [\tau_R \cdot T, \tau_R \cdot T + (T-1)]$ in the simulation of $\langle \mathcal{N}_\mathcal{L}, \ell\rangle$. For the simulation of $\langle \mathcal{N}_\mathcal{L}, \ell\rangle$ define for every neuron u and $i \geq 0$:

$$\sigma_i(u, \mathcal{N}_\mathcal{L}) = \begin{cases} 1 & u \text{ is strong and } u \text{ fires in every } \tau_\mathcal{L} \in L_i(u) \\ 1 & u \text{ is weak and } u \text{ fires in } \tau_\mathcal{L} \in L_i(u) \text{ only for } \tau_\mathcal{L} = i \cdot T \cdot t(u) \\ 0 & u \text{ never fires in } L_i(u) \\ \emptyset & \text{Otherwise.} \end{cases}$$

For the simulation of $\langle \mathcal{N}_R, t\rangle$ define for every neuron u and $i \geq 0$:

$$\sigma_i(u, \mathcal{N}_R) = \begin{cases} 1 & u \text{ fires in round } i \text{ of } u \text{ (i.e. in the second } i \cdot t(u)) \\ 0 & \text{Otherwise.} \end{cases}$$

▶ **Definition 48.** *The simulations $\langle \mathcal{N}_R, t\rangle, \langle \mathcal{N}_\mathcal{L}, \ell\rangle$ are* similar*, denoted as $\langle \mathcal{N}_R, t\rangle \sim \langle \mathcal{N}_\mathcal{L}, \ell\rangle$, if for every neuron u and $i \geq 0$ it holds that $\sigma_i(u, \mathcal{N}_\mathcal{L}) = \sigma_i(u, \mathcal{N}_R)$.*

A network $\mathcal{N}_\mathcal{L}$ in the L-bounded edge-delay model and a network $\mathcal{N}_R$ in the T-bounded node-delay model are similar*, denoted by $\mathcal{N}_\mathcal{L} \sim \mathcal{N}_R$, if for every node-delay function $t : V(\mathcal{N}_R) \to \mathbb{N}_{\leq T}$ there exists a latency function $\ell : E(\mathcal{N}_\mathcal{L}) \times \mathbb{N} \to \mathbb{N}_{\leq L}$ such that $\langle \mathcal{N}_R, t\rangle \sim \langle \mathcal{N}_\mathcal{L}, \ell\rangle$.*

The key simulation lemma used in the synchronization scheme is as follow:

▶ **Lemma 49.** *Given a network $\mathcal{N}_\mathcal{L}$ in the L-bounded edge delay model such that:*

1. *$b(u) > 0$ for every neuron u.*
2. *Every weak neuron v has no self-loop.*
3. *There is no edge from a strong neuron to a strong neuron.*
4. *Every negative edge has weight $-\infty$.*
5. *For every neuron u, either any excitatory incoming neighbor of u is weak, or any excitatory incoming neighbor of u is strong.*
6. *Let v be a strong incoming neighbor of a neuron u, and let f be an inhibitor. Then if f has an edge to v, it also has an edge to u.*

Then there exists a network $\mathcal{N}_R$ in the T-bounded node-delay model with $T \leq \sqrt{L/5}$ with $V(\mathcal{N}_R) = V(\mathcal{N}_\mathcal{L})$ such that $\mathcal{N}_R$ and $\mathcal{N}_\mathcal{L}$ are similar.

Defining the node-delay network $\mathcal{N}_R$. The network $\mathcal{N}_R$ is exactly as $\mathcal{N}_\mathcal{L}$, up to small adaption of the weights. Denote by $w_\mathcal{L} : V \to \mathbb{R}$ the weight function of the network $\mathcal{N}_\mathcal{L}$. Define the weight function w_R of $\mathcal{N}_R$ as

$$w_R(v,u) = \begin{cases} T \cdot w_\mathcal{L}(v,u) & v \neq u,\ v \text{ is strong,} \\ w_\mathcal{L}(v,u) & \text{Otherwise.} \end{cases}$$

Correctness. We will show that $\mathcal{N}_\mathcal{L}$ and $\mathcal{N}_R$ are similar. Fix a node-delay function $t : V \to \mathbb{N}_{\leq T}$. First, we define the corresponding latency function ℓ and prove it is valid, i.e. that ℓ is nice and $\ell(v,u,\tau) \in [1,L]$ for every neurons v,u and round τ. Then, we restate Lemma 49 in order to prove its correctness by induction on the round.

Definition of the latency function ℓ. First, set the latency of self-spikes to be of value 1. For a neuron u, we say that u is *weak-incoming* if any excitatory incoming neighbor of u is weak, and we say that u is *strong-incoming* if any excitatory incoming neighbor of u is strong. Note that by property 5, every neuron u is either weak-incoming or strong-incoming. For a strong-incoming neuron u, an inhibitor v and $\tau_\mathcal{L} \geq 0$, set $\ell(v,u,\tau_\mathcal{L}) = 2T^2 + 1$. Now consider the remaining spikes, which are either positive spikes, or spikes to a weak-incoming neuron u.

For every $\tau_\mathcal{L} \geq 0$ define the latency value for the spike event $\langle v,u,\tau_\mathcal{L}\rangle$ as follows. Let j be an integer satisfying that $\tau_\mathcal{L} \in L_j(v)$, and let i be such that $j \cdot t(v) + 1 \in R_i(u)$, hence $(v, j \cdot t(v)) \in A(u,i)$.

If v is weak, then for $\tau_\mathcal{L} = j \cdot T \cdot t(v)$ set $\ell(v,u,\tau_\mathcal{L}) = i \cdot T \cdot t(u) - \tau_\mathcal{L}$. That is, the spike $\langle v,u,\tau_\mathcal{L}\rangle$ is scheduled to arrive in the first round of $L_i(u)$. For $\tau_\mathcal{L} > j \cdot T \cdot t(v)$, set $\ell(v,u,\tau_\mathcal{L}) = 1$. Otherwise, if v is strong, consider the following argument. For every second $\tau_\mathcal{L}$ in the edge-latency simulation, let τ_R be the second in the node-delay simulation such that $\tau_\mathcal{L} \in B_{\tau_R}$.

- **Case (I):** there exists a second in $[\tau_R + 1, \tau_R + 2T]$ such that u fires in the node-delay simulation, let τ'_R be the first such second. Set $\ell(v,u,\tau_\mathcal{L}) = \tau'_R \cdot T - \tau_\mathcal{L}$, that is schedule $\langle v,u,\tau_\mathcal{L}\rangle$ to arrive in round $\tau'_R \cdot T$.
- **Case (II):** case I does not apply, and there is an inhibitor f which is an incoming neighbor of u, and a second $\tau'_R \in [\tau_R - T, \tau_\mathcal{L} + 2T]$ such that f fires in τ'_R in the node-delay simulation. Then for such τ'_R, set $\ell(v,u,\tau_\mathcal{L}) = \tau'_R \cdot T + (2T^2 + 1) - \tau_R$, that is schedule $\langle v,u,\tau_\mathcal{L}\rangle$ to arrive in round $\tau'_R \cdot T + (2T^2 + 1)$.
- **Case (III):** neither case (I) nor case (II) apply. Set $\ell(v,u,\tau_\mathcal{L}) = 1$.

The intuition is that for a positive spike in the edge-delay simulation, we look for a round such that u is supposed to fire in the next $2T^2$ rounds. If we cannot find one, we want to send the spike to a round that we know it will not activate u. This is a round in which u receives a negative spike (since negative spikes are of weight $-\infty$). If such round also does not exist, it implies that the total weight of positive incoming neighbors of u that fired in round $\tau_{\mathcal{L}}$ is low, and we can schedule all these spikes to arrive together in $\tau_{\mathcal{L}}+1$ without activating u. We next show that ℓ is valid.

▷ Claim 50. ℓ is a valid latency function for $\mathcal{N}_{\mathcal{L}}$.

Proof. First, since all self-spikes have latency value 1, ℓ is nice. For a negative spike $\langle v,u,\tau_{\mathcal{L}}\rangle$ such that u is strong-incoming, it holds that $\ell(v,u,\tau)=2T^2+1<L$. Therefore we are left to show validity for positive spikes, and for negative spikes that are fired towards a weak-incoming neuron. Consider a spike $\langle v,u,\tau_{\mathcal{L}}\rangle$, and let j be an integer satisfying that $\tau_{\mathcal{L}}\in L_j(v)$. Furthermore, let i be an integer such that $j\cdot t(v)+1\in R_i(u)$.

Next, assume that v is weak. We distinguish between two cases depending whether $\tau_{\mathcal{L}}$ is the first round in the block or not. For $\tau_{\mathcal{L}}=i\cdot T\cdot t(v)$ we have $\ell(v,u,\tau_{\mathcal{L}})=i\cdot T\cdot t(u)-\tau_{\mathcal{L}}$. Recall that $R_i(u)=[(i-1)\cdot t(u)+1, i\cdot t(u)]$, thus $j\cdot t(v)+1\leq i\cdot t(u)$, and $\ell(v,u,\tau_{\mathcal{L}})=T\cdot(i\cdot t(u)-j\cdot t(v))\geq T\geq 1$. Furthermore $j\cdot t(v)+1\geq (i-1)\cdot t(u)+1$, hence $i\cdot t(u)-j\cdot t(v)\leq t(u)\leq T$, and $\ell(v,u,\tau_{\mathcal{L}})\leq T\cdot(i\cdot t(u)-j\cdot t(v))\leq L$. Otherwise, i.e. for $\tau_{\mathcal{L}}\geq i\cdot T\cdot t(v)$, it holds that $\ell(v,u,\tau_{\mathcal{L}})=1$, and thus $\ell(v,u,\tau_{\mathcal{L}})\in[1,L]$.

It remains to consider the case where v is strong. Let τ_R be the second in the node-delay simulation such that $\tau_{\mathcal{L}}\in B_{\tau_R}$. Consider the definition of ℓ for a spike $\langle v,u,\tau_{\mathcal{L}}\rangle$. In case (I), we have $\ell(v,u,\tau_{\mathcal{L}})=\tau'_R\cdot T-\tau_{\mathcal{L}}$, and since $\tau'_R\in[\tau_R+1,\tau_R+2T]$ it holds that $1\leq\tau'_R\cdot T-\tau_{\mathcal{L}}\leq 2T^2<L$. In case (II), since $\tau'_R\in[\tau_R-T,\tau_R+2T]$, we have that $\ell(v,u,\tau_{\mathcal{L}})=\tau'_R\cdot T+(2T^2+1)-\tau_{\mathcal{L}}\in[1,5T^2]$. Finally, in case (III) we simply have $\ell(v,u,\tau_{\mathcal{L}})=1$. Hence, in all cases it holds that $\ell(v,u,\tau_{\mathcal{L}})\in[1,L]$. ◁

In order to show that $\langle\mathcal{N}_R,t\rangle\sim\langle\mathcal{N}_{\mathcal{L}},\ell\rangle$, we restate the condition for similarity in the following lemma. We then prove the lemma by induction on the round $\tau_{\mathcal{L}}$.

▶ **Lemma 51** (Restating Lemma 49). *For every round $\tau_{\mathcal{L}}\geq 0$ of the simulation $\langle\mathcal{N}_{\mathcal{L}},\ell\rangle$ and for every neuron u, let i be such that $\tau_{\mathcal{L}}\in L_i(u)$. Then:*

1. *If $\sigma_i(u,\mathcal{N}_R)=1$:*
   - *If $\tau_{\mathcal{L}}=i\cdot T\cdot t(u)$ then u fires in $\tau_{\mathcal{L}}$.*
   - *If $\tau_{\mathcal{L}}>i\cdot T\cdot t(u)$ then u fires iff u is strong.*
2. *If $\sigma_i(u,\mathcal{N}_R)=0$ then u does not fire in $\tau_{\mathcal{L}}$.*

For the base case $\tau_{\mathcal{L}}=0$, the correctness follows the fact that both simulations have the same starting configuration. Now, let $\tau_{\mathcal{L}}\geq 1$ and assume correctness for every $\tau'_{\mathcal{L}}\leq\tau_{\mathcal{L}}-1$. Fix a neuron u and let i be an integer such that $\tau_{\mathcal{L}}\in L_i(u)$. We start with a useful auxiliary claim.

▷ Claim 52. Let u be a weak-incoming neuron, v an incoming neighbor of u, and $\tau'_{\mathcal{L}}\geq 0$. Furthermore, let j be such that $\tau'_{\mathcal{L}}\in L_j(v)$, and i such that $j\cdot t(v)+1\in R_i(u)$. Then the spike $\langle v,u,\tau'_{\mathcal{L}}\rangle$ occurs and arrives to u in round $\tau_{\mathcal{L}}=i\cdot T\cdot t(u)$ in the simulation $\langle\mathcal{N}_{\mathcal{L}},\ell\rangle$ iff $\tau'_{\mathcal{L}}=j\cdot t(v)$ and the spike $\langle v,u,j\cdot t(v)\rangle$ occurs and arrives to u in $R_i(u)$ in the simulation $\langle\mathcal{N}_R,t\rangle$.

Proof of Claim 52. Since u is weak-incoming v is weak, then by the induction assumption for $\tau'_{\mathcal{L}}$ and the definition of ℓ, the spike event $\langle v,u,\tau'\rangle$ occurs and arrives in round $\tau_{\mathcal{L}}$ iff there exists j such that $\tau'_{\mathcal{L}}=j\cdot T\cdot t(v)$ and $\sigma_j(v,\mathcal{N}_R)=1$. This happens iff in the simulation of $\langle\mathcal{N}_R,t\rangle$ the spike event $\langle v,u,j\cdot t(v)\rangle$ occurs and arrives to u in $R_i(u)$. ◁

We split the proof of Lemma 51 into two cases.

Case 1: u is weak-incoming. Assume $\tau_{\mathcal{L}} = i \cdot T \cdot t(u)$, we want to show that u fires in round $\tau_{\mathcal{L}}$ iff $\sigma_i(u, \mathcal{N}_R) = 1$. By Claim 52, we get that the mapping $\langle v, u, j \cdot T \cdot t(v) \rangle \mapsto \langle v, u, j \cdot t(v) \rangle$ is a bijection between the set of non self-spikes that u receives in $\tau_{\mathcal{L}}$ in the simulation $\langle \mathcal{N}_{\mathcal{L}}, \ell \rangle$ and the set of non-self spikes that u receives in $R_i(u)$ in the simulation $\langle \mathcal{N}_R, t \rangle$. As for self-spikes, note that if u is weak it has no self-loop. If u is strong, then by the induction assumption u fires in $\tau_{\mathcal{L}} - 1$ iff $\sigma_{i-1}(u, \mathcal{N}_R)$. Thus, u receives the self-spike $\langle u, u, \tau_{\mathcal{L}} - 1 \rangle$ in $\tau_{\mathcal{L}}$ iff it receives the self-spike $\langle u, u, (i-1) \cdot T \cdot t(u) \rangle$ in $R_i(u)$. Since $w_{\mathcal{L}}(v, u) = w_R(v, u)$ for every weak neuron v and for $v = u$, we get that the total spike weight that u receives in $\tau_{\mathcal{L}}$ equals to the total spike weight it receives in $R_i(u)$. Thus, u fires in round $\tau_{\mathcal{L}}$ iff $\sigma_i(u, \mathcal{N}_R) = 1$.

Now, assume $\tau_{\mathcal{L}} > i \cdot T \cdot t(u)$ and that either v is weak, or v is strong and $\sigma_i(u, \mathcal{N}_R) = 0$. We want to show that u does not fire. Note that if v is weak then it has no self-loop, and if v is strong and $\sigma_i(u, \mathcal{N}_R) = 0$ then by the induction assumption for $\tau_{\mathcal{L}} - 1$, u does not fire in $\tau_{\mathcal{L}} - 1$. Thus, in both cases u does not receive a self-spike in $\tau_{\mathcal{L}}$. Furthermore, u has no strong neighbors, therefore by Claim 52 u does not receive any positive spikes from incoming neighbors. Since $b(u) > 0$, u does not fire in $\tau_{\mathcal{L}}$.

Finally, assume $\tau_{\mathcal{L}} > i \cdot T \cdot t(u)$, and assume u is strong and $\sigma_i(u, \mathcal{N}_R) = 1$. We want to show that u fires. Note that by Claim 52, u does not receive a negative spike in $\tau_{\mathcal{L}}$. Furthermore, since $\sigma_i(u, \mathcal{N}_R) = 1$ by the induction assumption for $\tau_{\mathcal{L}} - 1$, u fires in $\tau_{\mathcal{L}} - 1$ and therefore u receives a self-spike in $\tau_{\mathcal{L}}$. Since $w_{\mathcal{L}}(u, u) \geq b(u)$, u fires in $\tau_{\mathcal{L}}$.

Case 2: u is strong-incoming. By the properties of $\mathcal{N}_{\mathcal{L}}$ there is no edge between strong neurons, and weak neurons have no self-loop. Hence u is weak and has no self-loop. We handle separately the following sub-cases:

Case 2.1: $\sigma_i(u, \mathcal{N}_R) = 1$ and $\tau_{\mathcal{L}} = i \cdot T \cdot t(u)$. We want to show that u fires in $\tau_{\mathcal{L}}$. Let $\langle v, u, j \cdot t(v) \rangle$ be a positive spike in the simulation $\langle \mathcal{N}_R, t \rangle$ that arrives to u in $R_i(u)$, and let $\tau'_{\mathcal{L}}$ be one of the T rounds $[j \cdot T \cdot t(v), j \cdot T \cdot t(v) + (T-1)]$. Since v is strong then by the induction assumption for $\tau'_{\mathcal{L}}$ v fires in $\tau'_{\mathcal{L}}$, and therefore the spike event $\langle v, u, \tau'_{\mathcal{L}} \rangle$ occurs in the simulation $\langle \mathcal{N}_{\mathcal{L}}, \ell \rangle$. We now show that $\langle v, u, \tau'_{\mathcal{L}} \rangle$ arrives to u in $\tau_{\mathcal{L}}$, according to the definition of ℓ for spikes from strong neurons.

Since $\sigma_i(u, \mathcal{N}_R) = 1$, u fires in the second $r_R = i \cdot t(u)$ in the simulation $\langle \mathcal{N}_R, t \rangle$. Note that $j \cdot t(v) + 1 \in R_i(u)$ implies that $i \cdot t(u) - j \cdot t(v) \leq T$. Hence in particular $i \cdot t(u) \in [j \cdot t(v) + 1, j \cdot t(v) + 2T]$. Let $r'_R \in [j \cdot t(v) + 1, j \cdot t(v) + 2T]$ with $r'_R < i \cdot t(u)$. Note that $r'_R \in R_i(u)$, therefore r'_R is not an end of a round of u. Hence u does not fire in r'_R. Therefore the second $r_R = i \cdot t(u)$ is the first second in $[j \cdot t(v) + 1, j \cdot t(v) + 2T]$ that u fires, and due to the definition of ℓ the spike $\langle v, u, \tau'_{\mathcal{L}} \rangle$ arrives in round $\tau_{\mathcal{L}}$.

Now, let f be an inhibitory incoming neighbor of u. By the definition of ℓ, a spike from f to u can arrive only in a round of the form $\tau'_R \cdot T + T^2 + 1$ for some second τ'_R, which is not a multiplicity of T. Note that $\tau_{\mathcal{L}} = i \cdot T \cdot t(u)$ is a multiplicity of T. Thus u does not receive a negative spike in $\tau_{\mathcal{L}}$.

We get that in round $\tau_{\mathcal{L}}$, u receives only positive spikes in $\tau_{\mathcal{L}}$, and for every positive spike $\langle v, u, j \cdot t(v) \rangle$ that arrives to u in $R_i(u)$ and every $\tau'_{\mathcal{L}} \in [j \cdot T \cdot t(v), j \cdot T \cdot t(v) + (T-1)]$, u receives a spike $\langle v, u, \tau'_{\mathcal{L}} \rangle$. Since $w_R(v, u) = T \cdot w_{\mathcal{L}}(v, u)$ for every strong v and $[j \cdot T \cdot t(v), j \cdot T \cdot t(v) + (T-1)]$ contains T rounds, we get that the total spike weight that u receives in $\tau_{\mathcal{L}}$ is at least the total spike weight it receives in $R_i(u)$ in the node-delay simulation. Since $\sigma_i(u, \mathcal{N}_R) = 1$, u receives in $R_i(u)$ a spike weight of at least $b(u)$, which implies the same for round $\tau_{\mathcal{L}}$ in the edge-delay simulation. Thus u fires in round $\tau_{\mathcal{L}}$.

Case 2.2: $\sigma_i(u, \mathcal{N}_R) = 0$ or $\tau_{\mathcal{L}} > i \cdot T \cdot t(u)$. We want to show that u does not fire in $\tau_{\mathcal{L}}$. Towards contradiction, assume that it does. First note that if u receives no positive spikes in $\tau_{\mathcal{L}}$, then since $b(u) > 0$ u does not fire in $\tau_{\mathcal{L}}$. Otherwise, let $\langle v, u, \tau'_{\mathcal{L}} \rangle$ be a positive spike that arrives to u in round $\tau_{\mathcal{L}}$. Recall that since v is strong, there are three cases for defining the latency value of $\langle v, u, \tau'_{\mathcal{L}} \rangle$.

We will now show that $\langle v, u, \tau'_{\mathcal{L}} \rangle$ belongs to case (II). It does not belong to case (I), since it does not hold that $\sigma_i(u) = 1$ and $\tau_{\mathcal{L}} = i \cdot T$ If we are in case (II), then there exists an inhibitor f which is connected to u that fired in second τ'_R in the node-delay simulation that arrived in $\tau_{\mathcal{L}}$, i.e. such that $\tau_{\mathcal{L}} = \tau'_R \cdot T + (T^2 + 1)$. By the induction assumption for $\tau'_R \cdot T$, u fires in round $\tau'_R \cdot T$ in the edge-delay simulation, and since u is strong-incoming then by the definition of ℓ the spike $\langle f, u, \tau'_R \cdot T \rangle$ arrives to u in round $\tau'_R \cdot T + (T^2 + 1) = \tau_{\mathcal{L}}$. Since negative spikes are of weight $-\infty$, u does not fire in $\tau_{\mathcal{L}}$. Therefore, $\langle v, u, \tau'_{\mathcal{L}} \rangle$ belongs to case (III).

By the definition of case (III), $\langle v, u, \tau'_{\mathcal{L}} \rangle$ was generated in round $\tau'_{\mathcal{L}} = \tau_{\mathcal{L}} - 1$. Let j be such that $\tau_{\mathcal{L}} - 1 \in L_j(v)$, and let τ_R such that $\tau_{\mathcal{L}} - 1 \in B_{\tau_R}$. Furthermore, let v be an excitatory incoming neighbor that fires in $\tau_{\mathcal{L}} - 1$, let j be such that $\tau_{\mathcal{L}} - 1 \in L_j(v)$, and let τ_R such that $\tau_{\mathcal{L}} - 1 \in B_{\tau_R}$. Our goal is to show that v fires in $[\tau_R + 1, \tau_R + T]$ in the node-delay simulation, by showing that it receives enough positive spikes from its neighbors in this interval.

Let f be an inhibitor that has an edge to v. By the network properties f also has an edge to u, and since we are not in case (II) in the definition of ℓ, f does not fire in the interval $[\tau_R - T, \tau_{\mathcal{L}} + 2T]$ in the node-delay simulation. This implies that v does not receive a negative spike in $[\tau_R - T + 1, \tau_R + 2T + 1]$. Notice that $\tau_R \in [j \cdot t(v), (j+1) \cdot t(v) - 1]$, and since $t(v) \leq T$ we get

$$R_{j+1}(v) = [j \cdot t(v) + 1, (j+1) \cdot t(v)] \subseteq [\tau_R - T + 1, \tau_R + 2T + 1].$$

Therefore, v does not receive a negative spike in $R_{j+1}(v)$.

By the induction assumption for $\tau_{\mathcal{L}} - 1$ we have $\sigma_j(v, \mathcal{N}_R) = 1$. Together with the fact that v is strong and receives no negative spikes in $R_{j+1}(v)$, we get that $\sigma_{j+1}(v, \mathcal{N}_R) = 1$, i.e. v fires in the node-delay simulation in the second $(j+1) \cdot t(v)$. This implies that u receives a spike from v in $(j+1) \cdot t(v) + 1$, which is inside the interval $[\tau_R + 1, \tau_R + T]$. If so, let W the total weight of the incoming neighbors of u that fired in round $\tau_{\mathcal{L}} - 1$. Since u fires in round $\tau_{\mathcal{L}}$, it holds that $W \geq b(u)$. We will show this implies that u fires in some round in $[\tau_R + 1, \tau_R + 2T]$, which contradicts the fact that none of the arriving spikes belong to case I.

We showed that for every neuron v that fires in $\tau_{\mathcal{L}} - 1$ in the edge-latency simulation, u receive a spike from v in some round $\tau'_R \in [\tau_R + 1, \tau_R + T]$ in the node-delay simulation. By the definition of w_R it holds that $w_R(v, u) = T \cdot w_{\mathcal{L}}(v, u)$, and therefore we get

$$\sum_{\tau'_R = \tau_R + 1}^{\tau_R + T} W_{\tau'_R} \geq T \cdot W.$$

By an averaging argument there is a second $\tau'_R \in [\tau_R + 1, \tau_R + T]$ with $W_{\tau'_R} \geq (T \cdot W)/T = W$.

Let i' be an integer such that $\tau'_R \in R_{i'}(u)$. Therefore u receive in $R_{i'}(u)$ a total positive spike weight of at least $W \geq b(u)$. Furthermore, since no spike belongs to case C.2, u do not receive a negative spike in $R_{i'}(u) \subseteq [\tau_R + 1, \tau_R + 2T]$. Thus, u fires in the second $i' \cdot t(u) \in [\tau_R + 1, \tau_R + 2T]$, a contradiction.

C.3 The Complete Synchronization Scheme

We are now ready to complete the proof of Theorem 5. We consider a neural network $\mathcal{N}$ and an integer parameter T. Set $L = 5T^2$ and let $\mathcal{N}_\mathcal{L} = \mathsf{sync}_E(\mathcal{N}, L)$ be the synchronized network of $\mathcal{N}$ in the L-bounded node-delay model. We will now show that $\mathcal{N}_\mathcal{L}$ satisfies the properties in the conditions of Lemma 49.

Showing that $\mathcal{N}_\mathcal{L}$ satisfies the properties of Lemma 49. Note that by the definition of the edge-delay synchronization scheme given in Section 4.3, every neuron $u \in \mathcal{N}_\mathcal{L}$ is contained in one of the following modules: (1) an $\mathsf{OR}_{\mathsf{sync}}$ or $\mathsf{NOT}_{\mathsf{sync}}$ subnetwork (Section 4.1), (2) a chain of a threshold gate which is a implemented as a boolean circuit subnetwork (Section B.3), or (3) the chain of the global pulse generator (Section 4.3). By the definitions of these modules, properties 1 and 2 hold. Moreover, together with the fact that edges between the modules connect only weak excitatory neurons, we also get property 3. Furthermore, note that the only inhibitors in the network are r and v_r neurons (which is later added in 4.3) in the $\mathsf{NOT}_{\mathsf{sync}}$ module, and all their edges have weight $-\infty$. Therefore, property 4 is satisfied.

The remaining properties 5 and 6 are relevant only for strong neurons. Therefore, consider the $\mathsf{NOT}_{\mathsf{sync}}$ module (Lemma 14), which is the only module that contains strong neurons. By the module definition, there are two possible types of strong neurons: (i) the memory neuron m, that is only connected to the reset neuron r; and (ii) intermediate neuron v_i, that is only connected to the output neuron z. In case (i), v has only one incoming inhibitor, which is the neuron v_r that resets the whole network after it finishes. Thus v_r also has an edge to r. In case (ii), v has two incoming inhibitors, v_r and r, which both have an edge to z. Therefore property 6 holds. Furthermore, both r and z have no edges from weak neurons. Hence, property 5 holds.

Indeed, $\mathcal{N}_\mathcal{L}$ satisfies the conditions of Lemma 49, and therefore there exists a network $\mathcal{N}_R$ in the T-bounded node delay model which is similar to $\mathcal{N}_\mathcal{L}$. We are left to show the transitivity of similarity, i.e. that if $\mathcal{N} \sim \mathcal{N}_\mathcal{L}$ and $\mathcal{N}_\mathcal{L} \sim \mathcal{N}_R$, also $\mathcal{N} \sim \mathcal{N}_R$.

Showing transitivity of similarity. Let t be a node-delay function for $\mathcal{N}_R$. First, by the similarities of the networkss we get $V(\mathcal{N}) = V(\mathcal{N}_\mathcal{L}) = V(\mathcal{N}_R)$. Moreover, by the definition of $\mathcal{N}_\mathcal{L} \sim \mathcal{N}_R$ there exists a latency function ℓ for $\mathcal{N}_\mathcal{L}$ such that $\langle \mathcal{N}_\mathcal{L}, \ell \rangle \sim \langle \mathcal{N}_R, t \rangle$. Let Π be the execution of $\mathcal{N}$, $\Pi_\mathcal{L}$ be the execution of $\langle \mathcal{N}, \ell \rangle$, and Π_R be the execution of $\langle \mathcal{N}, t \rangle$. Let the interval $[r_\mathcal{L}(v,p), r_\mathcal{L}(v,p+1))$ be the p^{th} phase of $\Pi_\mathcal{L}$, and define $[r_R(v,p), r_R(v,p+1))$ as the p^{th} phase of Π_R, where the definition of $r_R(v,p)$ is as follows. Let $L_p^*(v)$ be the earliest block $L_i(v)$ whose first round τ_p^* is contained in phase p of $\Pi_\mathcal{L}$, then $r_R(v,p) = \tau_p^*/T$. We wish to prove the following claim.

▷ Claim 53. For every neuron v and $p \geq 0$, v fires in round p of Π iff v fires in phase p of Π_R.

First, note that by the construction of $\mathcal{N}_\mathcal{L} = \mathsf{sync}_(\mathcal{N}, \mathcal{L})$, every neuron $v \in V(\mathcal{N})$ can fire only after the chain neuron $c_{\alpha L^4 + L}$ fires. Since $\alpha L^4 > t(v) \cdot T$ this implies that v does not fire in the first $t(v) \cdot T$ rounds of each phase in $\Pi_\mathcal{L}$. We prove the two directions of the claim.

- Assume neuron v fires in round p in Π. Because $\mathcal{N} \sim \langle \mathcal{N}_\mathcal{L}, \ell \rangle$ there is a round $\tau_\mathcal{L}$ in phase p of $\Pi_\mathcal{L}$ where v fires. Since v does not fire in the first $t(v) \cdot T$ rounds of each phase we have $\tau_\mathcal{L} \geq r_\mathcal{L}(v,p) + t(v) \cdot T$. Since $L_j(v)$ consists of $t(v) \cdot T$ rounds, the first round of $L_j(v)$ is in phase p. Therefore, $j \cdot t(v) \cdot T \geq \tau^*$, and therefore $j \cdot t(v) \geq r_R(v,p)$.

Furthermore we have that $j \cdot t(v)$ is not in phase $p+1$. Hence also $j \cdot (v) < r_R(v, p+1)$, i.e. $j \cdot t(v)$ is in phase p of Π. Due to the similarity $\langle \mathcal{N}_{\mathcal{L}}, \ell \rangle \sim \langle \mathcal{N}_R, t \rangle$, since v fires in $L_j(v)$ it also fires in $j \cdot t(v)$. Hence v fires in phase p of Π_R.

- Assume that v fires in phase p in Π_R. Assume this happens in round τ_R, then $\tau_R \geq \tau_p^*/T$. Thus $j \cdot T \cdot t(v) \geq \tau_p^* \geq r_{\mathcal{L}}(v, i)$. Furthermore, $j \cdot t(v) < \tau_p^*/T$ implies that round $j \cdot T \cdot t(v)$ was before phase $p+1$ of $\Pi_{\mathcal{L}}$. Therefore $j \cdot T \cdot t(v)$ is in phase p of $\Pi_{\mathcal{L}}$. By the similaritiy $\langle \mathcal{N}_{\mathcal{L}}, \ell \rangle \sim \langle \mathcal{N}_R, t \rangle$ we have that v fires in round $j \cdot T \cdot t(v)$ in $\Pi_{\mathcal{L}}$. Hence v fires in phase p of $\Pi_{\mathcal{L}}$. By the similarity $\mathcal{N} \sim \langle \mathcal{N}_{\mathcal{L}}, \ell \rangle$ we get that v fires in round p of Π.

Certified Algorithms: Worst-Case Analysis and Beyond

Konstantin Makarychev
Northwestern University, Evanston, IL, USA
https://konstantin.makarychev.net

Yury Makarychev
Toyota Technological Institute at Chicago, Chicago, IL, USA
https://ttic.uchicago.edu/~yury
yury@ttic.edu

Abstract

In this paper, we introduce the notion of a *certified algorithm*. Certified algorithms provide worst-case and beyond-worst-case performance guarantees. First, a γ-certified algorithm is also a γ-approximation algorithm – it finds a γ-approximation no matter what the input is. Second, it exactly solves γ-perturbation-resilient instances (γ-perturbation-resilient instances model real-life instances). Additionally, certified algorithms have a number of other desirable properties: they solve both maximization and minimization versions of a problem (e.g. Max Cut and Min Uncut), solve weakly perturbation-resilient instances, and solve optimization problems with hard constraints.

In the paper, we define certified algorithms, describe their properties, present a framework for designing certified algorithms, provide examples of certified algorithms for Max Cut/Min Uncut, Minimum Multiway Cut, k-medians and k-means. We also present some negative results.

2012 ACM Subject Classification Mathematics of computing → Combinatorial optimization; Theory of computation → Approximation algorithms analysis; Mathematics of computing → Approximation algorithms; Theory of computation → Facility location and clustering

Keywords and phrases certified algorithm, perturbation resilience, Bilu–Linial stability, beyond-worst-case analysis, approximation algorithm, integrality

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.49

Funding *Yury Makarychev*: Supported by NSF CCF-1718820.

1 Introduction

In this paper, we introduce and study certified algorithm, describe their properties, present a framework for designing certified algorithms, provide examples of certified algorithms for Max Cut/Min Uncut, Minimum Multiway Cut, k-medians and k-means. Recall the definition of an instance perturbation, which was given by Bilu and Linial [10].

▶ **Definition 1.** *Consider a combinatorial optimization or clustering problem. Suppose that every instance has a number of parameters $p_1, \dots, p_m > 0$; for example, if the problem is a graph partitioning problem, the parameters are edge weights; if it is a constraint satisfaction problem, the parameters are constraint weights; if it is a clustering problem, the parameters are distances between points.*

Let $\gamma \geq 1$. An instance $\mathcal{I}'$ is a γ-perturbation of $\mathcal{I}$ if it differs from $\mathcal{I}$ only by the values of the parameters, and the parameters $p'_1, \dots, p'_m$ of $\mathcal{I}'$ satisfy the following inequality

$$p_i \leq p'_i \leq \gamma p_i \quad \text{for every } i \tag{1}$$

alternatively, we may require that

$$p_i/\gamma \leq p'_i \leq p_i \quad \text{for every } i. \tag{2}$$

Note that if $\gamma = 1$, then $\mathcal{I}' = \mathcal{I}$. Loosely speaking, the closer γ to 1 is, the closer $\mathcal{I}'$ to $\mathcal{I}$ is. All problems we consider are scale invariant, so it will not matter whether we use formula (1) or (2). It will be convenient to use (1) for combinatorial optimization problems and (2) for clustering problems. The central definition of this paper is that of a certified algorithm.

▶ **Definition 2.** *A γ-certified solution for instance $\mathcal{I}$ is a pair $(\mathcal{I}', s^*)$, where $\mathcal{I}'$ is a γ-perturbation of $\mathcal{I}$ and s^* is an optimal solution for $\mathcal{I}'$. A γ-certified algorithm (or a γ-certified approximation algorithm) is an algorithm that finds a γ-certified solution.*[1]

The definition of a certified algorithm is inspired by the notions of smoothed analysis [18] and perturbation-resilience (also known as Bilu-Linial stability) [10]. Recall that in the smoothed analysis framework, we analyze the performance of an algorithm on a small random perturbation of the input instance. That is, we show that, after we randomly perturb the input, the algorithm can solve it with the desired accuracy in the desired time. A certified approximation algorithm perturbs the input instance *on its own* and then solves the obtained instance exactly. Importantly, the perturbation does not have to be random or small. Now, let us talk about perturbation resilience.

▶ **Definition 3** ([10])**.** *An instance $\mathcal{I}$ is γ-perturbation-resilient*[2] *if every γ-perturbation of $\mathcal{I}$ has the same optimal solution as $\mathcal{I}$ (we require that $\mathcal{I}$ have a unique optimal solution).*

Bilu and Linial [10] initiated the study of perturbation resilience in 2010. Perturbation-resilient instances model practical instances, and the model is particularly well-suited for capturing problems arising in machine learning. As Bilu-Linial (as well as other authors, see [6, 9]) argued most practically relevant instances should be perturbation-resilient. Since the seminal paper by Bilu and Linial, there has been a lot of research on algorithms for perturbation-resilient instances (see e.g., [10, 9, 6, 16, 7, 3, 11, 8], see also [15] for a survey of known results) and by now there are a number of algorithms for perturbation-resilient instances of various graph partitioning, clustering, and other problems. A closely related notion to perturbation resilience is that of weak perturbation resilience.

▶ **Definition 4** ([16])**.** *Consider an instance $\mathcal{I}$. Let s^* be an optimal solution and N be a set of solutions, which includes all optimal solutions. Assume that for every γ-perturbation $\mathcal{I}'$ of $\mathcal{I}$, solution s^* is better than every $s \notin N$ with respect to the $\mathcal{I}'$ objective. Then $\mathcal{I}$ is (γ, N)-weakly perturbation-resilient. We say that an algorithm solves a weakly perturbation-resilient instance $\mathcal{I}$, if given a (γ, N)-weakly perturbation-resilient instance, it finds a solution $s \in N$ (crucially, the algorithm does not know N.)*

Intuitively, N is the set of solutions that are close to s^* in some sense. Say, N might be the set of solutions that are at most ε far from s^* in some metric or have similar structural properties to s^*. Note that $(\gamma, \{s^*\})$-weak perturbation resilience is equivalent to γ-perturbation resilience. In general, the requirement that an instance $\mathcal{I}$ be weakly perturbation-resilient is somewhat less restrictive than the one that $\mathcal{I}$ be perturbation-resilient.

---

[1] We call the solution "certified", because, as we will see later, $\mathcal{I}'$ "certifies" that s^* is a γ-approximation.
[2] Perturbation-resilient instances are also known as Bilu–Linial stable instances.

1.1 Overview: Properties of Certified Algorithms

Consider a γ-certified algorithm $\mathcal{A}$ for a constraint satisfaction or graph partitioning problem (or any problem whose objective is homogeneous of degree 1)[3]. First of all, $\mathcal{A}$ is also a γ-approximation algorithm, it *always* finds a γ-approximate solution. Then it exactly solves γ-perturbation-resilient instances and solves (γ, N)-weakly perturbation-resilient instances. Thus, we have both worst-case and beyond-worst-case guarantees for $\mathcal{A}$. We believe that this property is very desirable in practice. In particular, if our instance is indeed perturbation-resilient, we will find an exact solution; if it is not, we will find a reasonably good approximate solution. Note that other algorithms do not satisfy this property: state-of-the-art approximation algorithms do not solve perturbation-resilient instances and state-of-the-art algorithms for perturbation-resilient instances do not provide a good approximation if the instance is not perturbation-resilient. Additionally, $\mathcal{A}$ also satisfies some other properties; we prove in Theorems 10, 11, and 13 that:

- **Worst-Case Guarantees.** Algorithm $\mathcal{A}$ finds a γ-approximate solution for any instance $\mathcal{I}$; further, it finds a γ-approximate solution for the complimentary objective. For example, if the problem is a constraint satisfaction problem (CSP), then $\mathcal{A}$ finds a solution that is a γ-approximation for the problems of (1) maximizing the weight of the satisfied constraints and (2) minimizing the weight of the unsatisfied constraints. Additionally, if the problem is a CSP, $\mathcal{A}$ is also a γ-approximation algorithm for the variant of the problem with hard constraints.
- **Beyond-Worst-Case Guarantees.** Algorithm $\mathcal{A}$ exactly solves γ-perturbation-resilient instances and solves (γ, N)-weakly perturbation-resilient instances.

▶ Remark 5. The running time of most algorithms that we consider will depend not only on the size of the instance but also on the magnitude of the parameters. In a sense, we will assume that all parameters are given in the unary. More precisely, let ρ be the ratio between the largest and smallest parameters (for constraint satisfaction and graph partitioning problems, ρ is the ratio between the largest and smallest constraint/edge/node weights; for clustering problems, ρ is the aspect ratio of the given metric space, the ratio between the largest and smallest distances). Then the running time will depend polynomially on the input size and ρ. Thus we will call our algorithms *pseudo-polynomial-time algorithms*.

▶ **Definition 6.** *We say that a certified algorithm runs in pseudo-polynomial time if its running time is polynomial in the size of the input n and $\rho = \frac{\max_i p_i}{\min_i p_i}$.*

1.2 Our Results

The main contribution of this paper is conceptual rather than technical. We introduce the notion of a certified algorithm and prove that certified algorithms satisfy the properties listed above. We believe that certified algorithms will prove useful in developing new algorithms for solving worst-case and beyond-worst-case instances. We also believe that even if one is primarily interested in designing an algorithm for solving perturbation-resilient instances, it is often more convenient to design a certified algorithm (as it is guaranteed to solve perturbation-resilient instances).

[3] That is, the value of the objective multiplies by α when we multiply all the parameters (exactly) by α.

We provide a general framework for designing polynomial-time certified algorithms for combinatorial optimization problems and give examples of algorithms for combinatorial optimization and clustering problems. We also present some negative results. In establishing these results, we heavily use techniques from papers on perturbation resilience [16, 3, 11, 8, 12]. Specifically, we give pseudo-polynomial-time γ-certified algorithms for

- Min Uncut and Max Cut with $\gamma = O(\sqrt{\log n}\log\log n)$ (*cf.* the state-of-the-art approximation algorithm for Min Uncut gives an $O(\sqrt{\log n})$ approximation [1]),
- Minimum Multiway Cut with $\gamma = 2 - 2/k + \varepsilon_n$ for every ε_n such that $\varepsilon_n > 1/\operatorname{poly}(n)$ (*cf.* the state-of-the-art approximation algorithm gives a ≈ 1.296 approximation [17])
- k-medians with $\gamma = 3 + \varepsilon$ for every fixed $\varepsilon > 0$ (*cf.* the state-of-the-art algorithm gives a ≈ 2.732 approximation [14]).

We also observe that the algorithm for $(1+\varepsilon)$ -perturbation resilient instances of Euclidean k-means and k-medians by Friggstad, Khodamoradi, and Salavatipour [12] is $(1+\varepsilon)$-certified (see Theorem 24). Additionally, we show that there are no polynomial-time or pseudo-polynomial-time $O(n^{1-\delta})$-certified algorithms for Minimum Vertex Cover, Set Cover, and Min 2-Horn Deletion if $\mathcal{P} \neq \mathcal{NP}$ (for every fixed $\delta > 0$).

▶ Note (Follow-up work). In a follow-up paper [2], Angelidakis, Awasthi, Blum, Chatziafratis, and Dan use our framework to design a number of new certified algorithms for such problems as Euclidean Maximum Independent Set and Node Multiway Cut.

2 Preliminaries

We start with formally defining what an instance of a combinatorial optimization problem is.

▶ **Definition 7.** *An instance of a combinatorial optimization problem is specified by a set of feasible solutions $\mathcal{S}$ (the solution space), a set of constraints $\mathcal{C}$, and constraint weights $w_c > 0$ for $c \in \mathcal{C}$. Typically, the solution space $\mathcal{S}$ is of exponential size and is not provided explicitly. Each constraint is a map from $\mathcal{S}$ to $\{0,1\}$. We say that a feasible solution $s \in \mathcal{S}$ satisfies a constraint c in $\mathcal{C}$ if $c(s) = 1$.*

We consider maximization and minimization objectives.

- *The maximization objective is to maximize the total weight of the satisfied constraints: find $s \in \mathcal{S}$ that maximizes $\operatorname{val}_{\mathcal{I}}(s) \equiv \sum_{c\in\mathcal{C}} w_c c(s)$.*
- *The minimization objective is to minimize the total weight of the unsatisfied constraints: find $s \in \mathcal{S}$ that minimizes $\sum_{c\in\mathcal{C}} w_c(1-c(s)) = w(\mathcal{C}) - \operatorname{val}_{\mathcal{I}}(s)$ (where $w(\mathcal{C}) = \sum_{c\in\mathcal{C}} w(c)$ is the total weight of all the constraints).*

We say that maximization and minimization are complementary objectives. Weights $\{w_c\}_{c\in\mathcal{C}}$ are the parameters of the instance in the sense of Definition 1.

Note that s^* is an optimal solution for $\mathcal{I}$ with the maximization objective if and only if it is an optimal solution for $\mathcal{I}$ with the minimization objective. Accordingly, $(\mathcal{I}', s)$ is a γ-certified solution for $\mathcal{I}$ with the maximization objective if and only if it is a γ-certified solution for $\mathcal{I}$ with the minimization objective.

▶ **Definition 8.** *An optimization problem $\mathcal{P}$ is a family $\mathcal{F}$ of instances. All instances in $\mathcal{F}$ have a fixed objective; either all of them have a maximization or all have a minimization objective. We assume that if instance $(\mathcal{S}, \mathcal{C}, w)$ is in $\mathcal{F}$, then so is $(\mathcal{S}, \mathcal{C}, w')$ for any choice of weights $w_c > 0$.*

This definition captures various constraint satisfaction, graph partitioning, and covering problems. Consider for example Min Uncut. In Min Uncut, the goal is to find a cut $(S, \bar{S})$ in a given graph $G = (V, E, w)$ that minimizes the total weight of the uncut edges (edges that

connect vertices within S or within $\bar{S}$). For Min Uncut, $\mathcal{S}$ is the set of all cuts in G. There is a constraint c_e for every edge $e \in E$; a cut satisfies constraint c_e if and only if it cuts edge e. The objective is to minimize $\sum_{c\in\mathcal{C}} w_c(1 - c((S, \bar{S})))$. The objective for the complementary problem, Max Cut, is $\sum_{c\in\mathcal{C}} w_c c((S, \bar{S}))$.

There are also certified algorithms for clustering problems. In this paper, we describe a $(3 + \varepsilon)$-certified algorithm for k-medians and note that algorithms for Euclidean k-means and k-medians from [12] are $(1 + \varepsilon)$-certified. Recall the definition of k-medians.

▶ **Definition 9.** *In k-medians, we are a given a set of points X, metric d on X, which satisfies triangle inequalities, and a parameter $k \geq 1$. The cost of a cluster $C \subset X$ is*

$$\operatorname{cost} C = \min_{c\in X} \sum_{u\in C} d(u, c).$$

The goal is to partition X into k clusters $C_1, \dots, C_k$ so as to minimize their total cost $\sum_{i=1}^{k} \operatorname{cost} C_i$. The parameters of the problem (in the sense of Definition 1) are the pairwise distances $d(u, v)$.

We say that c is an optimal center for C, if $c \in \arg\min_{c\in X} \sum d(u, c)$. Note that a set of centers $c_1, \dots, c_k$ defines a clustering $C_1, \dots, C_k$ of X; namely, $C_1, \dots, C_k$ is the Voronoi partition for $c_1, \dots, c_k$ (if there are ties, several partitions may correspond to the same set of centers).

A γ-perturbation of an instance (X, d) is an instance (X, d'), such that $\frac{1}{\gamma} d(u, v) \leq d'(u, v) \leq d(u, v)$ for every $u, v \in X$. Note that we do not require that d' satisfy triangle inequalities (if we did, we would get the definition of a metric perturbation; see [3] for details).

3 Properties of Certified Algorithms

In this section, we prove that certified algorithms satisfy the properties we described in Section 1.

▶ **Theorem 10.** *Consider a γ-certified algorithm $\mathcal{A}$ for a combinatorial optimization problem.*

- *$\mathcal{A}$ finds a solution that is a γ-approximate solution w.r.t. both the maximization and minimization objectives.*
- *If the instance is γ-perturbation-resilient, $\mathcal{A}$ finds the optimal solution. If it is (γ, N)-weakly perturbation-resilient, $\mathcal{A}$ finds a solution in N.*

Proof. Consider an instance $\mathcal{I}$. Denote its optimal solution by s^*. Denote the certified solution found by $\mathcal{A}$ by $(\mathcal{I}', s')$. For each $c \in \mathcal{C}$, let w_c and w'_c be its weights in $\mathcal{I}$ and $\mathcal{I}'$, respectively.

I. First, we prove that the algorithm always gives a γ-approximation for both objectives. Consider the maximization objective. We have,

$$\operatorname{val}_{\mathcal{I}}(s') = \sum_{c\in\mathcal{C}} w_c c(s') \geq \sum_{c\in\mathcal{C}} \frac{w'_c}{\gamma} c(s') = \frac{1}{\gamma} \sum_{c\in\mathcal{C}} w'_c c(s')$$
$$\overset{(\star)}{\geq} \frac{1}{\gamma} \sum_{c\in\mathcal{C}} w'_c c(s^*) \geq \frac{1}{\gamma} \sum_{c\in\mathcal{C}} w_c c(s^*) = \frac{\operatorname{val}_{\mathcal{I}}(s^*)}{\gamma}$$

where $(\star)$ holds since s' is an optimal solution for $\mathcal{I}'$. We conclude that s' is a γ-approximate solution for the maximization objective. Similarly, we analyze the minimization objective.

$$\sum_{c\in\mathcal{C}} w_c(1 - c(s')) \leq \sum_{c\in\mathcal{C}} w'_c(1 - c(s')) \leq \sum_{c\in\mathcal{C}} w'_c(1 - c(s^*)) \leq \gamma \sum_{c} w_c(1 - c(s^*)).$$

	problem $\mathcal{P}_{hard}$	problem $\mathcal{P}$
unperturbed instance	$\mathcal{I}$	$\mathcal{J}$
perturbed instance	$\mathcal{I}'$	$\mathcal{J}'$

Figure 1 Instances $\mathcal{I}$, $\mathcal{J}$, $\mathcal{I}'$, and $\mathcal{J}'$. If s is a feasible solution for $\mathcal{I}'$, $\mathrm{val}_{\mathcal{I}'}(s) = \mathrm{val}_{\mathcal{J}'}(s) - W_H$.

II. Now, assume that $\mathcal{I}$ is γ-perturbation-resilient. By the definition of perturbation resilience, $\mathcal{I}$ and $\mathcal{I}'$ have the same optimal solution. Thus, s^* is an optimal solution not only for $\mathcal{I}'$ but also for $\mathcal{I}$. Finally, assume that $\mathcal{I}$ is (γ, N)-weakly perturbation-resilient. Since $\mathcal{I}$ is (γ, N)-weakly perturbation-resilient and $\mathcal{I}'$ is a γ-perturbation of $\mathcal{I}$, we get form Definition 4 that either $s' \in N$ or s^* is better than s' w.r.t. to the $\mathcal{I}'$ objective. The latter is not possible, since s' is an optimal solution for $\mathcal{I}'$. We conclude that $s' \in N$. ◀

Consider an instance of an optimization problem. We may choose a subset of constraints $H \subset \mathcal{C}$ and require that all of them are satisfied. We call them *hard constraints* and the obtained instance an instance with hard constraints. Formally, given an instance $(\mathcal{S}, \mathcal{C}, w)$ and a subset of constraints H, we define the correspondent instance $(\mathcal{S}', \mathcal{C}', w)$ with hard constraints as follows: $\mathcal{S}' = \{a \in \mathcal{S} : c(s) = 1 \text{ for every } c \in H\}$; $\mathcal{C}' = \mathcal{C} \setminus H$; $w'(c) = w(c)$ for $c \in \mathcal{C}'$.

▶ **Theorem 11.** *Assume that $\gamma = \gamma_n$ is at most polynomial in n. If there is a pseudo-polynomial-time γ-certified algorithm for a problem $\mathcal{P}$, then there is also a pseudo-polynomial-time γ-certified algorithm for a variant $\mathcal{P}_{hard}$ of $\mathcal{P}$ with hard constraints.*

Proof. Consider an instance $\mathcal{I}$ of $\mathcal{P}_{hard}$. Let H be the set of hard constraints and S be the set of soft constraints. We transform $\mathcal{I}$ to an instance $\mathcal{J}$ of $\mathcal{P}$ by setting the weight of every hard constraint $c \in H$ to $(\gamma+1)W$ where $W = \sum_{c \in S} w_c$ is the total weight of the soft constraints (see Figure 1). Note that the parameter $\rho_{\mathcal{J}}$ for $\mathcal{J}$ (the ratio between the largest and smallest constraint weights in $\mathcal{J}$; see Definition 6) is polynomial in $\rho_{\mathcal{I}}$ and the input size:

$$\rho_{\mathcal{J}} = \frac{(\gamma+1)W}{\min_{c \in \mathcal{C}} w_c} \leq \frac{|S|(\gamma+1)\max_{c \in \mathcal{C}} w_c}{\min_{c \in \mathcal{C}} w_c} \leq |S|(\gamma+1)\rho_{\mathcal{I}}.$$

We run the algorithm for $\mathcal{P}$ on input $\mathcal{J}$ and get a certified solution $(\mathcal{J}', s^*)$. Let $W_H = \sum_{c \in H} w_c^{\mathcal{J}'}$, where $w_c^{\mathcal{J}'}$ is the weight of c in $w_c^{\mathcal{J}'}$. Every feasible solution s for $\mathcal{I}$ satisfies all the constraints in H and thus $\mathrm{val}_{\mathcal{J}'}(s) \geq W_H$. In particular, if $\mathcal{I}$ has a feasible solution s, then $\mathrm{val}_{\mathcal{J}'}(s^*) \geq \mathrm{val}_{\mathcal{J}'}(s) \geq W_H$. Conversely, every solution s for $\mathcal{J}'$ with $\mathrm{val}_{\mathcal{J}'}(s) \geq W_H$ satisfies all the hard constraints (since the total weight of the soft constraints is less than the weight of any hard constraint in $\mathcal{J}'$) and thus is a feasible solution for $\mathcal{I}$.

If $\mathrm{val}_{\mathcal{J}'}(s^*) < W_H$, we report that $\mathcal{I}$ has no feasible solution. Otherwise, s^* is a feasible solution for $\mathcal{I}$. We let $\mathcal{I}'$ be a perturbation of $\mathcal{I}$, in which all the soft constraints have the same weights as in $\mathcal{J}'$. Observe that for every feasible solution s of $\mathcal{I}'$, $\mathrm{val}_{\mathcal{I}'}(s) = \mathrm{val}_{\mathcal{J}'}(s) - W_H$. It follows that s^* is an optimal solution for $\mathcal{I}'$. We output $(\mathcal{I}', s^*)$. ◀

▶ **Corollary 12.** *Consider a problem $\mathcal{P}$ and its variant with hard constraints $\mathcal{P}_{hard}$. Let γ_n be at most polynomial in n (the instance size). If there is a pseudo-polynomial-time γ_n-certified algorithm for $\mathcal{P}$, then there are pseudo-polynomial-time γ_n-approximation algorithms for maximization and minimization variants of $\mathcal{P}_{hard}$.*

We prove an analog of Theorem 10 for k-medians and k-means in Appendix (its proof is very similar to that of Theorem 10).

▶ **Theorem 13.** *Consider a γ-certified algorithm $\mathcal{A}$ for k-medians or k-means. If $\mathcal{A}$ is for k-medians, then $\mathcal{A}$ is a γ-approximation algorithm; if $\mathcal{A}$ is for k-means, then $\mathcal{A}$ is a γ^2-approximation algorithm. If the instance is γ-perturbation-resilient, $\mathcal{A}$ finds the optimal solution. If it is (γ, N)-weakly perturbation-resilient, $\mathcal{A}$ finds a solution in N.*

4 Designing Certified Algorithms

4.1 Iterative Improvement Algorithms

We use an iterative approach to design certified algorithms: our certified algorithms start with an arbitrary solution and then iteratively improve it. The approach resembles that of local search, except that improvements will not necessarily be local. In this approach, the key component of a certified algorithm is a procedure for improving the current solution; namely, a procedure for solving the following task.

▶ **Task 14.** *Given an instance $\mathcal{I}$ and the current solution s,*
- *either find a new solution s', which is better than s (w.r.t. to the $\mathcal{I}$ objective), or*
- *find a γ certified solution $(\mathcal{I}', s)$.*

▷ Claim 15. Consider a combinatorial optimization or clustering problem. Let $\varepsilon = \varepsilon_n > 1/\operatorname{poly}(n)$. Assume the following.
1. There is a polynomial-time algorithm for Task 14 with $\gamma = \gamma_n$.
2. There is polynomial-time algorithm that finds a feasible solution.

Then there is a pseudo-polynomial-time $(1+\varepsilon_n)\gamma_n$-certified algorithm for the problem.

Proof. Let $p_{min} = \min_i p_i$ be the smallest parameter and $\varepsilon' = \varepsilon/2$. First, we apply a preprocessing step, where we round all parameters p_i to multiples of $q = \varepsilon' p_{min}$ as follows:

$$p'_i = (\lceil p_i/q \rceil + 1)q.$$

It is easy to see that $q \le p'_i - p_i \le 2q$. Thus, instance $\mathcal{I}'$ with parameters p'_i is a $(1+\varepsilon)$ perturbation of $\mathcal{I}$.

If the problem is k-medians (or for that matter another clustering problem), then parameters p_i are distances $d(u,v)$, satisfying triangle inequalities. Then the new distances $d'(u,v)$ also satisfy triangle inequalities:

$$d'(u,v) + d'(v,w) \ge (d(u,v)+q) + (d(v,w)+q) \ge d(u,w) + 2q \ge d'(u,w).$$

Now we proceed as follows. We find a feasible solution for $\mathcal{I}'$ and then iteratively improve it using the procedure for Task 14, until the procedure finds a γ-certified solution $(\mathcal{I}'', s)$ for $\mathcal{I}'$. Clearly, $\mathcal{I}''$ is a $(1+\varepsilon)\gamma$-perturbation of $\mathcal{I}$. Thus, $(\mathcal{I}'', s)$ is a $(1+\varepsilon)\gamma$-certified solution for $\mathcal{I}$.

It remains to prove that the algorithm runs in pseudo-polynomial time. To do so, we need to upper bound the number of iterations. Assume that the problem is a combinatorial optimization problem. Consider the maximization objective. Initially the value of the problem is non-negative. It increases by at least q in every iteration. When the program terminates, it is at most $\sum_{c\in\mathcal{C}} w'_c \le (\rho w_{\min} + 2q)|\mathcal{C}|$ (where ρ as in Definition 6). Thus the algorithm performs at most $\frac{(\rho w_{\min}+2q)|\mathcal{C}|}{q} \le 2(\rho/\varepsilon+1)|\mathcal{C}|$ iterations, which is polynomial in $|\mathcal{C}|$, ρ, and $1/\varepsilon$.

If the problem is k-medians, the cost of the initial clustering is at most $n \max_{u,v\in X} d'(u,v)$. It decreases by at least q in every iteration. The cost of the obtained clustering is non-negative. It is easy to see that the number of iterations is polynomial in n, ρ, and $1/\varepsilon$.

We conclude that the algorithm runs in pseudo-polynomial time. ◁

4.2 Designing Certified Algorithms for Combinatorial Optimization Problems

Consider a combinatorial optimization problem. We show that in order to solve Task 14, it suffices to solve the following task.

▶ **Task 16.** *Assume that we are given an instance $\mathcal{I} = (\mathcal{S}, \mathcal{C}, w)$, a partition of its constraints $\mathcal{C} = C_1 \cup C_2$, and a parameter $\gamma \geq 1$. We need to either*
- *find $s \in \mathcal{S}$ such that $\gamma \sum_{c \in C_1} w_c c(s) > \sum_{c \in C_2} w_c (1 - c(s))$, or*
- *report that for every $s \in \mathcal{S}$: $\sum_{c \in C_1} w_c c(s) \leq \sum_{c \in C_2} w_c (1 - c(s))$.*

(Note that the above options are not mutually exclusive.)

▶ **Theorem 17.** *Assume that (1) there is a polynomial-time procedure for Task 16 with $\gamma = \gamma_n$ and (2) there is a polynomial-time algorithm that finds some $s \in \mathcal{S}$. Then there exists a pseudo-polynomial-time $(\gamma_n + \varepsilon_n)$-certified algorithm for the problem (for every $\varepsilon_n > 1/\operatorname{poly}(n)$).*

Proof. By Claim 15, it suffices to design an algorithm for Task 14. Given a solution s, we will either find a better solution s' or return a certified solution $(\mathcal{I}', s)$. Let $C_1 = \{c \in \mathcal{C} : c(s) = 0\}$ and $C_2 = \{c \in \mathcal{C} : c(s) = 1\}$ be the sets of unsatisfied and satisfied constraints, respectively. Define weights w' as follows:

$$w'_c = \begin{cases} w_c, & \text{if } c \in C_1 \\ \gamma w_c, & \text{if } c \in C_2 \end{cases}$$

We run the procedure for Task 16 on instance $\mathcal{I}' = (\mathcal{S}, \mathcal{C}, w')$. Consider two cases. Assume first that the procedure returns a solution s' such that $\gamma \sum_{c \in C_1} w'_c c(s') > \sum_{c \in C_2} w'_c (1 - c(s'))$. We get that

$$\sum_{c \in C_1} w_c c(s') > \sum_{c \in C_2} w_c (1 - c(s'))$$

and thus

$$\operatorname{val}_{\mathcal{I}}(s') = \sum_{c \in C_1 \cup C_2} w_c c(s') > \sum_{c \in C_2} w_c = \operatorname{val}_{\mathcal{I}}(s).$$

In this case, we return s'. Assume now that the procedure reports that for every solution s':

$$\sum_{c \in C_1} w'_c c(s') \leq \sum_{c \in C_2} w'_c (1 - c(s'))$$

or, equivalently,

$$\operatorname{val}_{\mathcal{I}'}(s') = \sum_{c \in C_1 \cup C_2} w'_c c(s') \leq \sum_{c \in C_2} w'_c = \operatorname{val}_{\mathcal{I}'}(s).$$

Then s is an optimal solution for $\mathcal{I}'$. We return a γ-certified solution $(\mathcal{I}', s)$. ◀

4.3 Certified Algorithm via Convex Relaxations

We describe how to design certified algorithms for combinatorial optimization problems using convex relaxations. Consider a problem and a convex relaxation for it. We refer to problem solutions $s \in \mathcal{S}$ as *combinatorial* solutions and relaxation solutions x as *fractional* solutions;

we say that x is *integral* if it corresponds to a combinatorial solution $s \in \mathcal{S}$. We assume that in the relaxation we have a variable x_c for each constraint c so that $x_c = c(s)$ for every integral solution x and corresponding combinatorial solution s. The relaxations objective is to maximize $\mathrm{fval}(x) = \sum_{c\in\mathcal{C}} w_c x_c$ or (equivalently) minimize $\sum_{c\in\mathcal{C}} w_c(1-x_c)$.

Consider a *randomized* rounding scheme $\mathcal{R}$ that given a fractional solution x outputs a combinatorial solution $\mathcal{R}(x)$. Rounding schemes are widely used for designing approximation algorithms. When designing an algorithm for a *maximization* objective, one typically wants the rounding scheme to satisfy the following approximation condition.

- **Approximation Condition.** The probability that each constraint $c \in \mathcal{C}$ is satisfied by $\mathcal{R}(x)$ is at least x_c/α (the probability is over the random choices made by $\mathcal{R}$).

If $\mathcal{R}$ satisfies this condition, it gives an α approximation for the maximization objective (in expectation). On the other hand, when designing an algorithm for a *minimization* objective, one wants the rounding scheme to satisfy the co-approximation condition.

- **Co-approximation Condition.** The probability that each constraint $c \in \mathcal{C}$ is unsatisfied by $\mathcal{R}(x)$ is at most $\beta(1-x_c)$.

If $\mathcal{R}$ satisfies this condition, it gives a β approximation for the minimization objective (in expectation). Following [16, 3], we consider rounding schemes that simultaneously satisfy the approximation and co-approximation conditions.

▶ **Definition 18.** *We say that a rounding scheme $\mathcal{R}$ is an (α,β)-rounding if it simultaneously satisfies the approximation and co-approximation conditions with parameters α and β.*

We note that (α,β)-rounding schemes have been shown to be very useful for solving perturbation-resilient and weakly perturbation-resilient instances [16, 3]. In particular, if there is an (α,β)-rounding scheme, then the relaxation is integral for $(\alpha\beta)$-perturbation-resilient instances [16]. We now show that (α,β)-rounding schemes can be used for designing certified algorithms.

▶ **Theorem 19.** *Assume that there exists an (α,β)-rounding scheme $\mathcal{R}$. Additionally, assume that the support of $\mathcal{R}$ is of polynomial size and can be found in polynomial time.*

Then there exists a pseudo-polynomial-time certified $(\gamma+\varepsilon_n)$-approximation algorithm for the problem where $\gamma = \alpha\beta$ (for any $\varepsilon > 1/\operatorname{poly}(n)$). Further, the algorithm outputs a certified solution $(\mathcal{I}', s^*)$ such that s^* is an optimal solution not only for $\mathcal{I}'$ but also for the relaxation for $\mathcal{I}'$.

Proof. By Theorem 17, it suffices to design a polynomial-time procedure for solving Task 16. First, we solve the convex relaxation for the problem and obtain an optimal fractional solution $x = x^*$. If $\sum_{c\in C_1} w_c x_c \le \sum_{c\in C_2} w_c(1-x_c)$, then for every $s \in \mathcal{S}$

$$\sum_{c\in C_1} w_c c(s) + \sum_{c\in C_2} w_c c(s) \le \sum_{c\in C_1} w_c x_c + \sum_{c\in C_2} w_c x_c \le \sum_{c\in C_2} w_c(1-x_c) + \sum_{c\in C_2} w_c x_c = \sum_{c\in C_2} w_c. \quad (3)$$

So we report that $\sum_{c\in C_1} w_c c(s) \le \sum_{c\in C_2} w_c(1-c(s))$ for every s (option 2). Note that in this case, the certified algorithm from Theorem 17 returns a certified solution $(\mathcal{I}, s^*)$ of value $\mathrm{val}_{\mathcal{I}'}(s^*) = w(C_2) \equiv \sum_{c\in C_2} w_c$. Eq. (3) shows that the value of every fractional solution (let alone integral) is at most $w(C_2)$. Thus s^* is an optimal solution not only for $\mathcal{I}'$ but also for the relaxation for $\mathcal{I}'$.

Assume now that $\sum_{c\in C_1} w_c x_c > \sum_{c\in C_2} w_c(1-x_c)$. We apply rounding scheme $\mathcal{R}$ and obtain a solution $\mathcal{R}(x)$. From the approximation and co-approximation conditions, we get

$$\mathbf{E}\left[\gamma \sum_{c\in C_1} w_c c(\mathcal{R}(x)) - \sum_{c\in C_2} w_c(1-c(\mathcal{R}(x)))\right] \ge \frac{\gamma}{\alpha}\sum_{c\in C_1} w_c x_c - \beta \sum_{c\in C_2} w_c(1-x_c) > 0.$$

Here, we used that $\gamma = \alpha\beta$. Thus for some solution s in the support of $\mathcal{R}(x)$, we have $\gamma\sum_{c\in C_1} w_c c(s) > \sum_{c\in C_2} w_c(1-c(s))$. We find and return such a solution s. ◀

We note that it is sufficient to design a rounding procedure only for solutions that are close to integral solutions.

▶ **Definition 20.** *Let us say that a fractional solution x is δ-close to an integral if $x = (1-\delta)x^{int} + \delta x^{frac}$ for some integral solution x^{int} and fractional solution x^{frac}. Rounding scheme $\mathcal{R}$ is a δ-local (α, β)-rounding if it is defined and satisfies the approximation and co-approximation conditions for fractional solutions x that are δ-close to an integral solution.*

It turns out that it is sufficient to have a δ-local rounding scheme (for any $\delta > 0$) in Theorem 19. To see that, we slightly change the proof of Theorem 19. We first find an optimal fractional solution x^* and then apply the rest of the argument to solution $x = \delta x^* + (1-\delta)x^s$ (where x^s is the fractional solution corresponding to s).

Finally, we note that if the support of $\mathcal{R}$ is not of a polynomial size, we can get a *randomized* certified algorithm. To do so, instead of trying out all solutions s in the support of $\mathcal{R}(x)$, we apply $\mathcal{R}$ to x sufficiently many times and let s be the best of the obtained solutions (if we use a δ-local rounding scheme, we need that $\delta > 1/\operatorname{poly}(n)$).

5 Examples of Certified Algorithms for Optimization Problems

▶ **Theorem 21.**

I. *There exists a pseudo-polynomial-time $(1+\varepsilon_n)\alpha_n$-certified algorithm for Min Uncut and Max Cut (these problems are complementary), where $\alpha_n = O(\sqrt{\log n}\log\log n)$ is the approximation factor for Sparsest Cut with Non-uniform Demands from [4] and $\varepsilon_n > 1/\operatorname{poly}(n)$.*

II. *There exists a pseudo-polynomial-time $(2 - 2/k + \varepsilon_n)$-certified algorithm for Minimum Multiway Cut.*

Proof.

I. We show how to solve Task 16 in polynomial time. Recall that in our formulation of Min Uncut, $c_e((S,\bar{S})) = 1$ if edge e is cut. Let $E_1 = \{e \in E : c_e \in C_1\}$ and $E_2 = \{e \in E : c_e \in C_2\}$; denote the weight of the edges in E_i cut by $(S,\bar{S})$ by $w(E_i(S,\bar{S}))$. Let $\phi(S) = \frac{w(E_2(S,\bar{S}))}{w(E_1(S,\bar{S}))}$. Then our goal is to either find a cut $(S,\bar{S})$ such that $\phi(S) < \gamma$ or report that $\phi(S) \geq 1$ for every $(S,\bar{S})$. Now the problem of minimizing $\phi(S)$ over all cuts $(S,\bar{S})$ is the same as finding the sparsest cut with non-uniform demands in graph (V, E_2) with edge capacities w, demand pairs E_1, and demand weights w. We run the approximation algorithm for Sparsest Cut and get a cut $(S,\bar{S})$ that approximately – within a factor of γ – minimizes $\phi(S)$. If $\phi(S) < \gamma$, we report cut $(S,\bar{S})$; otherwise, we report that $\phi(S') \geq 1$ for every cut $(S',\bar{S}')$.

II. Consider the LP relaxation for Minimum Multiway Cut by Călinescu, Karloff, and Rabani. It is shown in [3] that there exists a δ-local (α,β)-rounding procedure for it with $\alpha\beta = 2 - 2/k$. It follows from Theorem 19, that there is a $(2 - 2/k + \varepsilon_n)$-certified algorithm. ◀

6 $(3+\varepsilon)$-Certified Local Search Algorithm for k-medians

In this section, we consider k-medians. We show that a local search algorithm is $(3+\varepsilon)$-certified. We note that our analysis is very similar to that in [11, 8].

We first apply the preprocessing step form Theorem 17 (where we round all distances to multiples of some q). Then, we consider an arbitrary set of centers $c_1, \ldots, c_k$ and the corresponding clustering $C_1, \ldots, C_k$. Then, at each iteration, we go over all possible swaps

of size r: we swap r centers among $c_1, \dots, c_k$ with r points outside of $c_1, \dots, c_k$. We choose a swap that decreases the cost of the clustering, if there is one, and recompute $C_1, \dots, C_k$. If there is no such swap, we terminate and output a certified solution $((X, d'), (C_1, \dots, C_k))$, where $d'(u, v) = \frac{1}{\gamma} d(u, v)$ if $u = c_i$ and $v \in C_i$ for some i (or the other way around), and $d'(u, v) = d(u, v)$, otherwise.

▶ **Theorem 22.** *The ρ-local search algorithm for k-medians (described above) is a pseudo-polynomial-time $(3 + O(1/\rho))$-certified algorithm.*

Proof. We use Theorem 14. It guarantees that the algorithm runs in pseudo-polynomial time. We need to show that when the algorithm terminates it indeed outputs a certified solution. Suppose that the algorithm outputs a clustering with centers $L = \{l_1, \dots, l_k\}$.

Consider an arbitrary set of centers $S = \{s_1, \dots, s_k\}$. We need to show that the cost of the k-median clustering with centers in L is at most the cost of the k-median clustering with centers in S with respect to the perturbed distance function d'. Let $l(u)$ and $s(u)$ be the closest centers to point u in L and S respectively with respect to d; and let $l'(u)$ and $s'(u)$ be the closest centers to point u in L and S respectively with respect to d'. Our goal is to prove that

$$\sum_{u \in X} d'(u, l'(u)) \le \sum_{u \in X} d'(u, s'(u)). \tag{4}$$

Observe that for every point $u \in X$, we have $d(u, v) = d'(u, v)$ for all v but $v = l(u)$. Thus, $l'(u) = l(u)$ and $d'(u, l'(u)) = d(u, l(u))/\gamma$. Consequently, the left hand side of (4) equals $\sum_{u \in X} d(u, l(u))/\gamma$. Similarly, $s'(u) = s(u)$ and $d'(u, s'(u)) = d(u, s(u))$ if $l(u) \notin S$. However, if $l(u) \in S$, then $d'(u, s'(u)) = \min\big(d(u, s(u)), d(u, l(u))/\gamma\big)$ as, in this case, the optimal center for u in S w.r.t. d' can be $l(u)$.

Let us split all vertices in X into two groups $A = \{u : l(u) \in S\}$ and $B = \{u : l(u) \notin S\}$. Then, for $u \in A$, we have $d'(u, s'(u)) = \min\big(d(u, s(u)), d(u, l(u))/\gamma\big)$; and for for $u \in B$, we have $d'(u, s'(u)) = d(u, s(u))$. Thus, inequality (4) is equivalent to

$$\sum_{u \in X} \frac{d(u, l(u))}{\gamma} \le \sum_{u \in A} \min\left(d(u, s(u)), \frac{d(u, l(u))}{\gamma}\right) + \sum_{u \in B} d(u, s(u)),$$

which after multiplying both parts by γ can be written as

$$\sum_{u \in X} d(u, l(u)) \le \sum_{u \in A} \min\Big(\gamma d(u, s(u)), d(u, l(u))\Big) + \sum_{u \in B} \gamma d(u, s(u)). \tag{5}$$

For $u \in A$, we have $d(u, s(u)) \le d(u, l(u))$ since both $s(u)$ and $l(u)$ are in S and $s(u) = \arg\min_{v \in S} d(u, v)$. Thus, $\min\big(\gamma d(u, s(u)), d(u, l(u))\big) \ge d(u, s(u))$. Consequently, inequality (5) follows from the following theorem from [11] (see also [5] and [13]).

▶ **Theorem 23** (Local Approximation; [11])**.** *Let L be a r-locally optimal set of centers with respect to a metric d and S be an arbitrary set of k centers. Define sets A and B as above. Then,*

$$\sum_{u \in X} d(u, l(u)) \le \sum_{u \in A} d(u, s(u)) + \gamma \sum_{u \in B} d(u, s(u)),$$

for some $\gamma = 3 + O(1/r)$. ◀

7 Euclidean k-means and k-medians

We note that the algorithm for $(1+\varepsilon)$-perturbation-resilient instances of Euclidean k-means and k-medians (in a fixed dimensional space) by Friggstad, Khodamoradi, and Salavatipour [12] is $(1+\varepsilon)$-certified. The algorithm finds a solution $\mathcal{S}$ and perturbed metric δ' on $X \cup \mathcal{S}$ such that (see Lemma 2.2 in [12])

$$\text{cost}'(\mathcal{S}) \leq \text{cost}'(\mathcal{O}),$$

where cost$'$ is the cost of the clustering w.r.t the perturbed metric δ'. In [12], $\mathcal{O}$ is the optimal solution for the non-perturbed instance; however, the proof does not use that $\mathcal{O}$ is an optimal solution and goes through if $\mathcal{O}$ is an arbitrary solution. Thus, Friggstad, Khodamoradi, and Salavatipour proved that their algorithm finds a solution $\mathcal{S}$ (specified by the list of centers), which is optimal w.r.t. the perturbed distances δ'. We get the following theorem.

▶ **Theorem 24.** *For every fixed $\varepsilon > 0$ and $d \geq 1$, there exist $(1+\varepsilon)$-certified algorithms for Euclidean instances of k-means and k-medians in $\mathbb{R}^d$ (namely, the local search algorithms from [12]). The algorithms run in time polynomial in the bit complexity of the input.*

8 Negative Results

In this section, we present several negative results for certified algorithms. They immediately follow from the properties of certified algorithm we saw in Section 3. We note that similar negative results were shown in [3] for robust algorithms for perturbation-resilient instances.

▶ **Theorem 25.** *There are no pseudo-polynomial-time $O(n^{1-\delta})$-certified algorithms for Minimum Vertex Cover, Set Cover, and Min 2-Horn Deletion if $\mathcal{P} \neq \mathcal{NP}$ (for every fixed $\delta > 0$).*

Proof. Let $\gamma = cn^{1-\delta}$ be the hardness of Maximum Independent Set (MIS) [19].

I. According o Theorem 10, if there were a pseudo-polynomial-time or polynomial-time γ-certified algorithm for Vertex Cover, then there would be a polynomial-time γ-approximation algorithm for Maximum Independent Set (the problem complementary to Minimum Vertex Cover).

II. Since each Vertex Cover instance is also a Set Cover instance, a certified algorithm for Set Cover would also be a certified algorithm for Vertex Cover.

III. Observe that Max 2-Horn SAT with hard constraints is more difficult than MIS. An instance (G, V, E) of MIS is equivalent to the following instance of Max 2-Horn SAT with hard constraints: there is a variable x_u for every $u \in V$, a hard constraint $x_u \vee x_v$ for every $(u, v) \in E$, and a soft constraint x_u. By Corollary 12, since there is no γ-approximation for MIS, there is no polynomial-time algorithm for γ-perturbation-resilient instances of Min 2-Horn Deletion. ◀

References

1 Amit Agarwal, Moses Charikar, Konstantin Makarychev, and Yury Makarychev. $O(\sqrt{\log n})$ approximation algorithms for Min UnCut, Min 2CNF Deletion, and directed cut problems. In *Proceedings of the Symposium on Theory of Computing*, pages 573–581, 2005.

2 Haris Angelidakis, Pranjal Awasthi, Avrim Blum, Vaggos Chatziafratis, and Chen Dan. Bilu-Linial stability, certified algorithms and the Independent Set problem. In *Proceedings of the European Symposium on Algorithms*, 2019.

3 Haris Angelidakis, Konstantin Makarychev, and Yury Makarychev. Algorithms for stable and perturbation-resilient problems. In *Proceedings of the Symposium on Theory of Computing*, pages 438–451, 2017.

4 Sanjeev Arora, James Lee, and Assaf Naor. Euclidean distortion and the sparsest cut. *Journal of the American Mathematical Society*, 21(1):1–21, 2008.

5 Vijay Arya, Naveen Garg, Rohit Khandekar, Adam Meyerson, Kamesh Munagala, and Vinayaka Pandit. Local search heuristics for k-median and facility location problems. *SIAM Journal on computing*, 33(3):544–562, 2004.

6 Pranjal Awasthi, Avrim Blum, and Or Sheffet. Center-based clustering under perturbation stability. *Information Processing Letters*, 112(1-2):49–54, 2012.

7 Maria-Florina Balcan, Nika Haghtalab, and Colin White. k-Center Clustering Under Perturbation Resilience. In *International Colloquium on Automata, Languages, and Programming*, 2016.

8 Maria-Florina Balcan and Colin White. Clustering under local stability: Bridging the gap between worst-case and beyond worst-case analysis. *arXiv preprint*, 2017. `arXiv:1705.07157`.

9 Yonatan Bilu, Amit Daniely, Nati Linial, and Michael Saks. On the practically interesting instances of MAXCUT. In *International Symposium on Theoretical Aspects of Computer Science*, 2013.

10 Yonatan Bilu and Nathan Linial. Are Stable Instances Easy? In *Innovations in Computer Science*, pages 332–341, 2010.

11 Vincent Cohen-Addad and Chris Schwiegelshohn. On the local structure of stable clustering instances. In *Proceedings of the Symposium on Foundations of Computer Science*, pages 49–60, 2017.

12 Zachary Friggstad, Kamyar Khodamoradi, and Mohammad R. Salavatipour. Exact Algorithms and Lower Bounds for Stable Instances of Euclidean K-means. In *Proceedings of the Symposium on Discrete Algorithms*, pages 2958–2972, 2019.

13 Anupam Gupta and Kanat Tangwongsan. Simpler analyses of local search algorithms for facility location. *arXiv preprint*, 2008. `arXiv:0809.2554`.

14 Shi Li and Ola Svensson. Approximating k-median via pseudo-approximation. *SIAM Journal on Computing*, 45(2):530–547, 2016.

15 Konstantin Makarychev and Yury Makarychev. Bilu–Linial Stability. In T. Hazan, G. Papandreou, and D. Tarlow, editors, *Perturbations, Optimization, and Statistics*, chapter 13. MIT Press, 2016.

16 Konstantin Makarychev, Yury Makarychev, and Aravindan Vijayaraghavan. Bilu-Linial stable instances of Max Cut and Minimum Multiway Cut. In *Proceedings of the Symposium on Discrete Algorithms*, pages 890–906, 2014.

17 Ankit Sharma and Jan Vondrák. Multiway Cut, Pairwise Realizable Distributions, and Descending Thresholds. In *Proceedings of the Symposium on Theory of Computing*, 2014.

18 Daniel A Spielman and Shang-Hua Teng. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. *Journal of the ACM (JACM)*, 51(3):385–463, 2004.

19 David Zuckerman. Linear Degree Extractors and the Inapproximability of Max Clique and Chromatic Number. In *Proceedings of the Symposium on Theory of Computing*, 2006.

A Proof of Theorem 13

Proof.

I. Denote the certified solution returned by the algorithm by $((X, d'), (C_1, \dots, C_k))$. Let $C_1^*, \dots, C_k^*$ be an optimal clustering. Let c_i and c_i^* be optimal centers of C_i and C_i^* (respectively). Consider the case of k-medians first. We upper bound the cost of $(C_1, \dots, C_k)$ w.r.t d as follows:

$$\sum_{i=1}^{k}\sum_{u\in C_i} d(u,c_i) \le \gamma\sum_{i=1}^{k}\sum_{u\in C_i} d'(u,c_i) \le \gamma\sum_{i=1}^{k}\sum_{u\in C_i^*} d'(u,c_i^*) \le \gamma\sum_{i=1}^{k}\sum_{u\in C_i^*} d(u,c_i^*).$$

Now, consider the case of k-means.

$$\sum_{i=1}^{k}\sum_{u\in C_i} d(u,c_i)^2 \le \sum_{i=1}^{k}\sum_{u\in C_i} (\gamma d'(u,c_i))^2 \le \gamma^2\sum_{i=1}^{k}\sum_{u\in C_i^*} d'(u,c_i^*) \le \gamma^2\sum_{i=1}^{k}\sum_{u\in C_i^*} d(u,c_i^*).$$

II. The proof is identical to that of Theorem 10. ◀

Low Diameter Graph Decompositions by Approximate Distance Computation

Ruben Becker
Gran Sasso Science Institute, L'Aquila, Italy
ruben.becker@gssi.it

Yuval Emek
Technion – Israel Institute of Technology, Haifa, Israel
yemek@technion.ac.il

Christoph Lenzen
MPI for Informatics, Saarland Informatics Campus, Saarbrücken, Germany
clenzen@mpi-inf.mpg.de

Abstract

In many models for large-scale computation, decomposition of the problem is key to efficient algorithms. For distance-related graph problems, it is often crucial that such a decomposition results in clusters of small diameter, while the probability that an edge is cut by the decomposition scales linearly with the length of the edge. There is a large body of literature on low diameter graph decomposition with small edge cutting probabilities, with all existing techniques heavily building on *single source shortest paths (SSSP)* computations. Unfortunately, in many theoretical models for large-scale computations, the SSSP task constitutes a complexity bottleneck. Therefore, it is desirable to replace exact SSSP computations with approximate ones. However this imposes a fundamental challenge since the existing constructions of low diameter graph decomposition with small edge cutting probabilities inherently rely on the subtractive form of the triangle inequality, which fails to hold under distance approximation.

The current paper overcomes this obstacle by developing a technique termed *blurry ball growing*. By combining this technique with a clever algorithmic idea of Miller et al. (SPAA 2013), we obtain a construction of low diameter decompositions with small edge cutting probabilities which replaces exact SSSP computations by (a small number of) approximate ones. The utility of our approach is showcased by deriving efficient algorithms that work in the CONGEST, PRAM, and semi-streaming models of computation. As an application, we obtain metric tree embedding algorithms in the vein of Bartal (FOCS 1996) whose computational complexities in these models are optimal up to polylogarithmic factors. Our embeddings have the additional useful property that the tree can be mapped back to the original graph such that each edge is "used" only logaritmically many times, which is of interest for capacitated problems and simulating CONGEST algorithms on the tree into which the graph is embedded.

2012 ACM Subject Classification Mathematics of computing → Graph algorithms; Theory of computation → Parallel algorithms; Theory of computation → Distributed algorithms

Keywords and phrases graph decompositions, metric tree embeddings, distributed graph algorithms, parallel graph algorithms, (semi-)streaming graph algorithms

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.50

Funding *Yuval Emek*: This work has been supported in part by an Israeli Science Foundation grant number 1016/17.

1 Introduction

Consider an n-vertex graph $G = (V, E, \ell)$, where $\ell : E \to \mathbb{Z}_{>0}$ is an edge *length* function.[1] The *distance* between two vertices u and v in G, denoted by $\mathrm{d}_G(u, v)$, is defined to be the length with respect to ℓ of a shortest (u, v)-path in G. The *diameter* of G is the maximum distance between any two vertices, denoted by $\mathrm{diam}(G) = \max_{u,v \in V}\{\mathrm{d}_G(u, v)\}$.

A *decomposition* D of G is a partition of the vertex set V into pairwise disjoint *clusters*. Such a decomposition induces a (multiway) *cut* on G and we use $E^{\mathrm{cut}}(D)$ to denote the subset of edges that cross this cut, namely, edges whose endpoints belong to different clusters of D. The *weight* of the decomposition D is defined to be the sum $\sum_{e \in E^{\mathrm{cut}}(D)} \frac{1}{\ell_e}$ of the reciprocal lengths of the edges crossing its cut. Our focus in this paper is on the construction of decompositions whose clusters' diameter is bounded by some specified parameter r (the notion of a cluster's diameter will be made clear soon), referred to hereafter as *low diameter decompositions*. The challenging part is to keep the weight of D small.

Low diameter decompositions with small weight were first studied by Awerbuch [5] (see also [6, 4]). Bartal [7] introduced their (combinatorially equivalent) probabilistic counterpart: An (r, λ)-*decomposition* of the graph $G = (V, E, \ell)$ is a random decomposition D of G such that (1) the diameter of each cluster in D is at most r; and (2) $\Pr[e \in E^{\mathrm{cut}}(D)] \leq \frac{\lambda \ell_e}{r}$ for every edge $e \in E$. Bartal presented a method that, for a given parameter r, constructs an $(r, O(\log n))$-decomposition and proved the resulting bound on the edge cutting probabilities to be asymptotically tight.

Low diameter decompositions with small edge cutting probabilities have proven to be very useful in the algorithmic arena (see Section 7) and several different techniques have been developed over the years for constructing them [3, 7, 18, 22, 43, 24]. A common thread of all the existing techniques is that they rely heavily on making calls to a single source shortest paths (SSSP) subroutine. While we know how to solve the SSSP problem efficiently in the sequential (centralized) model of computation, the situation is much more challenging in restricted models of computation such as the CONGEST model of distributed computing, the parallel random access memory (PRAM) model, or the semi-streaming graph algorithms model. As it stands, SSSP computations are the main obstruction to designing efficient constructions of low diameter decompositions with small edge cutting probabilities in the aforementioned computational models (and related ones).

1.1 Our Contribution

In this paper, we introduce a new technique that, given a graph $G = (V, E, \ell)$ and a parameter r, constructs an $(r, O(\log n))$-decomposition of G. The crux of our construction is that it does not rely on any *exact* SSSP computations. Rather, it efficiently reduces the task to a small number of calls to an approximate SSSP subroutine. The technical challenge in this regard stems from the fact that the existing constructions of low diameter decompositions with small edge cutting probabilities crucially rely on the subtractive form of the triangle inequality, stating that $\mathrm{d}_G(u, v) \geq \mathrm{d}_G(u, w) - \mathrm{d}_G(v, w)$ for every three vertices $u, v, w \in V$. Due to the subtraction on the right hand side, the inequality fails if one replaces exact distances with approximate ones. The main technical contribution of this paper lies in overcoming this difficulty.

[1] We sometimes use the shorthand ℓ_e for $\ell(e)$.

The approximate SSSP problem can be solved efficiently in the CONGEST [12], PRAM [14], and semi-streaming [12] models, hence we obtain efficient algorithms for constructing $(r, O(\log n))$-decompositions for the three computation models. These in turn can be invoked recursively to yield efficient CONGEST, PRAM, and semi-streaming constructions of path embeddable trees [16, 15] and hierarchically well-separated trees [7, 8, 9, 22] with low *stretch* – important combinatorial objects in their own right. In fact, our low diameter decompositions (and the resulting tree embeddings) admit an even stronger property.

Tree-Supported Decompositions

The notion of graph diameter naturally extends from the entire graph $G = (V, E, \ell)$ to a vertex subset $U \subseteq V$ by considering the maximum distance between any two vertices in U. This yields the following distinction: the *weak diameter* of U in G considers the distances in the underlying graph G, formally defined as $\max_{u,v\in U}\{\mathrm{d}_G(u, v)\}$; the *strong diameter* of U in G considers the distances in the subgraph $G(U)$ induced by G on U, formally defined as $\mathrm{diam}(G(U))$.[2] In the context of low diameter graph decompositions with small edge cutting probabilities, both the weak and strong notions of the cluster diameter have been considered in the literature. As we now explain, the current paper adopts a diameter notion that falls somewhere in between the two.

For a decomposition D of the graph $G = (V, E, \ell)$, we require that each cluster $C \in D$ is associated with a tree $T_C = (U_C, F_C)$, referred to as the *supporting tree* of C, that is a subgraph of G and spans C, i.e., $C \subseteq U_C \subseteq V$ and $F_C \subseteq E$. To emphasize this requirement, we refer to the decomposition D as a *tree-supported decomposition (TSD)*. The *diameter* of a TSD D of G is then defined to be the maximum diameter of any of its supporting trees, denoted by $\mathrm{diam}(D) = \max_{C\in D}\{\mathrm{diam}(T_C)\}$.

Notice that if the supporting tree T_C of each cluster $C \in D$ is required to be a spanning tree of $G(C)$, then $\mathrm{diam}(D)$ bounds the strong diameter of D's clusters. This requirement is not imposed in the current paper, allowing T_C to use edges (and vertices) outside of $G(C)$, meaning that $\mathrm{diam}(D)$ merely bounds the weak diameter of the clusters. However, we do require that the maximum edge *load* is kept small, where the load of edge $e \in E$ in D is defined to be the number of clusters $C \in D$ such that e is included in the supporting tree of C, denoted by $\mathrm{load}_D(e) = |\{C \in D : e \in F_C\}|$. The properties of our graph decomposition construction can now be formally stated.

▶ **Theorem 1.** *There exists a (randomized) algorithm that given a graph $G = (V, E, \ell)$ and a real parameter $r > 0$, constructs a random TSD D of G with the following guarantees: (1)* $\mathrm{diam}(D) \leq r$ *w.h.p.;*[3] *(2)* $\max_{e\in E}\{\mathrm{load}_D(e)\} \leq O(\log n)$ *w.h.p.; and (3)* $\Pr\left[e \in E^{\mathrm{cut}}(D)\right] \leq O\left(\frac{\ell_e \cdot \log n}{r}\right)$ *for every edge $e \in E$. The algorithm is based on an approximate SSSP subroutine without any exact SSSP computations.*

The algorithm promised in Theorem 1 is designed by combining a novel technique termed *blurry ball growing* with the algorithmic ideas of Miller et al. [43]. As discussed earlier, this combination allows us to bypass the need for exact SSSP computations, implementing our algorithm based solely on approximate SSSP. By example of the CONGEST, PRAM, and semi-streaming models, we show that this leads to efficient implementations. We stress that

[2] Unless stated otherwise, the edge length function of a subgraph H of G is the restriction of ℓ to H's edge set.

[3] We say that event A occurs *with high probability*, abbreviated *w.h.p.*, if $\Pr[A] \geq 1 - n^{-c}$, where c is an arbitrarily large constant chosen upfront.

what little computation is performed beyond approximate SSSP computations is very easy, if not trivial, to implement. Accordingly, we expect the technique to carry over to further computational models.

We emphasize that our decomposition maintains a small load of $O(\log n)$ on the edges. Consequently, in many situations, our decomposition can be used in an identical way as a strong diameter decomposition with only polylogarithmic overheads. For example, although we cannot construct low average stretch spanning trees as these are required to be subgraphs of the original graph, we can construct *projected trees* (see Section 5.2), a special case of path-embeddable trees [15, 16]. Projected trees have a mapping of their edges to the original graph such that, e.g., a CONGEST algorithm on the projected tree can be simulated on the original graph with a round complexity overhead proportional to the maximum edge load. Our result is related to the low-congestion shortcuts of Ghaffari and Haeupler [28] with the following differences. In Ghaffari and Haeupler's work, the partition is chosen by an adversary and the input is restricted to unweighted graphs. In contrast, our technique constructs the partition, but weighted graphs can be treated as well. A further possible application of our projected trees is in the field of solvers for symmetric diagonally dominant linear systems, utilizing them in a similar way as low average stretch spanning trees (cf. [16, 15]). Prior algorithms for metric tree embeddings lack this property and, accordingly, cannot take this role.

1.2 Structure of this Paper

We first fix some notation and state basic facts in the preliminaries in Section 2. In Section 3, we present the blurry ball growing technique that we use in Section 4 in order to obtain the routine for computing a random TSD of low diameter, load, and edge cutting probability, as promised in Theorem 1. In Section 5, we highlight some applications of this routine: We first explain how to obtain a hierarchical decompositions by applying the method recursively (Section 5.1) and then show how to obtain random projected trees (Section 5.2) and hierarchically well-separated trees (Section 5.3) with $O(\log^2 n)$ bound on the expected stretch. We also show that this bound can be improved to $O(\log n)$ by considering the relaxed notion of p-stretch [15, 16] (Section 5.4). In Section 6, we explain how to implement our algorithms in the CONGEST, PRAM, and semi-streaming models. Further related work is reviewed in Section 7.

2 Preliminaries

We start with basic notation. We consider a weighted, undirected, connected n-vertex graph $G = (V, E, \ell)$, where $\ell : E \to \mathbb{Z}_{\geq 0}$ is an edge length function. Notice that while some of our subroutines introduce edges $e \in E$ with zero length $\ell_e = 0$, it is assumed that all edges in the original graph input to the $(r, O(\log n))$-decomposition algorithm (as well as the algorithms built on top of it) admit positive lengths. We denote the set of positive length edges by $E_{>0} = \{e \in E \mid \ell_e > 0\}$ and let $\ell_{\min} = \min\{\ell_e \mid e \in E_{>0}\}$ and $\ell_{\max} = \max\{\ell_e \mid e \in E_{>0}\}$. The ratio of $\ell_{\max}$ to $\ell_{\min}$, denoted by $\sigma = \frac{\ell_{\max}}{\ell_{\min}}$ is referred to as the *aspect ratio* of G.

For a subgraph H of G, we denote by $\mathrm{d}_H(u, v)$ the length of the shortest path between two nodes u and v in H. If $H = G$, we may omit the subscript. For a set $B \subseteq V$ and a node $v \in V$, we use $\mathrm{d}(B, v) := \min_{u \in B}\{\mathrm{d}(u, v)\}$ to denote the distance of the node v to the set B. For a set of vertices $U \subseteq V$, we denote by $E^{\mathrm{cut}}(U) := \{e = \{u, v\} \in E \colon u \in U, v \in V \setminus U\}$ the set of edges that are "cut" by U.

Approximate Single Source Shortest Paths

The main subroutine we use in our algorithm computes $(1+\varepsilon)$-*approximate* SSSP in undirected graphs. A $(1+\varepsilon)$-*approximate SSSP algorithm* is an algorithm that takes as input a weighted undirected graph $G = (V, E, \ell)$ and a source node $s \in V$ and returns a spanning tree T of G such that, for every node $v \in V$, the length of the path from s to v in T is at most $(1+\varepsilon) \cdot \mathrm{d}(s, v)$, i.e., $\mathrm{d}(s, v) \leq \mathrm{d}_T(s, v) \leq (1+\varepsilon) \cdot \mathrm{d}(s, v)$.

Super-Source Graphs

Our approach requires $(1+\varepsilon)$-approximate SSSP computations in graphs G_s that result from subgraphs of G by adding a (virtual) *super-source node* $s \notin V$:

▶ **Definition 2** (Super-source graphs). *Fix a subgraph $H = (V_H, E_H, \ell|_H)$ of G. Construct $G_s = (V_H \dot\cup \{s\}, E_H \cup E_s, \ell^{G_s})$ by choosing $E_s \subseteq V_H \times \{s\}$, picking $\ell_e^{G_s} \in \{1, \ldots, n^c\}$ for $e \in E_s$, and setting $\ell_e^{G_s} = \ell_e$ for all $e \in E_H$. We refer to G_s as a* super-source graph *(of G) and to s as its* super-source.

We note that one way of obtaining a super-source graph of a graph G is to contract a subset of nodes, say B, into a super-source s. In this case $V_H = V \setminus B$ and the edges E_s and their lengths result from the contraction of B into s.

Exponential Distribution

We denote the exponential distribution with mean $\frac{1}{\beta}$ by Exp_β. Using the Heaviside step function that is defined as $H(x) = 0$ if $x < 0$ and $H(x) = 1$ otherwise, the density function of the exponential distribution is given by $f_{\mathrm{Exp}_\beta}(x) = \beta \exp(-\beta x) \cdot H(x)$. Its cumulative density function is $F_{\mathrm{Exp}_\beta}(x) = (1 - \exp(-\beta x)) \cdot H(x)$. A standard result is that drawing from this distribution results in values of $O(\beta \log n)$ w.h.p.:

▶ **Lemma 3.** *For parameters $0 < \varepsilon < 1$, $\beta > 0$, and a sufficiently large constant $c > 0$, let $t := \frac{c \log n}{4(1+\varepsilon)\beta}$ and $X \sim \mathrm{Exp}_\beta$. Then $P[X \geq t] = n^{-\Omega(c)}$, i.e., $X < t$ w.h.p.*

Proof. Using the form of the density function, we get

$$P[X \geq t] = \frac{\int_t^\infty \exp(-\beta x)\, dx}{\int_0^\infty \exp(-\beta x)\, dx} = \frac{\exp(-\beta t) \int_0^\infty \exp(-\beta x)\, dx}{\int_0^\infty \exp(-\beta x)\, dx} = \exp(-\Omega(c \log n)) = n^{-\Omega(c)}. \blacktriangleleft$$

We will make heavy use of the following lemma, see the paper by Miller et al. [43] for the proof. Note that in their paper they state the lemma with an upper bound of $O(\beta c)$ on the probability, although their proof in fact bounds the probability by exactly βc.

▶ **Lemma 4** (Lemma 4.4 in [43]). *Let $d_1 \leq \ldots \leq d_s$ be arbitrary values and $\delta_1, \ldots, \delta_s$ be independent random variables picked from Exp_β. Then the probability that the smallest and the second smallest values of $d_i - \delta_i$ are within c of each other is at most βc.*

Miller et al. [43] used this lemma to analyze the following ball growing technique that proceeds in time steps. Every node u in the graph grows a ball B_u independently and in parallel, but with a delay of δ_u time steps, where $\delta_u \sim \mathrm{Exp}_\beta$. Every ball increases its radius by 1 in each time step and we say that the ball B_v "arrives" at node u, if node v minimizes $\mathrm{d}(u, v) - \delta_v$ over all nodes. In this case u "gets absorbed" by v's ball B_v. The process stops when every node u is absorbed by some ball. Notice that u gets absorbed by its own ball B_u, if and only if no other ball arrives at u during the first δ_u time steps.

Now consider an arbitrary edge e in the graph and imagine it to be split into two equal length edges by a node v_e. If we let $d_1 \leq \ldots \leq d_n$ denote the n values $\mathrm{d}(u, v_e) - \delta_u$ for every $u \in V$, the above lemma shows that the arrival times of the first and second ball at node v_e differ by at least $2\ell_e$ with probability $1 - O(\beta\ell_e) = 1 - O(\frac{\ell_e \log n}{\varepsilon r})$, when choosing $\beta = \Theta(\frac{\log n}{\varepsilon r})$. Hence the lemma allows for bounding the probability of an edge being cut by such ball growing process with exponentially distributed delays.

We remark that the implementations in Section 6 draw from discrete distributions. Rounding continuous distributions to multiples of n^{-c} for sufficiently large $c \in O(1)$ yields w.h.p. the same results, but limits the number of random bits required to draw and store a random value to $O(\log n)$.

3 Blurry Ball Growing

In this section, we describe a subroutine called `blur` that takes as input a graph $G = (V, E, \ell)$ with aspect ratio $\sigma = \frac{\ell_{\max}}{\ell_{\min}} \leq \mathrm{poly}(n)$, a node set $B \subseteq V$, and a real $0 < \rho < \ell_{\max}$, and outputs a superset U of B. It guarantees that nodes in U are not too far from B, yet the probability to cut edges is small. More precisely, we establish the following theorem.

▶ **Theorem 5.** *Let $n \geq 2$. There is a routine $\texttt{blur}(G, \rho, B)$ that outputs a superset U of B such that:*

1. *For every edge $e \in E$, the probability that $e \in E^{\mathrm{cut}}(U)$ is bounded by $O(\frac{\ell_e}{\rho})$.*
2. *For every $v \in U$, it holds that $\mathrm{d}(B, v) \leq \frac{\rho}{1-\alpha}$, where $\alpha = \frac{1}{2 \log n}$.*

The routine `blur`, see Algorithm 1, is based on $(1+\varepsilon)$-approximate SSSP computations and contractions of node sets and thus can be readily parallelized. The basic idea is to grow a ball of uniformly random radius around B, where contraction of B yields the super-source of the SSSP computation. However, as approximating distances may imply that the "noise" due to the relative ε-error may cut a short edge with a comparatively large probability, the procedure is repeated with random radii drawn from uniform distributions with width that decrease by factor $\alpha = \frac{1}{2 \log n}$ in each step. To make this work, the approximation error of the SSSP algorithm must satisfy $\varepsilon \leq \alpha^2$. Accordingly, it would be desirable to chose α large for the sake of small computational costs in the approximate SSSP routine. However, it turns out that, in order to achieve Property 1 in Theorem 5, we have to set α such that $\alpha = O\big(\frac{\log \log n}{\log n}\big)$. In addition, the approximate SSSP computations must respect the 0-length edges in the sense that none of the balls we grow cuts these edges. This is ensured by updating the approximate distances so that all nodes in the same connected component of the graph induced by the 0-length edges hold the same value.

Analysis

We begin with two important properties of the distance approximations $\widetilde{\mathrm{d}}^i(\cdot)$ computed in line 9.

▶ **Lemma 6.** *For every iteration i and vertex v in $G^{[i]}$, the distance approximation $\widetilde{\mathrm{d}}^i(v)$ satisfies (I) $\mathrm{d}_{G^{[i]}}(s^{[i]}, v) \leq \widetilde{\mathrm{d}}^i(v) \leq (1+\alpha^2)\,\mathrm{d}_{G^{[i]}}(s^{[i]}, v)$; and (II) if $\{u, v\}$ is a 0-length edge in $G^{[i]}$, then $\widetilde{\mathrm{d}}^i(u) = \widetilde{\mathrm{d}}^i(v)$.*

Proof. Property (II) follows immediately from the definition of $\widetilde{\mathrm{d}}^i(v)$ (see line 9). To see that property (I) holds, notice that $\widetilde{\mathrm{d}}^i(v)$ captures the distance from $s^{[i]}$ to v in the subgraph of $G^{[i]}$ induced by the union of the edge set of $T^{[i]}$ and the 0-length edges. The assertion follows since $T^{[i]}$ is a $(1+\alpha^2)$-approximate SSSP tree of $G^{[i]}$. ◀

Algorithm 1 `blur`(G, ρ, B).

Input : graph $G = (V, E, \ell)$, real $0 < \rho < \ell_{\max}$, set $B \subset V$
Output : set $U \subseteq V$

1 $i := 0$, $B^{[0]} := B$, $\alpha := \frac{1}{2 \log n}$
2 **while** $\alpha^i \rho \geq \ell_{\min}$ **do**
3 $i := i + 1$, $r^{[i]} \in \mathcal{U}[0, \alpha^{i-1}\rho]$.
4 Obtain super-source graph $G^{[i]}$ from G by contracting $B^{[i-1]}$ into a super-source node $s^{[i]}$
5 Compute $(1 + \alpha^2)$-approximate SSSP tree $T^{[i]}$ of $G^{[i]}$
6 Let $G_0^{[i]}$ be the restriction of the graph $G^{[i]}$ to its 0-length edges
7 **for** *each* $v \in G^{[i]}$ **do**
8 Let $W(v)$ be the connected component of v in $G_0^{[i]}$
9 $\widetilde{\mathrm{d}}^{[i]}(v) := \min\{\mathrm{d}_{T^{[i]}}(s^{[i]}, u) \mid u \in W(v)\}$
10 $B^{[i]} := B^{[i-1]} \cup \{v \in G^{[i]} \mid \widetilde{\mathrm{d}}^{[i]}(v) \leq r^{[i]}\} \setminus \{s^{[i]}\}$
11 **return** $\bigcup_{j=0}^{i} B^{[i]}$

Next, we establish Property 2 of Theorem 5, which readily follows from the manner in which we sample $r^{[i]}$ from $\mathcal{U}[0, \alpha^{i-1}\rho]$.

▶ **Lemma 7.** *If* $\mathrm{d}_{G^{[i+1]}}(s^{[i+1]}, u) = \mathrm{d}(B^{[i]}, u) \geq \frac{\alpha^i \rho}{1-\alpha}$ *for some* i*, then* $u \notin U$*. In particular, it holds that* $\mathrm{d}_G(B, v) \leq \frac{\rho}{1-\alpha}$ *for every* $v \in U$*.*

Proof. Any $u \in B^{[k]}$ for $k > i$ has distance to $B^{[i]}$ at most $\sum_{j \geq i+1} r^{[j]} \leq \sum_{j \geq i+1} \alpha^{j-1}\rho < \alpha^i \rho \sum_{j=0}^{\infty} \alpha^j = \frac{\alpha^i \rho}{1-\alpha}$, showing the first claim. Setting $i = 0$ yields the second claim. ◀

It remains to verify Property 1 of Theorem 5, i.e., that the probability of cutting edge $e = \{u, v\} \in E$ is bounded from above by $O(\frac{\ell_e}{\rho})$. Property (II) of Lemma 6 ensures that this bound holds if $\ell_e = 0$ as in this case, in each iteration i, either both u and v join $B^{[i]}$ in line 10 or none of them does. In the remainder of this section, we therefore focus on the edges in $E_{>0} = \{e \in E \mid \ell_e > 0\}$, starting with the following definition.

▶ **Definition 8.** *We say that edge* $\{u, v\} \in E_{>0}$ *is* safe *after step* i *of* `blur`(G, ρ, B) *if either* $u, v \in B^{[i]}$ *or* $\min\{\mathrm{d}_{G^{[i+1]}}(s^{[i+1]}, u), \mathrm{d}_{G^{[i+1]}}(s^{[i+1]}, v)\} \geq \frac{\alpha^i \rho}{1-\alpha}$.

Clearly, if $\{u, v\} \in E$ is safe after step i of `blur`(G, ρ, B), then $e \notin E^{\mathrm{cut}}(U)$: if $u, v \in B^{[i]}$, then $u, v \in U$ by construction; if $\min\{\mathrm{d}_{G^{[i+1]}}(s^{[i+1]}, u), \mathrm{d}_{G^{[i+1]}}(s^{[i+1]}, v)\} \geq \frac{\alpha^i \rho}{1-\alpha}$, then $u, v \notin U$ by Lemma 7. See Figure 1 for an illustration of these two cases. Thus, in order to bound the probability of an edge being cut, it suffices to bound the probability that an edge never becomes safe. Accordingly, we define $X_{i,e}$ to be the event that e is not safe after step i of the algorithm conditioned on the event that e was not safe after step $i - 1$ and bound $P[X_{i,e}]$.

▶ **Lemma 9.** *For each iteration* i *and* $e \in E_{>0}$*, it holds that* $\Pr[X_{i,e}] \leq \frac{5}{4} \cdot \frac{\ell_e}{\alpha^{i-1}\rho} + \alpha \cdot (1 + 4\alpha)$.

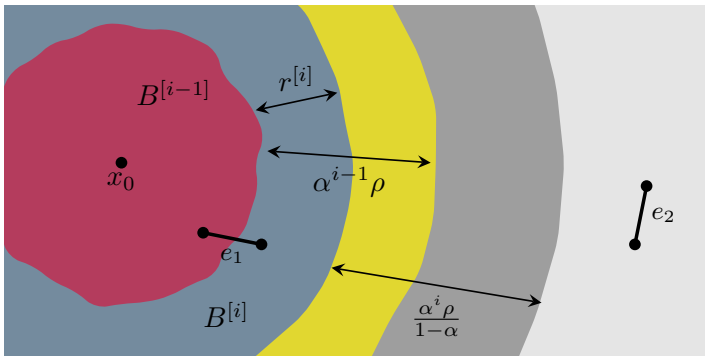


Figure 1 An illustration of the blurry ball growing procedure $\texttt{blur}(G, \rho, B)$ in iteration i. The radius $r^{[i]}$ is sampled uniformly from $[0, \alpha^{i-1}\rho]$ and $B^{[i]}$ is defined as all nodes whose $(1+\alpha^2)$-approximate distance to $B^{[i-1]}$ is at most $r^{[i]}$. Both edges e_1 and e_2 are safe from being cut after iteration i: e_1 has both endpoints in $B^{[i]} \subseteq U$ and both endpoints of e_2 are farther away from $B^{[i]}$ than $\frac{\alpha^i\rho}{1-\alpha}$, meaning that neither of them will lie in U after termination.

Proof. By Definition 8, it holds that if $e = \{u, v\} \in E_{>0}$ is not safe after step i, we must, w.l.o.g. over the choice of u, v, have $\mathrm{d}_{G^{[i+1]}}(s^{[i+1]}, u) < \frac{\alpha^i\rho}{1-\alpha}$ and $\{u, v\} \not\subseteq B^{[i]}$. By the approximation guarantee of the SSSP algorithm and the triangle inequality, we get

$$\begin{aligned} r^{[i]} < \max\{\widetilde{\mathrm{d}}^{[i]}(u), \widetilde{\mathrm{d}}^{[i]}(v)\} &\leq (1+\alpha^2)\max\{\mathrm{d}_{G^{[i]}}(s^{[i]}, u), \mathrm{d}_{G^{[i]}}(s^{[i]}, v)\} \\ &\leq (1+\alpha^2)(\mathrm{d}_{G^{[i]}}(s^{[i]}, u) + \ell_e)\,, \end{aligned}$$

where the first transition follows from property (I) in Lemma 6. From the former inequality, we get that

$$\mathrm{d}_{G^{[i]}}(s^{[i]}, u) \leq \mathrm{d}_{G^{[i+1]}}(s^{[i+1]}, u) + r^{[i]} < \frac{\alpha^i\rho}{1-\alpha} + r^{[i]}, \tag{1}$$

which yields $r^{[i]} \geq \mathrm{d}_{G^{[i]}}(s^{[i]}, u) - \frac{\alpha^i\rho}{1-\alpha}$. As $r^{[i]}$ is drawn uniformly from an interval of length $\alpha^{i-1}\rho$, these lower and upper bounds on $r^{[i]}$ readily imply a bound on the probability of $X_{i,e}$:

$$\begin{aligned} \Pr[X_{i,e}] &\leq \Pr\left[r^{[i]} \in \left(\mathrm{d}_{G^{[i]}}(s^{[i]}, u) - \frac{\alpha^i\rho}{1-\alpha}, (1+\alpha^2)\cdot(\mathrm{d}_{G^{[i]}}(s^{[i]}, u) + \ell_e)\right)\right] \\ &\leq \frac{(1+\alpha^2)\ell_e}{\alpha^{i-1}\rho} + \frac{\alpha^2\,\mathrm{d}_{G^{[i]}}(s^{[i]}, u)}{\alpha^{i-1}\rho} + \frac{\alpha}{1-\alpha}. \end{aligned} \tag{2}$$

Moreover, from (1) and $r^{[i]} \leq \alpha^{i-1}\rho$, we conclude that $\mathrm{d}_{G^{[i]}}(s^{[i]}, u) < \alpha^{i-1}\rho\cdot(1+\frac{\alpha}{1-\alpha}) = \frac{\alpha^{i-1}\rho}{1-\alpha}$. Plugging into (2), with $\alpha \leq \frac{1}{2}$ we get that $\Pr[X_{i,e}] \leq \frac{5}{4}\cdot\frac{\ell_e}{\alpha^{i-1}\rho} + \frac{\alpha^2}{1-\alpha} + \frac{\alpha}{1-\alpha} \leq \frac{5}{4}\cdot\frac{\ell_e}{\alpha^{i-1}\rho} + \alpha(1+4\alpha)$. ◀

Applying this lemma to all iterations in which e has a significant probability to become safe (i.e., all iterations i for which $\alpha^{i-1}\rho \geq \ell_e$), we obtain the desired bound on the probability that e is cut.

▶ **Lemma 10.** *For every edge $e \in E$, it holds that* $\Pr[e \in E^{\mathrm{cut}}(U)] = O\left(\frac{\ell_e}{\rho}\right)$.

Proof. The case where $\ell_e = 0$ has already been treated, so assume hereafter that $e \in E_{>0}$. If $\ell_e > \rho$, then trivially $\Pr[e \in E^{\text{cut}}(U)] \leq 1 < \frac{\ell_e}{\rho}$. Otherwise, we let $i_e \geq 1$ be the largest index such that $\ell_e \leq \alpha^{i_e-1}\rho$. By Lemma 9, for all i, the probability that an edge that is not safe after $i-1$ steps is still not safe after step i is bounded by $\Pr[X_{i,e}] \leq \frac{5}{4} \cdot \frac{\ell_e}{\alpha^{i-1}\rho} + \alpha \cdot (1+4\alpha)$. Depending on the index i, we differentiate this upper bound further:

- Case $i = i_e$: As $\alpha^{i_e}\rho < \ell_e$, we get that $\alpha < \frac{\ell_e}{\alpha^{i_e-1}\rho}$. With $\alpha \leq \frac{1}{2}$, $\Pr[X_{i_e,e}] < \frac{5\ell_e}{\alpha^{i_e-1}\rho}$ follows.
- Case $i = i_e - 1$: As $\ell_e \leq \alpha^{i_e-1}\rho$, we conclude that $\frac{\ell_e}{\alpha^{i-1}\rho} \leq \frac{\alpha^{i_e-1}}{\alpha^{i_e-2}} = \alpha$, yielding with $\alpha \leq \frac{1}{2}$ that $\Pr[X_{i_e-1,e}] < 5\alpha$.
- Case $i \leq i_e - 2$: This entails that $\frac{\ell_e}{\alpha^{i-1}\rho} \leq \alpha^2$ and thus $\Pr[X_{i,e}] < 2\alpha^2 + \alpha \cdot (1+4\alpha) = \alpha \cdot (1+6\alpha)$.

Using these bounds and distinguishing cases based on i_e, we can bound the overall probability that the edge is cut.

- Case $i_e = 1$: $\Pr[e \in E^{\text{cut}}(U)] \leq \Pr[X_{1,e}] = \Pr[X_{i_e,e}] < \frac{5\ell_e}{\rho}$.
- Case $i_e = 2$: $\Pr[e \in E^{\text{cut}}(U)] \leq \Pr[X_{2,e}] \cdot \Pr[X_{1,e}] = \Pr[X_{i_e,e}] \cdot \Pr[X_{i_e-1,e}] < \frac{5\ell_e}{\alpha\rho} \cdot 5\alpha = \frac{25\ell_e}{\rho}$.
- Case $i_e \geq 3$: $\Pr[e \in E^{\text{cut}}(U)] \leq \frac{25\ell_e}{\alpha^{i_e-2}\rho} \cdot \prod_{i \leq i_e-2} \Pr[X_{i,e}] \leq \frac{25\ell_e}{\alpha^{i_e-2}\rho} \cdot (\alpha(1+6\alpha))^{i_e-2} < \frac{25\ell_e}{\rho} \cdot (1+6\alpha)^{i_e}$.

Hence, it remains to bound $(1+6\alpha)^{i_e} = O(1)$. Since $\rho < \ell_{\max}$, it follows that

$$\ell_{\min} \leq \ell_e \leq \alpha^{ie-1}\rho < \alpha^{ie-1}\ell_{\max}\,.$$

Recalling that $\sigma = \frac{\ell_{\min}}{\ell_{\max}} \leq n^{O(1)}$, we conclude that $i_e = O\left(\frac{\log n}{\log(1/\alpha)}\right)$. Therefore,

$$(1+6\alpha)^{i_e} = (1+6\alpha)^{O(\log n/\log(1/\alpha))} = \left((1+6\alpha)^{1/(6\alpha)}\right)^{O(\alpha \log n/\log(1/\alpha))} = e^{O(\alpha \log n/\log(1/\alpha))}\,.$$

The assertion follows by the choice of $\alpha = \frac{1}{2\log n}$. ◀

Theorem 5 now follows from Lemmas 7 and 10.

4 Tree-Supported Decomposition

In this section, we present the construction of TSDs that admit low diameter, low load, and low edge cutting probability, establishing Theorem 1. Our method is inspired by the partition technique from [43] that allows for efficient parallel and distributed implementations. However, we seek to rely on approximate rather than on exact distance computations.

To motivate our approach, consider a naive application of the decomposition technique from [43] using approximate rather than exact distance computations. This would look as follows: One would add a super-source s to the graph, assign exponentially sampled lengths to the edges adjacent to s, compute a $(1+\varepsilon)$-approximate SSSP tree T rooted at s for some small enough ε, and partition the node set V according to the subtrees of T rooted at the children of s. This approach certainly leads to a decomposition of G. However, a consequence of the approximate distance computation is that the probability to cut a short edge is dominated by the approximation error, which is ε times the distance to the source – an expression that may be very large compared to the length of the edge.

In order to still ensure the desired bound, we seek to employ the blurring technique from the previous section to clusters obtained as described above. This introduces the new obstacle that the clusters need to be separated from each other first, as the blurring procedure grows the clusters by a random radius. We enforce this separation by removing from each cluster every node that is too close to its boundary; Property 2 of Theorem 5, stating that

the distance of any node in the blurred cluster from the original cluster is at at most $\frac{\rho}{1-\alpha}$, determines what precisely is "too close." While this may result in a large portion of the graph not being contained in any cluster even after blurring all clusters, we can ensure that each edge is contained in some cluster with probability at least $p = \Omega(1)$ (or is very long and can be safely deleted). Hence, repeating the procedure $O(\log n)$ times completes the decomposition w.h.p.

The blurring procedure presented in Section 3 requires that the aspect ratio $\sigma = \frac{\ell_{\max}}{\ell_{\min}}$ of its input graph is bounded by $\text{poly}(n)$ and that the parameter ρ is smaller than $\ell_{\max}$. Therefore, we have to slightly modify the graph so that it satisfies these two conditions before invoking the blurring procedure. The latter requirement is readily satisfied by deleting all edges that are sufficiently long for the edge cutting probability bound to be greater than 1 (clearly, these edges can be safely deleted). For the former requirement, we reset the length of all edges $e \in E$ that are significantly shorter than the cluster diameter bound, thus ensuring that the aspect ratio of the graph is $n^{O(1)}$. The original length of all 0-length edges is then recovered after the decomposition is constructed. Since any simple path in the graph contains at most $n-1$ short edges, it follows that the length recovery operation does not increase the diameter of any cluster in the decomposition by "too much".

Algorithm 2 `ts_decompose` (G, Δ).

Input : graph $G = (V, E, \ell)$ and $\Delta \in \mathbb{N}$
Output : decomposition $D = (C_1, \dots, C_k)$ of G, trees $\mathcal{T} = (T_1, \dots, T_k)$ of depth $\leq \frac{\Delta}{2}$ s.t. T_i spans a superset of C_i

$\beta := \frac{3c \log n}{\Delta}$, $\varepsilon := \frac{1}{c \log^2 n}$, $D := \emptyset$, $\mathcal{T} := \emptyset$ *// c sufficiently large constant*
delete all edges $e \in E$ of length $\ell_e > \frac{1}{40\beta}$ *// long edges*
reset the length of each edge $e \in E$ with $\ell_e < \frac{\Delta}{6n}$ by setting $\ell_e = 0$ *// short edges*
while $E(G) \neq \emptyset$ **do**
  *// * initial decomposition by exponential shifts **
  pick $\delta_u \sim \text{Exp}_\beta$ for each $u \in V$ independently
  $G_s :=$ super-source graph of G with edges $\{u, s\}$ of length $\ell_{us} = 1 + \max_{v \in V}\{\delta_v\} - \delta_u$ for $u \in V$
  $T :=$ $(1+\varepsilon)$-approximate SSSP tree for G_s with source s
  $R :=$ roots of $T \setminus \{s\}$ and $\mathcal{V} := (V_u)_{u \in R}$, where V_u are the nodes in u's subtree
  *// * separate cells **
  $\partial\mathcal{V} := \bigcup_{u \in R}\{v \in V_u \mid \exists\{v, w\} \in E \colon w \notin V_u\}$
  $G'_s :=$ super-source graph of G with edges $\{u, s\}$ of length 1 for $u \in \partial\mathcal{V}$
  $T' :=$ $(1+\varepsilon)$-approximate SSSP tree for G'_s with source s
  for *each* $u \in R$ **do**
    $V_u^\circ := V_u \setminus \{v \in V_u \mid \text{d}_{T'}(s, v) \leq \frac{1+\varepsilon}{4\beta}\}$
    // V_u° is the interior of cell V_u
    $C_u := \texttt{blur}(G, \rho, V_u^\circ)$, where $\rho := \frac{1 - \frac{1}{2 \log n}}{4\beta}$
    append C_u to D and the subtree of T rooted at u to $\mathcal{T}$
    $G := G \setminus C_u$
recover the original length of each edge $e \in E$ with $\ell_e = 0$
return $(D, \mathcal{T})$

▶ Remark. The operations of resetting the lengths of the short edges (line 3) and recovering their original lengths (line 17) are necessary only if the aspect ratio of G is large and can be ignored if the aspect ratio is guaranteed to be bounded by $n^{O(1)}$. In this case, the graphs handed to the blurring procedure (line 14) have no zero length edges.

Algorithm

The pseudocode of our procedure `ts_decompose` is given in Algorithm 2. The value β chosen in Line 1 is the parameter chosen for the exponential distributions: up to normalization, the density of the distribution is $\exp(-\beta x)$. The diameter of each (initial) cluster is bounded by $\max_{v\in V}\{\delta_v\}$, which we need to be smaller than $\frac{\Delta}{2}$ w.h.p. However, the probability to cut edges increases as we make the distributions "narrower," i.e., β larger. Accordingly, we choose $\beta = \Theta\left(\frac{\log n}{\Delta}\right)$, just small enough to ensure $\delta_v \leq \frac{\Delta}{2}$ w.h.p. for all $v \in V$.

The partition from [43] can be interpreted as a Voronoi decomposition in which each cell center x_v is a virtual copy of its corresponding node $v \in V$ that is attached to v by an edge of length $\max_{w\in V}\{\delta_w\} - \delta_v$. Note that the children of the virtual node s in the (approximate) shortest path tree T are exactly the nodes which have not been "absorbed" into another node's Voronoi cell before they started to grow their own. Lines 9 to 13 remove from each cluster nodes that are in distance (roughly) $\frac{1}{4\beta}$ from the boundary of the Voronoi cell containing them. Choosing a distance of $O\left(\frac{1}{\beta}\right)$ here ensures a constant probability that edges of this length remain in a shrunk cluster; longer edges can safely be cut, as the required bound on the probability for cutting them is trivial (i.e., 1), which is why they are removed at the start of the routine. We then proceed to applying the blurring subroutine to each (remaining) shrunk cluster. Note that, as the clusters remain separated due to the choice of parameters, we can realize this step concurrently for all clusters. The algorithm iterates until all nodes are assigned to clusters, which requires $O(\log n)$ loop iterations w.h.p.

The remainder of this section is dedicated to proving Theorem 1.

Number of Iterations

We first prove the key statement that, with at least constant probability, for any node w, a ball of radius $\Theta(\frac{1}{\beta})$ around it is contained within the interior of a cell.

▶ **Lemma 11.** *Consider an iteration of the while loop of Algorithm 2 and (by slight abuse of notation) denote by $G = (V, E)$ the subgraph that remains at the beginning of the iteration. For any $w \in V$, with at least constant probability a ball of radius $\frac{1}{40\beta}$ around it is contained in the interior of a cell computed in Line 13.*

Proof. For $x \in V$, set $\mathrm{d}_x := \mathrm{d}_{G_s}(x, w) + 1 + \max_{y\in V}\{\delta_y\}$. Moreover, set $X_x := \mathrm{d}_x - \delta_x = \ell_{sx} + d_{G_s}(x, w)$ for $x \in V$ and let $X^{(i)}$ be the i'th order statistic of the variables X_v (i.e., the i'th smallest element). Denote by $x_{\min} \in V$ the node for which $X_{x_{\min}} = X^{(1)}$. By Lemma 4, with constant probability $X^{(2)} - X^{(1)} \geq \frac{7}{8\beta}$. Condition on this event. Accordingly, we have for all $x \in V \setminus \{x_{\min}\}$ that $X_x - X_{x_{\min}} \geq X^{(2)} - X^{(1)} \geq \frac{7}{8\beta}$.

Denote for each $v \in V$ by x_v the child of s in T in whose subtree v is situated. Then the assumption that $x_v \neq x_{\min}$ implies by copious use of the triangle inequality that

$$\begin{aligned}
\mathrm{d}_T(s,v) - \mathrm{d}_{G_s}(s,v) &= \ell_{x_v s} + \mathrm{d}_T(x_v,v) - \mathrm{d}_{G_s}(s,v) \\
&\geq \ell_{x_v s} + \mathrm{d}_{G_s}(x_v,v) - \mathrm{d}_{G_s}(s,v) \\
&\geq \ell_{x_v s} + \mathrm{d}_{G_s}(x_v,w) - \mathrm{d}_{G_s}(v,w) - (\mathrm{d}_{G_s}(s,w) + \mathrm{d}_{G_s}(v,w)) \\
&\geq \ell_{x_v s} + \mathrm{d}_{G_s}(x_v,w) - (\ell_{x_{\min} s} + \mathrm{d}_{G_s}(x_{\min},w)) - 2\,\mathrm{d}_{G_s}(v,w) \\
&= X_{x_v} - X_{x_{\min}} - 2\,\mathrm{d}_{G_s}(v,w) \geq \frac{7}{8\beta} - 2\,\mathrm{d}_{G_s}(v,w).
\end{aligned}$$

On the other hand, the approximation guarantee of the SSSP algorithm yields that

$$\mathrm{d}_T(s,v) - \mathrm{d}_{G_s}(s,v) \leq \varepsilon\,\mathrm{d}_{G_s}(s,v) \leq \varepsilon \ell_{vs} \leq \varepsilon \max_{x \in V}\{1+\delta_x\}.$$

By Lemma 3, w.h.p. $\max_{x\in V}\{\delta_x\} \leq t = \frac{c\log n}{4(1+\varepsilon)\beta}$ after sampling the δ-values in Line 5 of this iteration. Condition on this event as well. Using that $\varepsilon = \frac{1}{c\log^2 n}$ and c is sufficiently large, we get that $\mathrm{d}_T(s,v) - \mathrm{d}_{G_s}(s,v) \leq \varepsilon(1+t) < \frac{1}{4\beta}$.

In summary, if both events on which we conditioned occur, $x_v \neq x_{\min}$ entails that

$$\mathrm{d}_{G_s}(v,w) > \frac{5}{16\beta}. \tag{3}$$

In particular, choosing $v = w$ yields the contradiction $0 = \mathrm{d}_{G_s}(w,w) > \frac{5}{16\beta}$, i.e., $x_w = x_{\min}$.

We proceed to show that $\mathrm{d}_G(v,w) \leq \frac{1}{40\beta}$ implies that also $v \in V^\circ_{x_{\min}}$. By a union bound over the two events on which we conditioned, this will complete the proof. To this end, observe that Inequality (3) shows that a ball of radius $\frac{5}{16\beta}$ around w in G_s is contained within $V_{x_{\min}}$. Because longer edges have been deleted, nodes in $\partial\mathcal{V}$ are connected to neighbors outside their cell by edges of length at most $\frac{1}{40\beta}$. Together with the approximation guarantee of the second SSSP computation used to compute T', it follows that nodes $v \in V$ for which $\mathrm{d}_{G_s}(v,w) \leq \frac{1/16 - 1/40 - \varepsilon}{\beta} < \frac{5}{16\beta} - \frac{(1+\varepsilon)^2}{4\beta} - \frac{1}{40\beta}$ end up in $V^\circ_{x_{\min}}$. In particular, as trivially $\mathrm{d}_{G_s}(v,w) \leq \mathrm{d}_G(v,w)$ and ε is sufficiently small, we conclude that $\mathrm{d}_G(v,w) \leq \frac{1}{40\beta}$ implies that $v \in V^\circ_{x_{\min}}$. ◀

▶ **Corollary 12.** *Algorithm 2 terminates after $O(\log n)$ iterations of the while loop w.h.p.*

Proof. Consider any edge $e \in E$ that is not deleted right away, i.e., $\ell_e \leq \frac{1}{40\beta}$. By Lemma 11, in each iteration in which e is present in the remaining subgraph of G, there is a constant probability that it is contained in V°_u for some node u. Thus, the probability that the edge remains for $c\log n$ iterations is bounded by $2^{-\Omega(c\log n)} = n^{-\Omega(c)}$. By a union bound, this implies that all edges are either cut or included in a part within $O(\log n)$ iterations w.h.p., i.e., the termination condition that $E(G)$ is empty becomes satisfied. ◀

The Diameter Bound

In order to prove that the diameter bound holds, we first show that for each iteration of the while loop of Algorithm 2 and each $u \in R$, we have that $C_u \subseteq V_u$.

▶ **Lemma 13.** *Fix any iteration of the while loop of Algorithm 2 and $u \in R$. It holds that $C_u \subseteq V_u$.*

Proof. Again, denote for simplicity the remaining subgraph at the beginning of the loop iteration by $G = (V,E)$. By the approximation guarantee of the second call to the SSSP algorithm, $v \in V^\circ_u$ implies that $\mathrm{d}(v,\partial\mathcal{V}) \geq \frac{1}{4\beta}$. By Theorem 5, $w \in C_u$ implies that

$\mathrm{d}_G(w, V_u^\circ) \leq \frac{\rho}{1-1/(2\log n)} = \frac{1}{4\beta}$. Consider the node $v \in V_u^\circ$ that is closest to w and fix a shortest path from v to w. By the second bound, the path is no longer than $\frac{1}{4\beta}$, which by the first bound implies that it cannot leave V_u. Hence, $w \in V_u$, showing the claim of the lemma. ◀

We observe that the above lemma yields that the algorithm indeed outputs a partition of V, and each set in the partition is spanned by the corresponding tree in $\mathcal{T}$. We now apply the tail bound on Exp_β given in Lemma 3 to infer that the diameter of the computed parts is appropriately bounded w.h.p.

▶ **Lemma 14.** *W.h.p., each cluster in the decomposition D returned by* `ts_decompose`(G, Δ) *has weak diameter at most $\frac{\Delta}{2}$. This is witnessed by the corresponding tree in $\mathcal{T}$.*

Proof. We prove that each tree $T \in \mathcal{T}$ has diameter at most $\frac{\Delta}{3}$ assuming that the lengths of the short edges e are reset to $\ell_e = 0$ (see line 3 of `ts_decompose`(G, Δ)). This implies the desired $\frac{\Delta}{2}$-bound on the diameter of T in the original graph since each path P in T includes less than n short edges and each such short edge adds at most a $\frac{\Delta}{6n}$-term to the total length of P when recovering its original length (line 17 of `ts_decompose`(G, Δ)).

By Lemma 3 and a union bound over all nodes, w.h.p. always $\max_{v\in V}\{\delta_v\} \leq 1 + t$ for $t = \frac{c\log n}{4(1+\varepsilon)\beta}$ in Line 5 of `ts_decompose`(G, Δ). Assume that v ends up in the subtree of T rooted at the child x_v of s in G_s. From the above bound, it follows that, w.h.p.,

$$\begin{aligned}\mathrm{d}_T(x_v, v) &= \mathrm{d}_T(s, v) - \ell_{x_v s} \leq (1+\varepsilon)\cdot \mathrm{d}_{G_s}(s, v) - \ell_{x_v s} \leq (1+\varepsilon)\cdot \ell_{vs} - \ell_{x_v s}\\ &= \varepsilon\cdot(1 + \max_{x\in V}\{\delta_x\} - \delta_v) + \delta_{x_v} - \delta_v \leq \varepsilon\cdot(1+t) + t = (1+\varepsilon)\cdot t + \varepsilon.\end{aligned}$$

Recalling that R denotes the children of the root node in T, it follows that for each $x \in R$, we have that T_x has (weighted) depth at most $(1+\varepsilon)t+\varepsilon$ w.h.p. in Line 8 of `ts_decompose`(G, Δ). We conclude that w.h.p., for all $u \in R$, it holds that the subgraph induced by V_u has diameter at most $2[(1+\varepsilon)t + \varepsilon] \leq \frac{c\log n}{2\beta} + 2\varepsilon \leq \frac{\Delta}{3}$, using that $\varepsilon = \frac{1}{c\log^2 n} \leq \frac{1}{12}$ for sufficiently large c. Using Lemma 13 concludes the proof. ◀

The Edge Cutting Probability Bound

We proceed to showing that the probability to cut an edge is sufficiently small. This follows from the analysis of Algorithm 1 and the probabilistic progress guarantee from Lemma 11.

▶ **Corollary 15.** *The probability that edge $e \in E$ is cut by* `ts_decompose`(G, Δ) is $O\left(\frac{\ell_e \log n}{\Delta}\right)$.

Proof. Consider edge $e = \{v, w\} \in E$. If e is deleted right away, then $\ell_e > \frac{1}{40\beta} = \Omega\left(\frac{\Delta}{\log n}\right)$ and the claim trivially holds. Accordingly, assume that $\ell_e \leq \frac{1}{40\beta}$ in the following.

As shown in Lemma 13, in each iteration the parts $(V_u^\circ)_{u\in C}$ satisfy that $V_u^\circ \subseteq V_u$. Thus, if $v \in V_x$ and $w \in V_y$ for some $x, y \in C$ after Line 8, e can be only cut by v ending up in C_x, while w does not, or w ending up in C_y, while v does not. Lemma 10 shows that the probability for either event is bounded by $O\left(\frac{\ell_e \log n}{\Delta}\right)$, independently of the subgraph the calls to Algorithm 1 are executed on.

Combining this observation with the fact that, in each iteration in which e is still present by Lemma 11 it ends up in some part with probability at least $p \in \Omega(1)$, we can bound the probability that e is cut by

$$\sum_{i=1}^{\infty}(1-p)^{i-1} O\left(\frac{\ell_e \log n}{\Delta}\right) = O\left(\frac{\ell_e \log n}{p\Delta}\right) = O\left(\frac{\ell_e \log n}{\Delta}\right).$$ ◀

The Load Bound

As the trees added to the output in a single iteration are subtrees of the same shortest path tree, these trees are disjoint. Hence, the bound on the number of iterations also bounds the number of trees in which an edge may participate and thus the load of that edge in the output decomposition D. This concludes the proof of Theorem 1.

5 Sampling from Low Stretch Tree Embeddings

Consider some graph $G = (V, E, \ell)$ with positive edge lengths. We say that graph $G' = (V', E', \ell')$ with $V' \supseteq V$ *dominates* G if $\mathrm{d}_{G'}(u, v) \geq \mathrm{d}_G(u, v)$ for every two vertices $u, v \in V$. In that case, we define the stretch of edge $e = \{u, v\} \in E$ in G' to be

$$\mathrm{str}_{G'}(e) = \frac{\mathrm{d}_{G'}(u, v)}{\ell_e}.$$

Our goal in this section is to construct random dominating trees of a given graph $G = (V, E, \ell)$ that guarantee low expected stretch for each edge in E. The dominating trees we construct, referred to hereafter as *virtual trees*, are not spanning trees of G, because they may include vertices and edges that do not belong to V and E, respectively. Nevertheless, they admit some useful characteristics. Specifically, we consider two types of virtual (dominating) trees: *projected trees* (a special case of the path embeddable trees of [15, 16]) addressed in Section 5.2 and *hierarchically well separated trees (HSTs)* addressed in Section 5.3. In both cases, the respective constructions are based on recursive applications of the graph decomposition technique presented in Section 4, generating a *hierarchical* version of TSDs as presented in Section 5.1.

5.1 Hierarchical Decompositions

A *hierarchical tree-supported decomposition (HTSD)* $\mathbf{D}$ of a graph G is a sequence $\mathbf{D} = (D_0, D_1, \ldots, D_k)$ of TSDs that satisfies (i) $D_0 = \{V\}$; (ii) $D_k = \{\{v\} \mid v \in V\}$; and (iii) for every $1 \leq i \leq k$ and $C \in D_i$, there exists some $C' \in D_{i-1}$ such that $C \subseteq C'$. The TSDs $D_0, D_1, \ldots, D_k$ are referred to as the *levels* of $\mathbf{D}$ and the parameter k is referred to as its *depth*. The *load* of edge $e \in E$ in $\mathbf{D}$ is defined to be $\mathrm{load}_{\mathbf{D}}(e) = \sum_{i=0}^{k} \mathrm{load}_{D_i}(e)$.

The real sequence $\mathbf{d} = (d_0, d_1, \ldots, d_k)$ is said to be *diameter bounding* for the HTSD $\mathbf{D}$ if $\mathrm{diam}(D_i) \leq d_i$ for every $0 \leq i \leq k$. Of particular interest are HTSDs that admit a *geometrically decreasing* diameter bounding sequence, namely a sequence $\mathbf{d} = (d_0, d_1, \ldots, d_k)$ that satisfies $d_i \leq \alpha \cdot d_{i-1}$, $1 \leq i \leq k$, for some constant $\alpha > 1$.

Consider some HTSD $\mathbf{D} = (D_0, D_1, \ldots, D_k)$ of G with a geometrically decreasing diameter bounding sequence $\mathbf{d} = (d_0, d_1, \ldots, d_k)$. Edge $e = \{u, v\} \in E$ is said to be *decoupled* on level $0 \leq i \leq k-1$ if u and v belong to the same cluster in level i and to different clusters in level $i+1$, that is $e \in E^{\mathrm{cut}}(D_{i+1}) - E^{\mathrm{cut}}(D_i)$. In that case, we define the *stretch* of e in $\mathbf{D}$ with respect to $\mathbf{d}$ to be

$$\mathrm{str}_{\mathbf{D},\mathbf{d}}(e) = \frac{d_i}{\ell_e}.$$

▶ **Theorem 16.** *There exists a (randomized) algorithm that, given a graph $G = (V, E, \ell)$ with positive integral edge lengths and aspect ratio $\sigma = \frac{\ell_{\max}}{\ell_{\min}}$, constructs a random HTSD $\mathbf{D}$ of G with the following guarantees: (1) the depth of $\mathbf{D}$ is $n^{O(1)}$ w.h.p.; (2) $\mathbf{D}$ admits a geometrically decreasing diameter bounding sequence $\mathbf{d} = (d_0, d_1, \ldots, d_k)$ w.h.p.; (3) $\mathrm{load}_{\mathbf{D}}(e) = O(\log \sigma)$ for every edge $e \in E$ w.h.p.; and (4) $\mathbb{E}_{\mathbf{D}}[\mathrm{str}_{\mathbf{D},\mathbf{d}}(e)] = O(\log^2 n)$ for every edge $e \in E$.*

Proof. We first present the construction of the (random) HTSD $\mathbf{D} = (D_0, D_1, \ldots, D_k)$ and its geometrically decreasing diameter bounding sequence $\mathbf{d} = (d_0, d_1, \ldots, d_k)$ and then establish their desired properties.

Let $D_0 = \{V\}$. Construct a 2-approximate SSSP tree T_0 of G and set $d_0 = \mathrm{diam}(T_0)$. Assume by induction that we have already constructed the TSD D_i, $i \geq 0$, with clusters $C_i^1, \ldots, C_i^q$ and corresponding supporting trees $T_i^1, \ldots, T_i^q$. Set

$$d_i \;=\; \max\left\{\mathrm{diam}(T_i^j) \mid j = 1, \ldots, q\right\}$$

to be the maximum diameter of the trees supporting D_i's clusters. If $d_i = 0$, i.e., all clusters in D_i are singletons, then we set $k = i$ and the construction of $\mathbf{D}$ and $\mathbf{d}$ is completed. Otherwise, we call

$$\texttt{ts\_decompose}\left(G(C_i^j), \Delta_{i+1}\right), \quad j = 1, \ldots, q,$$

where $\Delta_{i+1} = \frac{d_i}{2}$, and take D_{i+1} to be the union of the clusters returned by these q calls.

To analyze this construction, we first observe that since $d_{i+1} \leq \Delta_{i+1} \leq d_i/2$ for every $0 \leq i \leq k-1$, it follows that

$$k \;\leq\; O(\log \sigma)\,.$$

Moreover, each d_i corresponds to the diameter of some approximate SSSP tree w.h.p., thus it provides a constant approximation for the actual distance in G between some pair of nodes. As there are $\binom{n}{2} < n^2$ such node pairs, we conclude that the depth of $\mathbf{d} = (d_0, d_1, \ldots, d_k)$ satisfies

$$k \;\leq\; O\left(n^2\right)$$

w.h.p.

For the expected stretch bound, consider some edge $e \in E$ of length $\ell_e \in \mathbb{Z}_{>0}$. As long as `ts_decompose` is invoked with diameter bound $\Delta_i > 6n\ell_e$, the length of e is reset (see line 3 in Algorithm 2), ensuring that e is not decoupled on that level. For each level i such that $\Delta_i \leq 6n\ell_e$, Corollary 15 guarantees that the probability that e is decoupled on level i is at most

$$O\left(\frac{\ell_e \cdot \log(n)}{\Delta_i}\right) \leq O\left(\frac{\ell_e \cdot \log(n)}{d_i}\right).$$

Taking i_e to be the smallest $i \geq 0$ such that $\Delta_i \leq 6n\ell_e$ and i'_e to be the smallest i such that $\Delta_i \leq \ell_e \log n$, we can bound the expected stretch of e in $\mathbf{D}$ with respect to $\mathbf{d}$ as

$$\begin{aligned}
\mathbb{E}_{\mathbf{D}}[\mathrm{str}_{\mathbf{D},\mathbf{d}}(e)] &\leq \sum_{i=0}^{k-1} \Pr(e \text{ is decoupled on level } i) \cdot \frac{d_i}{\ell_e} \\
&\leq \sum_{i=i_e}^{i'_e-1} O\left(\frac{\ell_e \cdot \log(n)}{d_i}\right) \cdot \frac{d_i}{\ell_e} + \sum_{i=i'_e}^{k-1} O(1) \cdot \frac{d_i}{\ell_e} \\
&\leq (i'_e - i_e) \cdot O(\log n) + O(1)\,,
\end{aligned}$$

where the last transition holds as the sequence $\mathbf{d}$ is geometrically decreasing. Since $\Delta_{i+1} \leq d_i/2 \leq \Delta_i/2$ for every i, it follows, by the definitions of i_e and i'_e, that $i'_e - i_e \leq O(\log n)$, thus yielding the desired bound $\mathbb{E}_{\mathbf{D}}[\mathrm{str}_{\mathbf{D},\mathbf{d}}(e)] \leq O(\log^2 n)$.

It remains to show that the load of every edge $e \in E$ in $\mathbf{D}$ is $O(\log \sigma)$ w.h.p. To that end, recall that in Section 4 we proved that the load on edge e in the TSD D_i is stochastically dominated by a geometric random variable with parameter $\Omega(1)$. The claim follows recalling that the depth $k \leq O(\log \sigma)$ as the sum of k such random variables is $O(k)$ w.h.p. ◀

We note that a crucial point is, of course, that the algorithm can be implemented efficiently due to relying on approximate SSSP computations only. However, as the resulting complexities are model-specific, the respective discussion is postponed to Section 6.

5.2 Embedding into a Random Projected Tree

Consider some graph $G = (V, E, \ell)$. Graph $G' = (V', E', \ell')$ with $V' \supseteq V$ is said to be a *projected* graph of G if there exists a mapping $\pi : V' \to V$ so that
(a) $\pi(v) = v$ for every $v \in V$;
(b) if $e' = \{u', v'\} \in E'$, then $\pi(e') := \{\pi(u'), \pi(v')\} \in E$; and
(c) $\ell'(e') = \ell(e)$ for every $e \in E$ and $e' \in E'$ such that $\pi(e') = e$.
The *load* of edge $e \in E$ under the projected graph G' of G (with respect to π) is defined to be the size of its preimage under π, denoted by $\mathrm{load}_{G'}(e) = |\{e' \in E' \mid \pi(e') = e\}|$. Notice that, by definition, every projected graph of G dominates G. Observe also that ℓ' is fully determined by π and ℓ, hence we may omit it from the notation in the following. Our goal in this section is to prove the following theorem.

▶ **Theorem 17.** *There exists a (randomized) algorithm that, given a graph $G = (V, E, \ell)$ with $\mathrm{poly}(n)$-bounded edge lengths, constructs a random projected tree T of G that satisfies the following guarantees for every edge $e \in E$: (1) $\mathrm{load}_T(e) = O(\log \sigma)$ w.h.p.; and (2) $\mathbb{E}_T[\mathrm{str}_T(e)] = O(\log^2 n)$.*

Theorem 17 is established by combining Theorem 16 with the following lemma.

▶ **Lemma 18.** *There exists an algorithm that given a graph $G = (V, E, \ell)$, a HTSD $\mathbf{D}$ of G, and a geometrically decreasing diameter bounding sequence $\mathbf{d}$ for $\mathbf{D}$, constructs a projected tree $T = (V_T, E_T, \ell_T)$ of G such that $\mathrm{load}_T(e) = \mathrm{load}_{\mathbf{D}}(e)$ and $\mathrm{str}_T(e) = O(\mathrm{str}_{\mathbf{D},\mathbf{d}}(e))$ for each $e \in E$.*

The rest of Section 5.2 is dedicated to proving Lemma 18. This is done by a series of graph transformations that results in the desired projected tree T. Let k be the depth of $\mathbf{D} = (D_0, D_1, \ldots, D_k)$. For $0 \leq i \leq k$, let $H_i = (V_i^H, E_i^H)$ be the forest obtained by taking the (graph) union over all level i supporting trees of $\mathbf{D}$, where each level i supporting tree $T_C = (U_C, F_C)$ contributes its own (distinct) copies of the vertices in U_C and edges in F_C (this means, in particular, that $|V_i^H| = \sum_{C \in D_i} |U_C|$ and $|E_i^H| = \sum_{C \in D_i} |F_C|$). Define the function $\pi_i^H : V_i^H \to V$ by mapping each vertex $v \in V_i^H$ to the vertex $\pi_i^H(v) \in V$ from which it originates, recalling that T_C is a subgraph of G. Although the preimage of vertex $u \in V$ under π_i^H may consist of several vertices, it includes exactly one vertex $v_i \in U_C$, where C is the (unique) level i cluster that contains v. We hereafter refer to this vertex v_i as the level i *clone* of v.

Recalling that the level k clusters of $\mathbf{D}$ are singletons, we identify the vertices in V_k^H with their images under (the bijection) π_k^H so that $V_k^H = V$. Let $H = (V^H, E^H)$ be the forest obtained by taking the (graph) union over $H_0, H_1, \ldots, H_k$ and let $\pi^H : V^H \to V$ be the function defined by mapping each vertex $v \in V_i^H$, $0 \leq i \leq k$, to $\pi^H(v) = \pi_i^H(v)$. Notice that H is a projected graph of G realized by π^H and that $\mathrm{load}_H(e) = \mathrm{load}_{\mathbf{D}}(e)$ for every edge $e \in E$. It remains to show that we can turn H into a projected tree $T = (V_T, E_T)$ by

connecting its connected components without increasing the load on the edges while ensuring that the stretch of every edge $e \in E$ in T is at most $O(1)$ times larger than its stretch in $\mathbf{D}$ with respect to $\mathbf{d}$.

Given a level $0 \leq i \leq k$ and a level i cluster C, we refer to the vertex with smallest ID in C as the *leader* of cluster C, denoted by $\lambda(C)$. Notice that every vertex $v \in V$ is a leader of its level k cluster and that if v is the leader of its level i cluster, then it is also the leader of its level j cluster for all $i \leq j \leq k$.

We now construct a projected tree $T = (V^T, E^T)$ of G from H in two additional steps. First, we connect each connected component T_C of H_i, $1 \leq i \leq k$, to the unique connected component $T_{C'}$ of H_{i-1} that satisfies $C' \supseteq C$. Assuming that the leader of cluster C is $v = \lambda(C)$, this connection is realized by augmenting H with a 0-length edge that connects v_i with v_{i-1}, i.e., the level i and level $i-1$ clones of v. (Note that v_{i-1} is not necessarily the leader of cluster C'.) We call this new edge connecting v_i and v_{i-1} a *vertical* edge and denote the set of all vertical edges added to H during this step of the construction by $E_\uparrow$. Observe that the graph obtained from H by augmenting it with the vertical edges is a tree denoted hereafter by $T^\uparrow = (V^H, E^H \cup E_\uparrow)$. This holds since starting from the forest H, we connected each connected component in level $1 \leq i \leq k$ to a connected component in level $i-1$ using a single vertical edge and since H_0 is a tree.

The next and final step simply contracts all vertical edges in $T^\uparrow$, resulting in the tree $T = (V^T, E^T)$. Since the vertical edge $\{v_i, v_{i-1}\} \in E^\uparrow$ connects the clones v_i and v_{i-1} of the same vertex $v \in V$, it follows that both endpoints of the vertical edge are mapped to v under π^H. Accordingly, we readily obtain a projection $\pi^T \colon V^T \to V$ from π^H by mapping each vertex $v^T \in V^T$ to $\pi^H(v')$, where $v' \in V^H$ is any node that participated in the contraction that created v^T. Finally, note that there is a natural bijection $b \colon E^H \to E^T$ between edges in H and T, as T is obtained by first augmenting H with $E^\uparrow$ of vertical edges and then contracting these edges. By construction, we have $\pi^T(b(e)) = \pi^H(e)$ for all $e \in E^H$. In particular, T is indeed a projected tree of G and $\mathrm{load}_T(e) = \mathrm{load}_H(e) = \mathrm{load}_{\mathbf{D}}(e)$ for all $e \in E$.

It remains to prove that $\mathrm{str}_T(e) = O(\mathrm{str}_{\mathbf{D},\mathbf{d}}(e))$ for every edge $e = \{x, y\} \in E$. Since T is obtained from $T^\uparrow$ by contracting 0-length edges, it follows that $\mathrm{d}_{T^\uparrow}(x, y) = \mathrm{d}_T(x, y)$, hence it suffices to prove that $\mathrm{str}_{T^\uparrow}(e) = O(\mathrm{str}_{\mathbf{D},\mathbf{d}}(e))$. To this end, fix some node $v \in V$ and let $C_i \in D_i$, $0 \leq i \leq k$, be the (unique) level i cluster that contains v. Let $\lambda(i) = \lambda(C_i)$ be the leader of C_i and denote the level j clone of $\lambda(i)$ by $\lambda_j(i)$.

▶ Remark 19. For every $0 \leq i \leq k$, we have $\mathrm{d}_{T^\uparrow}(v, \lambda_i(i)) \leq \sum_{j=i}^{k-1} d_j$.

Proof. By induction on i. The base case $i = k$ holds since every vertex is the leader of its (singleton) level k cluster, hence $\lambda_i(i) = v$. For the inductive step from $i+1$ to $0 \leq i \leq k-1$, we notice that

$$\mathrm{d}_{T^\uparrow}(v, \lambda_i(i)) = \mathrm{d}_{T^\uparrow}(v, \lambda_{i+1}(i+1)) + \mathrm{d}_{T^\uparrow}(\lambda_{i+1}(i+1), \lambda_i(i+1)) + \mathrm{d}_{T^\uparrow}(\lambda_i(i+1), \lambda_i(i)) .$$

Recalling that $\lambda_{i+1}(i+1)$ and $\lambda_i(i+1)$ are connected in $T^\uparrow$ by a vertical edge, we conclude that $\mathrm{d}_{T^\uparrow}(\lambda_{i+1}(i+1), \lambda_i(i+1)) = 0$. Moreover, since $\lambda_i(i+1)$ and $\lambda_i(i)$ belong to the same level i cluster $C_i \in D_i$, their distance in $T^\uparrow$ is equal to their distance in the supporting tree of C_i whose diameter is bounded by d_i, hence $\mathrm{d}_{T^\uparrow}(\lambda_i(i+1), \lambda_i(i)) \leq d_i$. The assertion follows by the inductive hypothesis ensuring that $\mathrm{d}_{T^\uparrow}(v, \lambda_{i+1}(i+1)) \leq \sum_{j=i+1}^{k-1} d_j$. ◀

Now, consider some edge $e = \{u, v\} \in E$ and let $0 \leq i \leq k-1$ be the level on which e is decoupled. Let $C \in D_i$ to be the level i cluster that contains u and v and let w be the level i clone of the leader $\lambda(C)$ of C. Remark 19 guarantees that $\mathrm{d}_{T^\uparrow}(u, w) \leq \sum_{j=i}^{k-1} d_j$ and

$\mathrm{d}_{T^\uparrow}(v,w) \leq \sum_{j=i}^{k-1} d_j$, hence

$$\mathrm{d}_{T^\uparrow}(u,v) \;\leq\; 2\sum_{j=i}^{k-1} d_j \;=\; O(d_i)\,,$$

where the last transition holds since $\mathbf{d} = (d_0, d_1, \ldots, d_k)$ is geometrically decreasing. The proof of Lemma 18 is completed by the definitions of $\mathrm{str}_{\mathbf{D},\mathbf{d}}(e) = \frac{d_i}{\ell_e}$ and $\mathrm{str}_{T^\uparrow}(e) = \frac{\mathrm{d}_{T^\uparrow}(u,v)}{\ell_e}$.

5.3 Embedding into a Random HST

In this section we show how to construct an embedding into a random *hierarchically* 2-*separated* dominating tree (HST) with small expected stretch from the projected trees constructed in the previous section.

▶ **Definition 20** (Hierarchically Separated Trees)**.** *An* embedding *of a weighted graph $G = (V, E, \ell)$ into a (rooted) tree $T = (V^T, E^T, \ell^T)$ is given by a one-to-one mapping $\iota\colon V \to V^T$. For $k > 1$, the tree is* hierarchically k-separated*, if for each internal non-root node, the weight of edges connecting it to its children is exactly by factor k smaller than the weight of the edge connecting it to its parent. The stretch of edge $e = \{u, v\} \in E$ w.r.t. T is defined as* $\mathrm{str}_T(e) := \frac{\mathrm{d}_T(\iota(u),\iota(v))}{\ell_e}$.

We note that our definition of hierarchical well-separation is (formally) weaker than that of hierarchically well-separated trees from the literature [7], as we dropped the requirement that the tree is balanced, i.e., all leaves are in the same depth. However, this can be easily achieved, and our construction does so without modification.

Construction

We construct our HST from a projected tree (see Section 5.2). The construction of $T = (V^T, E^T, \ell^T)$ is straightforward. Let $\mathbf{D} = (D_0, \ldots, D_k)$ be the HTSD from which the projected tree was constructed. We recall that we had assigned a leader $\lambda(C)$ to each cluster C, namely the smallest ID vertex in C. We construct V^T simply as the multiset[4] of leaders of all clusters in $\mathbf{D}$. Note that the nodes constructed for level k clusters, correspond, one-to-one, to the original nodes V of the graph. This enables us to define an embedding $\iota : V \to V^T$ as required in Definition 20. We construct the set of edges E^T as follows: Let $\lambda \in V^T$ be a node corresponding to an arbitrary level i cluster C with $i < k$. We introduce an edge $e := \{\lambda(C), \lambda(C')\}$, for every level $i+1$ cluster C' that cluster C decomposes into, i.e., $C' \subseteq C$. We assign length $\ell^T_e := d_i$ to such an edge e between nodes corresponding to level i and level $i+1$ clusters. Rooting the tree at the node in V^T corresponding to the leader of the (unique) level 0 cluster V, it is clear that the resulting tree $T = (V^T, E^T, \ell^T)$ is a hierarchically 2-separated tree of depth k.

Regarding distances, we get essentially the same result as for the projected tree we could have constructed. Denote for $v \in V$ by $\lambda(i)$ the leader of the unique level i cluster $C_i \in D_i$ such that $v \in C_i$ and denote by $\lambda^T(i) \in V^T$ its copy in T corresponding to C_i.

▶ Remark 21. For every $0 \leq i \leq k$, we have $\mathrm{d}_T(v, \lambda^T(i)) = \sum_{j=i}^{k-1} d_j$.

Proof. $\mathrm{d}_T(v, \lambda^T(i)) = \sum_{j=i}^{k-1} \ell^T_{\{\lambda^T(j),\lambda^T(j+1)\}} = \sum_{j=i}^{k-1} d_j$. ◀

[4] For each cluster C a node $v \in V$ is leader of, there is a separate copy of v.

▶ **Corollary 22.** *T is a dominating hierarchically 2-separated tree with $\mathrm{E}_T[\mathrm{str}_T(e)] = O(\log^2 n)$ for each edge $e \in E$.*

Proof. As discussed, T is hierarchically 2-separated by construction and we have the desired embedding $\iota\colon V \to V^T$. By Remarks 19 and 21, distances between leaves of T are at least as large as in the projected tree constructed in Section 5.2, which dominates G. The stretch bound follows analogous to Section 5.2, where Remark 19 takes the place of Remark 21. ◀

We remark that this establishes a straightforward relation between our projected trees and the HSTs constructed here. The HST edges are realized by the corresponding paths in the projected tree. In particular, while the HST may incur large loads on some graph edges, the "more fine-grained" view provided by the projected tree shows that a low-load mapping of paths in the HST to the original graph is feasible. On the other hand, this relation also demonstrates that a projected tree "behaves" like an HST due to the geometrically decreasing diameter bounding sequence of the underlying HTSD.

5.4 Bounding the p-Stretch

Cohen et al. [16] introduced the notion of p-stretch.

▶ **Definition 23** (p-Stretch)**.** *For a graph G, an embedding of G into T, and a real $p \in (0,1]$, the* p-stretch *of an edge $e = \{u,v\} \in E$ is given by $\left(\frac{\mathrm{d}_T(u,v)}{\ell_e}\right)^p$. Analogously, we define the p-stretch of an HTSD for edge e as $\left(\frac{d_i}{\ell_e}\right)^p$, where i is the level on which e is decoupled.*

Note that the 1-stretch coincides with the definition of the standard stretch defined at the beginning of this section. Our constructions meet a stronger bound of $O(\log n)$ on the p-stretch for $p < 1$, owed to the fact that for $p < 1$ larger stretch is weighed less.

▶ **Lemma 24.** *For $p \in (0,1)$, the tree embeddings presented in Sections 5.2 and 5.3 satisfy that for each edge $e \in E$ the expected p-stretch is $O(\log n)$.*

Proof. When bounding the stretch in the proof of Theorem 16, we summed over all levels of the decomposition. Recall that the probability to decouple edge e on level i is, by Corollary 15, $O\left(\frac{\ell_e \cdot \log n}{d_i}\right)$. Denote by i_e the level such that $d_{i_e} \leq \ell_e < 2d_{i_e}$. If $i > i_e$, then the stretch of e w.r.t. the HTSD is smaller than 1. For $p < 1$, the sum now can thus be bounded as

$$\begin{aligned}\sum_{i=1}^{k} O\left(\frac{\ell_e \cdot \log n}{d_i}\right) \cdot \left(\frac{d_i}{\ell_e}\right)^p &= O\left(1 + \log n \cdot \left(\frac{\ell_e}{d_{i_e}}\right)^{1-p} \cdot \sum_{i=1}^{i_e} \left(\frac{d_{i_e}}{d_i}\right)^{1-p}\right)\\ &= O\left(1 + \log n \cdot \sum_{i=1}^{i_e} \left(\frac{1}{2}\right)^{(1-p)(i_e-i)}\right)\\ &= O(\log n),\end{aligned}$$

where the final step exploits that the sum is a geometric series due to $1 - p > 0$. ◀

6 Implementation in Different Models

In this section, we describe how to implement the above techniques in the CONGEST, PRAM, and multipass streaming models. These should be considered as exemplary computational models and it seems likely that our techniques transfer to other models in which a discrepancy between exact and approximate SSSP computations exist. Under the CONGEST model, some

effort is needed in order to transfer the $(1+\varepsilon)$-approximate SSSP result to $(1+\varepsilon)$-approximate SSSP in super-source graphs (see Definition 2); this transformation is immediate under the PRAM and multipass streaming models.

For simplicity, we assume throughout this section that the edge lengths in the input graph $G = (V, E, \ell)$ are integers in the range $[1, n^{O(1)}]$. This means, in particular, that we do not have to worry about 0-length edges in the blurring procedure (Algorithm 1).

6.1 CONGEST Model

In the CONGEST model of computation [44], every node is a computing unit (of unlimited computational power) and is labeled by a unique $O(\log n)$-bit identifier. Computation proceeds in synchronous rounds, in each of which a node (1) performs local computations, (2) sends $O(\log n)$-bit messages to its neighbors, and (3) receives the messages that its neighbors sent. Initially, every node in the input graph $G = (V, E, \ell)$ knows its identifier and its incident edges together with their length. We note that the restriction to polynomially bounded edge lengths implies that distances can be encoded using $O(\log n)$ bits.

At termination every node needs to know its part of the output. For the task of constructing the random TSD, this means that every node $v \in V$ knows (1) the ID of its own cluster's leader (i.e., the vertex with minimum ID, see Section 5.2); (2) the ID of the leader of cluster C if $v \in U_C$, that is, if v participates in the supporting tree T_C of cluster C; and (3) its incident edges in T_C for each supporting tree T_C in which v participates. For the task of constructing the random HTSD, v should hold that knowledge for every level of the hierarchy. As discussed in Section 5, this also provides the nodes with all what they need in order to reconstruct the resulting projected tree or HST.

In order to avoid confusion with the weighted diameter $\mathrm{diam}(G)$, in what follows, we use $\mathrm{hop}(G)$ to denote the "unweighted" diameter of G, also called the *hop diameter*.

The following corollary discusses how to compute $(1+\varepsilon)$-approximate SSSP in a super-source graph H of a graph G in the CONGEST model. We assume that each node $v \in V$ initially knows which of its incident edges in G are in H, whether it is connected to s, and, if so, the length $\ell(\{s, v\})$.

▶ **Corollary 25** (of [12]). *Let $\varepsilon = \frac{1}{\mathrm{polylog}\, n}$. Then $(1+\varepsilon)$-approximate SSSP in super-source graphs can be solved in $\tilde{O}(\sqrt{n} + \mathrm{hop}(G))$ rounds w.h.p. in the* CONGEST *model.*

Proof. The algorithm from [12] consists of three main steps:

1. Let S be a set composed of s and $\tilde{\Theta}(\sqrt{n})$ nodes sampled uniformly at random. Let each node $v \in S$ learn a $(1+\frac{\varepsilon}{3})$-approximation to the minimum length of $\tilde{O}(\sqrt{n})$-hop paths to each sampled node $w \in S$ (if no such node exists, any result of at least $d(v, w)$ is fine, including ∞). For each finite value, nodes on a (unique) path in G learn about them being part of this path and the next node on it.
2. Simulate a broadcast congested clique[5] $(1+\varepsilon/3)$-approximate SSSP algorithm on the (virtual) graph on S with edge lengths given by the result from the previous step (∞ means no edge).
3. Run $\tilde{O}(\sqrt{n})$ iterations of single source Bellman-Ford on G, where the distance values of nodes in S are initialized to the distances obtained from the previous step.

[5] The broadcast congested clique is the special case of the Congest model restricted to complete graphs and, for each round, nodes sending the same message to each of their neighbors.

Assuming w.l.o.g. that $\varepsilon \leq 1$, this yields $(1+\varepsilon)$-approximate distances to s. As the first step yields suitable routing information and the result of the second (i.e., an approximate SSSP-tree on the virtual graph) is global knowledge, nodes can locally determine their parent in the output tree T.

We adapt the algorithm to super-source graphs as follows.

1. The first step is based on a pipelined version of the (multi-source) Bellman-Ford algorithm that also works on directed graphs [39, Corollary 5.8]. Formally, we orient all edges of s towards it (no other change is made). Then we can easily simulate the procedure on the resulting graph, as all communication by s over one of its edges can be inferred from its length (which is known to the recipient).
   Note that the result is not exactly the same as that of the first step above: all paths containing s as non-starting node have been removed. However, the decisive property of the constructed graph is that it preserves G-distances to s up to a factor of $1+\varepsilon$. The virtual graph also needs to be undirected, which is achieved by dropping the directionality of the computed distances.
2. The simulation of the broadcast congested clique algorithm in the Congest model is based on making all communication global knowledge. Using pipelining over a BFS tree, the input of s in the virtual graph (i.e., its incident edges and their lengths) can be made global knowledge in $\tilde{O}(\sqrt{n}+\mathrm{hop}(G))$ rounds. Together, this implies that all nodes can locally simulate s.
3. Simulating the communication by s in the third step of the algorithm, which is a standard Bellman-Ford computation, is straightforward.

As all steps can be adjusted preserving the guarantees of the algorithm and the asymptotic running time is increased by additive $\tilde{O}(\sqrt{n}+\mathrm{hop}(G))$ only, the result now follows from [12]. ◀

This leads to the following result for Algorithm 1 from Section 3. As it is basically a sequence of approximate SSSP computations, a running time bound is immediate from Corollary 25.

▶ **Corollary 26.** *Suppose* $\alpha = \frac{1}{\mathrm{polylog}\, n}$ *and* $\rho = n^{O(1)}$. *Then Algorithm 1 can be executed in the* CONGEST *model in* $\tilde{O}(\sqrt{n}+\mathrm{hop}(G))$ *rounds w.h.p.*

Proof. The while loop terminates after at most $\lceil \log_{1/\alpha} \rho = O(\log n) \rceil$ iterations. In each iteration, $r^{[i]}$ can be chosen by an arbitrary node (e.g. the one with lowest identifier) and broadcasted via a BFS tree in $O(\mathrm{hop}(G))$ rounds. Each node then can infer from the result from the previous iteration (or the input if $i=1$) whether it is part of $B^{[i-1]}$. Nodes adjacent to $B^{[i-1]}$ can learn about this in one communication round and infer the length of the edge connecting them to $s^{[i]}$ in $G^{[i]}$. Thus, all that remains is the approximate SSSP computation, which can be performed in the stated running time by Corollary 25 w.h.p. The $\tilde{O}$-notation absorbs the $O(\log n)$-factor from the number of loop iterations. ◀

We turn to Algorithm 2 from Section 4. As each iteration can be performed within $\tilde{O}(\sqrt{n}+\mathrm{hop}(G))$ rounds w.h.p., this implies a bound on the running time of the overall algorithm.

▶ **Corollary 27.** *There exists a (randomized)* CONGEST *algorithm that given a graph* $G=(V,E,\ell)$ *with* $\mathrm{poly}(n)$*-bounded edge lengths and a real parameter* $\Delta > 0$*, constructs a random TSD* D *of* G *with the following guarantees in* $\tilde{O}(\sqrt{n}+\mathrm{hop}(G))$ *rounds w.h.p.: (1)* $\mathrm{diam}(D) \leq \Delta$ *w.h.p.; (2)* $\max_{e\in E}\{\mathrm{load}_D(e)\} \leq O(\log n)$ *w.h.p.; and (3)* $\Pr\left[e \in E^{\mathrm{cut}}(D)\right] \leq O\left(\frac{\ell_e \cdot \log n}{\Delta}\right)$ *for every edge* $e \in E$.

Proof. All computations in Algorithm 2 with the exception of the approximate SSSP computations and the call to `blur` are local. By Corollary 26, the stated running time bound follows for a single iteration of the while loop. Here we use that the instances of `blur` can be run in parallel by Lemma 13: As $C_u \subseteq V_u$, we can delete all edges which are not connecting two nodes within the same V_u for some u and then run a single $(1+\varepsilon)$-approximate SSSP instance, where we identify the super-sources of all calls to `blur`. Therefore, Corollary 12 and a union time bound yield the claim. ◀

We now turn to the techniques from Section 5. As the recursive calls for each level of the decomposition hierarchy when computing an HTSD can be executed concurrently with a single call to the approximate SSSP subroutine, we obtain the following result.

▶ **Corollary 28.** *There exists a (randomized)* CONGEST *algorithm that, given a graph $G = (V, E, \ell)$ with $\mathrm{poly}(n)$-bounded edge lengths, constructs a random HTSD $\mathbf{D}$ of G with the following guarantees in $\tilde{O}(\sqrt{n} + \mathrm{hop}(G))$ rounds w.h.p.: (1) the depth of $\mathbf{D}$ is $O(\log n)$; (2) $\mathbf{D}$ admits a (deterministic) geometrically decreasing diameter bounding sequence $\mathbf{d}$ w.h.p.; (3) $\mathrm{load}_{\mathbf{D}}(e) = O(\log n)$ for every edge $e \in E$ w.h.p.; and (4) $\mathbb{E}_{\mathbf{D}}[\mathrm{str}_{\mathbf{D},\mathbf{d}}(e)] = O(\log^2 n)$ for every edge $e \in E$.*

Proof. For each of the $O(\log n)$ levels of the decomposition, the recursive SSSP calls for each of the clusters can be merged into a single one by identifying their super-sources. The claim hence follows from Theorem 16 and Corollary 27. ◀

From the hierarchical decomposition, we obtain embeddings into projected trees and hierarchically 2-separated trees as described in Sections 5.2 and 5.3.

▶ **Corollary 29.** *There exists a (randomized)* CONGEST *algorithm that, given a graph $G = (V, E, \ell)$ with $\mathrm{poly}(n)$-bounded edge lengths, constructs a random projected tree T of G in $\tilde{O}(\sqrt{n} + \mathrm{hop}(G))$ rounds that satisfies the following guarantees for every edge $e \in E$: (1) $\mathrm{load}_T(e) = O(\log n)$ w.h.p.; and (2) $\mathbb{E}_T[\mathrm{str}_T(e)] = O(\log^2 n)$.*

Proof. We obtain an HTSD using Corollary 28. Inspection of the construction in Section 5.2 reveals that all operations are local once we identify the leaders of clusters. This is, e.g., achieved by rooting all supporting trees at the respective cluster's leader, which can be done by using the Garay-Kutten-Peleg minimum spanning tree algorithm [27, 38] to compute a spanning forest of H. As the load of each edge is $O(\log n)$, the algorithm on H can be simulated at a multiplicative overhead of $O(\log n)$, resulting in running time $\tilde{O}(\sqrt{n} + \mathrm{hop}(G))$. ◀

▶ **Corollary 30.** *There exists a (randomized)* CONGEST *algorithm that, given a graph $G = (V, E, \ell)$ with $\mathrm{poly}(n)$-bounded edge lengths, constructs an embedding into a random dominating hierarchically 2-separated tree T of G in $\tilde{O}(\sqrt{n} + \mathrm{hop}(G))$ rounds with expected stretch $\mathbb{E}_T[\mathrm{str}_T(e)] = O(\log^2 n)$ for each edge $e \in E$.*

Proof. Analogous to Corollary 29. ◀

6.2 PRAM Model

In the PRAM model, multiple processors share a random access memory to jointly solve a computational problem. Various contention models exist for concurrent access to the same memory cell by multiple processors, but are equivalent up to small (sub-logarithmic) factors in complexity, so we assume that there is no contention. Then we can view the computation as a DAG whose nodes represent elementary computational steps and edges

dependencies. The input is represented by the sources of the DAG. The crucial complexity measures are *work*, the total size of the DAG (or, equivalently, the sequential complexity of the computation) and *depth*, the maximum length of a path in the DAG (or, equivalently, the time to complete the computation with an unbounded number of processors executing steps at unit speed).

We use a result on approximate SSSP computations due to Cohen, who introduced hop sets for this purpose. Following standard notation, we use $m := |E|$, where $G = (V, E, \ell)$ is the input graph.

▶ **Corollary 31** (of [14, 20]). *Let $\varepsilon_0 > 0$ be a constant and $\varepsilon = \frac{1}{\operatorname{polylog} n}$. Then $(1+\varepsilon)$-approximate SSSP in super-source graphs can be solved in $O(m^{1+\varepsilon_0})$ work and $\operatorname{polylog} n$ time w.h.p.*

We remark that the assumption that the graph G is connected implies $m^{\varepsilon_0} \geq \Omega(n^{\varepsilon_0})$ and thus the term m^{ε_0} can absorb $\operatorname{polylog} n$ factors.

Following the same route as for the CONGEST model, we obtain a string of corollaries. As coordination between processes is easier in the PRAM model, in most cases the results are immediate.

▶ **Corollary 32.** *Suppose $\alpha = \frac{1}{\operatorname{polylog} n}$, $\rho = n^{O(1)}$, and ε_0 is a constant. Then Algorithm 1 can be executed in the PRAM model with depth $\operatorname{polylog} n$ and work $O(m^{1+\varepsilon_0})$ w.h.p.*

▶ **Corollary 33.** *Fix any constant $\varepsilon_0 > 0$. There exists a (randomized) PRAM algorithm of depth $\operatorname{polylog} n$ and work $O(m^{1+\varepsilon_0})$ that, for a graph $G = (V, E, \ell)$ with $\operatorname{poly}(n)$-bounded edge lengths and a real parameter $\Delta > 0$, constructs a random TSD D of G with the following guarantees: (1) $\operatorname{diam}(D) \leq \Delta$ w.h.p.; (2) $\max_{e \in E}\{\operatorname{load}_D(e)\} \leq O(\log n)$ w.h.p.; and (3) $\Pr[e \in E^{\mathrm{cut}}(D)] \leq O\left(\frac{\ell_e \cdot \log n}{\Delta}\right)$ for every edge $e \in E$.*

Combining this corollary with Theorem 16, we obtain the following result.

▶ **Corollary 34.** *Fix any constant $\varepsilon_0 > 0$. There exists a (randomized) PRAM algorithm of depth $\operatorname{polylog} n$ and work $O(m^{1+\varepsilon_0})$ that, for a graph $G = (V, E, \ell)$ with $\operatorname{poly}(n)$-bounded edge lengths, constructs a random HTSD $\mathbf{D}$ of G with the following guarantees w.h.p.: (1) the depth of $\mathbf{D}$ is $O(\log n)$; (2) $\mathbf{D}$ admits a (deterministic) geometrically decreasing diameter bounding sequence $\mathbf{d}$ w.h.p.; (3) $\operatorname{load}_{\mathbf{D}}(e) = O(\log n)$ for every edge $e \in E$ w.h.p.; and (4) $\mathbb{E}_{\mathbf{D}}[\operatorname{str}_{\mathbf{D},\mathbf{d}}(e)] = O(\log^2 n)$ for every edge $e \in E$.*

▶ **Corollary 35.** *Fix any constant $\varepsilon_0 > 0$. There exists a (randomized) PRAM algorithm of depth $\operatorname{polylog} n$ and work $O(m^{1+\varepsilon_0})$ that, given a graph $G = (V, E, \ell)$ with $\operatorname{poly}(n)$-bounded edge lengths, constructs a random projected tree T of G that satisfies the following guarantees for every edge $e \in E$: (1) $\operatorname{load}_T(e) = O(\log n)$ w.h.p.; and (2) $\mathbb{E}_T[\operatorname{str}_T(e)] = O(\log^2 n)$.*

Proof. Again, the main step after obtaining an HTSD is to identify cluster leaders. This can be easily done by pointer jumping within the stated complexity bounds. ◀

▶ **Corollary 36.** *Fix any constant $\varepsilon_0 > 0$. There exists a (randomized) PRAM algorithm of depth $\operatorname{polylog} n$ and work $O(m^{1+\varepsilon_0})$ that, given a graph $G = (V, E, \ell)$ with $\operatorname{poly}(n)$-bounded edge lengths, constructs an embedding into a random dominating hierarchically 2-separated tree T of G with expected stretch $\mathbb{E}_T[\operatorname{str}_T(e)] = O(\log^2 n)$ for each edge $e \in E$.*

6.3 Semi-Streaming Model

In the *streaming model* [32, 41], the input graph is given as a stream of edges without repetitions. The performance of an algorithm is measured by the space it uses, whereby space is organized in memory words of $O(\log n)$ bits. In the *multipass streaming model*, the input is presented to the algorithm in several such passes, and the goal is to keep both the number of required passes and the space consumption small. For algorithms for graph problems, it is usual to assume arbitrary arrival order of the edges. The special case where the computational problem takes an n-vertex graph as input and the amount of memory is $\tilde{O}(n)$ is also known as the *semi-streaming model* [23]. All our results in this subsection are for this setting.

▶ **Corollary 37** (of [12]). *In the semi-streaming model, $(1+\varepsilon)$-approximate SSSP in super-source graphs can be solved in* $\operatorname{polylog} n$ *passes w.h.p. for any* $\varepsilon = \frac{1}{\operatorname{polylog} n}$.

All computational steps that are not SSSP computations can be either directly executed in memory (because only graphs of size $\tilde{O}(n)$ are involved) or easily performed by storing $\operatorname{polylog} n$ words for each node and streaming once (e.g., finding cluster leaders). Thus, corollaries analogous to the CONGEST and PRAM models are immediate.

▶ **Corollary 38.** *Suppose* $\alpha = \frac{1}{\operatorname{polylog} n}$, $\rho = n^{O(1)}$. *Then Algorithm 1 can be executed in the semi-streaming model with* $\operatorname{polylog} n$ *passes w.h.p.*

▶ **Corollary 39.** *There exists a (randomized) semi-streaming algorithm that given a graph $G = (V, E, \ell)$ with* $\operatorname{poly}(n)$*-bounded edge lengths and a real parameter $\Delta > 0$, constructs a random TSD D of G with the following guarantees in* $\operatorname{polylog} n$ *passes w.h.p.: (1)* $\operatorname{diam}(D) \leq \Delta$ *w.h.p.; (2)* $\max_{e \in E}\{\operatorname{load}_D(e)\} \leq O(\log n)$ *w.h.p.; and (3)* $\Pr[e \in E^{\text{cut}}(D)] \leq O\left(\frac{\ell_e \cdot \log n}{\Delta}\right)$ *for every edge $e \in E$.*

▶ **Corollary 40.** *There exists a (randomized) semi-streaming algorithm that given a graph $G = (V, E, \ell)$ with* $\operatorname{poly}(n)$*-bounded edge lengths, constructs a random HTSD $\mathbf{D}$ of G with the following guarantees in* $\operatorname{polylog} n$ *passes w.h.p.: (1) the depth of $\mathbf{D}$ is $O(\log n)$; (2) $\mathbf{D}$ admits a (deterministic) geometrically decreasing diameter bounding sequence $\mathbf{d}$ w.h.p.; (3)* $\operatorname{load}_{\mathbf{D}}(e) = O(\log n)$ *for every edge $e \in E$ w.h.p.; and (4)* $\mathbb{E}_{\mathbf{D}}[\operatorname{str}_{\mathbf{D},\mathbf{d}}(e)] = O(\log^2 n)$ *for every edge $e \in E$.*

Corollary 40 readily implies semi-streaming algorithms that construct low stretch projected trees and dominating hierarchically 2-separated trees in $\operatorname{polylog} n$ passes w.h.p. However, one can, in fact, obtain such constructions in a single pass by (i) partitioning the edges to ($O(\log n)$ many) length classes; (ii) using known techniques to construct a spanner with $O(n)$ edges and $O(\log n)$ stretch for each length class; and (iii) construct the desired low stretch trees for the graph obtained from the union of these spanners in an offline fashion, observing that this graph has $O(n \log n)$ edges.[6]

7 Related Work

Low diameter graph decompositions with small edge cutting probabilities (or with small weight) play a major role in many algorithmic applications. These include the construction of low stretch spanning trees [2, 3, 4, 11, 18] and low distortion probabilistic embeddings of

[6] We thank an anonymous ITCS 2020 reviewer for pointing out this simple alternative construction.

metric spaces into hierarchically well-separated trees [7, 8, 9, 22], fast approximate solvers of symmetric diagonally dominant linear systems [15, 36, 37, 47], constructing graph spanners [42, 45], and spectral sparsification [33, 35]. The literature in this field being vast, we can only give an incomplete review of it. We first focus on related work in distributed and parallel models of computation, as these results are closest to ours, and then turn to the related work in the streaming model. Our discussion of the related work in the former models starts with reviewing the literature on low diameter graph decompositions, and then it turns to their applications, focusing on low average stretch spanning trees and tree embeddings.

Low Diameter Graph Decompositions

In the LOCAL and CONGEST models of distributed computation[7] low diameter graph decompositions for unweighted graphs, i.e., $G = (V, E, \mathbb{1})$, play a special role as they can be leveraged to design fast algorithms for a large class of problems. More precisely, the decomposition task is complete for a certain class of local problems [30], where a problem is called local if it does not require $\Omega(\mathrm{hop}(G))$ rounds of communication (recall the definition of the hop diameter $\mathrm{hop}(G)$ from Section 6.1). Here, $\mathrm{hop}(G)$ is of relevance even in problems where the input graph is weighted, as communication over large hop distances is an inherent obstacle to small running times in distributed algorithms.

Several distributed decomposition algorithms with round complexities of $\mathrm{polylog}\, n$ and small edge cut probabilities are known for the unweighted case [19, 40, 43].[8] However, the weighted setting considered in this work is fundamentally different. A lower bound of $\mathrm{hop}(G)$ is trivial, i.e., the task is not local: intuitively, decoupling hop distance from graph distance implies that finding close-by nodes may require communication over $\mathrm{hop}(G)$ hops. In the LOCAL model, this bound is trivially tight, as nodes can learn about the entire graph in $\mathrm{hop}(G)$ rounds. In the CONGEST model, a reduction from 2-party communication complexity shows a lower bound of $\Omega\left(\frac{\sqrt{n}}{\log n}\right)$ rounds for computing an (r, λ)-decomposition for any non-trivial values of r and λ. This lower bound even holds if $\mathrm{hop}(G) = O(\log n)$ [46].[9]

Miller et al. [43] show how to compute low diameter graph decomposition with small edge cutting probabilities in unweighted graphs in the PRAM model. Their approach relies on exact SSSP computations. Given the current discrepancy in the state of the art of exact and approximate SSSP in the PRAM model, it thus cannot lead to satisfying bounds in the weighted setting.

Low Stretch Spanning Trees

Nevertheless, there has been some work applying decompositions in the vein of Miller et al. in order to obtain low average stretch[10] spanning trees for weighted graphs. A construction by Alon et al. [4] reduces weighted graphs to unweighted (multi)graphs. As a result Blelloch et al. [13] were able to give an efficient PRAM construction of low stretch spanning trees based on the decomposition technique by Miller et al. As shown by Blelloch et al., computation

[7] See Section 6.1 for the formal definition of CONGEST. The LOCAL model is identical, except that it does not restrict message sizes to $O(\log n)$.

[8] Some works only care about the chromatic number of the graph resulting from contracting clusters. However, the cited works achieve this by cutting few edges only.

[9] A low-diameter decomposition can be used to determine whether or not there is a light s-t cut in the family of lower bound graphs from [46]; s and t end up in the same cluster if and only if there is no light cut between them, as otherwise their distance is large.

[10] The ratio of distance in the tree to edge length, averaged over all edges.

of such trees is of use for efficient PRAM solvers for symmetric diagonally dominant linear systems. A similar connection was exploited by Ghaffari et al. [29], who transferred the approach of Blelloch et al. to the CONGEST model, obtaining a low average stretch tree construction that they leveraged for approximate maximum flow computations. A downside of the aforementioned approaches is that the construction by Alon et al. suffers from a poor average stretch of $2^{\Theta(\sqrt{\log n \log\log n})}$, resulting in respective overheads in work and depth resp. round complexity in the two models when applying the computed trees in further computations.

For the CONGEST model, Becker et al. [11] gave a construction of low-average stretch spanning trees that combines the decomposition technique of Miller et al. with the star decomposition technique of Elkin et al. [18]. This approach achieves polylog n average stretch. Again, the complexity of their approach is essentially determined by an exact SSSP computation. Thus, the resulting algorithm is round-optimal up to polylogarithmic factors in the unweighted case (i.e., the running time is $\mathrm{hop}(G)\,\mathrm{polylog}\, n$), while essentially matching the round complexity of exact SSSP in the weighted case. Exact SSSP computation in the CONGEST model is still not too well understood, with the best upper bound of $\tilde{O}(\min\{\sqrt{n\,\mathrm{hop}(G)}, \sqrt{n}\,\mathrm{hop}(G)^{1/4} + n^{3/5} + \mathrm{hop}(G)\})$ [25] still being polynomially far from the $\tilde{\Omega}(\sqrt{n} + \mathrm{hop}(G))$ lower bound.

Tree Embeddings

We apply our decomposition technique in order to obtain a metric tree embedding, following the same route as Bartal [7], obtaining the same $O(\log^2 n)$ bound on the expected stretch (note that the bound in [7] holds for any edge lengths whereas in the current paper, we make the simplifying assumption that the ratio of the maximum to minimum edge length is $\mathrm{poly}(n)$). Bartal later improved this bound to $O(\log n \log\log n)$ [8] and subsequently to asymptotically optimal $O(\log n)$ [9]. Although we cannot readily apply the same techniques, Bartal's work suggests that future improvements to our stretch bound are feasible.

Fakcharoenphol et al. [22] achieved the $O(\log n)$ stretch bound earlier, following a different approach in which the graph is not (explicitly) decomposed. However, at its core the main idea is very similar: randomization is leveraged to keep the probability of "cutting" edges proportional to their length based on the subtractive form of the triangle inequality. Also here, PRAM and CONGEST algorithms have been developed that try to mitigate the bottleneck imposed by exact SSSP computations. In the CONGEST model, it is straightforward to implement the algorithm from [22] with a round complexity that is (up to a factor of $O(\log n)$) equal to the running time of the Bellman-Ford algorithm [34]. However, shortest paths may have hop length up to $n-1$, resulting in a running time far from the $\tilde{\Omega}(\sqrt{n} + \mathrm{hop}(G))$ lower bound. Ghaffari and Lenzen broke down shortest paths by sampling a "skeleton" of $\tilde{\Theta}(\sqrt{n})$ nodes uniformly, computing a spanner (refer to the sequel of this section for the definition of a spanner) of a graph representing the induced metric, computing a tree embedding of this spanner, and finally extending this embedding to one of the original graph with modified weights via a Bellman-Ford computation. This can be seen as distorting the original distance metric such that it becomes sufficiently simple to solve exact SSSP fast, resulting in a round complexity of $\tilde{O}(n^{0.5+\varepsilon} + \mathrm{hop}(G))$ for stretch $O(\varepsilon^{-1}\log n)$. In particular, by setting $\varepsilon = \frac{1}{\log n}$, the stretch and running time bounds match our results. However, Ghaffari and Lenzen do not guarantee bounded load. We also note that their approach is inherently limited to stretch $\Omega(\log^2 n)$ when requiring a running time bound within polylog n of the lower bound, as both spanners with a near-linear number of edges and metric tree embedding must incur $\Omega(\log n)$ stretch each.

Friedrichs and Lenzen [26] provide fast PRAM and CONGEST algorithms for tree embeddings with stretch $O(\log n)$. The main difference to [31] is the use of hop sets [14] to provide "shortcuts" for distance computation that are not present in the original graph. Again, distances are then distorted by metric embeddings such that exact distance computation by a Bellman-Ford style computation becomes efficient. This leads to a $2^{O(\sqrt{\log n})}(\sqrt{n} + \mathrm{hop}(G))$-round algorithm in CONGEST and a PRAM algorithm of depth $\mathrm{polylog}\, n$ and work $O(m^{1+\varepsilon})$ (for any fixed constant $\varepsilon > 0$), where m is the number of edges and $\Omega(m)$ a trivial lower bound on the work. While the stretch guarantee is better than in our case, it should be noted that also here fundamental barriers limit this technique: lower bounds on the size of hop sets due to Abboud et al. [1] imply that any hop-set based approach must incur running time resp. work overheads of $2^{\Omega(\sqrt{\log n})}$. Although in the PRAM model we suffer the same work overhead by relying on hop-sets for the currently best known approximate SSSP algorithms [14, 20], our result shows that one can trade the additional log-factor in stretch for a logarithmic load bound that the method of Friedrichs and Lenzen cannot guarantee.

Streaming Algorithms

To the best of our knowledge, constructions of low diameter decompositions with small edge cutting probabilities have not been addressed so far in the semi-streaming literature. A related graph theoretic object whose construction has been studied in the context of streaming algorithms is *spanners*. Similarly to low (average) spanning trees, spanners also provide a sparse distance preserving representation of the graph, only that they are not required to be trees. On the other hand, their notion of distance preservation is stronger in the sense that it is required to hold in the worst case, rather than on average. Specifically, a κ-spanner of graph $G = (V, E, \ell)$ is a spanning subgraph of G that guarantees a stretch bound of at most κ for every edge in E. One is typically interested in constructing κ-spanners with a small number of edges, where $O(n^{1+2/(\kappa+1)})$ edges is the asymptotically tight bound. Streaming constructions of sparse spanners exist only for unweighted graphs [10, 17, 23], as there, the distance computations are typically restricted to the sparse subgraph maintained by the algorithm. A related notion in unweighted graphs, which has also been studied in the streaming literature [21], is an (α, β)-spanner, where the distance between vertices $u, v \in V$ in the spanner is required to be at most $\alpha \cdot \mathrm{d}_G(u, v) + \beta$ for every $u, v \in V$.

References

1. Amir Abboud, Greg Bodwin, and Seth Pettie. A Hierarchy of Lower Bounds for Sublinear Additive Spanners. *SIAM J. Comput.*, 47(6):2203–2236, 2018.
2. Ittai Abraham, Yair Bartal, and Ofer Neiman. Nearly Tight Low Stretch Spanning Trees. In *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS*, pages 781–790, 2008.
3. Ittai Abraham and Ofer Neiman. Using petal-decompositions to build a low stretch spanning tree. In *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC*, pages 395–406, 2012.
4. Noga Alon, Richard M. Karp, David Peleg, and Douglas B. West. A Graph-Theoretic Game and Its Application to the k-Server Problem. *SIAM J. Comput.*, 24(1):78–100, 1995.
5. Baruch Awerbuch. Complexity of Network Synchronization. *J. ACM*, 32(4):804–823, 1985.
6. Baruch Awerbuch and David Peleg. Sparse Partitions (Extended Abstract). In *31st Annual Symposium on Foundations of Computer Science, St. Louis, Missouri, USA, October 22-24, 1990, Volume II*, pages 503–513, 1990.

7 Yair Bartal. Probabilistic Approximations of Metric Spaces and Its Algorithmic Applications. In *37th Annual Symposium on Foundations of Computer Science, FOCS*, pages 184–193, 1996.

8 Yair Bartal. On Approximating Arbitrary Metrices by Tree Metrics. In *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, STOC*, pages 161–168, 1998.

9 Yair Bartal. Graph Decomposition Lemmas and Their Role in Metric Embedding Methods. In *Algorithms - ESA 2004, 12th Annual European Symposium*, pages 89–97, 2004.

10 Surender Baswana. Streaming algorithm for graph spanners - single pass and constant processing time per edge. *Inf. Process. Lett.*, 106(3):110–114, 2008.

11 Ruben Becker, Yuval Emek, Mohsen Ghaffari, and Christoph Lenzen. Distributed Algorithms for Low Stretch Spanning Trees. In *33rd International Symposium on Distributed Computing, DISC*, 2019. To appear.

12 Ruben Becker, Andreas Karrenbauer, Sebastian Krinninger, and Christoph Lenzen. Near-Optimal Approximate Shortest Paths and Transshipment in Distributed and Streaming Models. In *31st International Symposium on Distributed Computing, DISC 2017, October 16-20, 2017, Vienna, Austria*, pages 7:1–7:16, 2017.

13 Guy E. Blelloch, Anupam Gupta, Ioannis Koutis, Gary L. Miller, Richard Peng, and Kanat Tangwongsan. Nearly-Linear Work Parallel SDD Solvers, Low-Diameter Decomposition, and Low-Stretch Subgraphs. *Theory of Computing Systems*, 55(3):521–554, 2014.

14 Edith Cohen. Polylog-time and near-linear work approximation scheme for undirected shortest paths. *J. ACM*, 47(1):132–166, 2000.

15 Michael B. Cohen, Rasmus Kyng, Gary L. Miller, Jakub W. Pachocki, Richard Peng, Anup B. Rao, and Shen Chen Xu. Solving SDD linear systems in nearly $m\log^{1/2}n$ time. In *Symposium on Theory of Computing, STOC*, pages 343–352, 2014.

16 Michael B. Cohen, Gary L. Miller, Jakub W. Pachocki, Richard Peng, and Shen Chen Xu. Stretching Stretch. *CoRR*, abs/1401.2454, 2014. `arXiv:1401.2454`.

17 Michael Elkin. Streaming and fully dynamic centralized algorithms for constructing and maintaining sparse spanners. *ACM Trans. Algorithms*, 7(2):20:1–20:17, 2011.

18 Michael Elkin, Yuval Emek, Daniel A. Spielman, and Shang-Hua Teng. Lower-Stretch Spanning Trees. *SIAM J. Comput.*, 38(2):608–628, 2008.

19 Michael Elkin and Ofer Neiman. Distributed Strong Diameter Network Decomposition: Extended Abstract. In *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing*, pages 211–216, 2016.

20 Michael Elkin and Ofer Neiman. Hopsets with Constant Hopbound, and Applications to Approximate Shortest Paths. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 128–137, 2016.

21 Michael Elkin and Jian Zhang. Efficient algorithms for constructing (1+epsilon, beta)-spanners in the distributed and streaming models. *Distributed Computing*, 18(5):375–385, 2006.

22 Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar. A tight bound on approximating arbitrary metrics by tree metrics. *J. Comput. Syst. Sci.*, 69(3):485–497, 2004.

23 Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. On graph problems in a semi-streaming model. *Theor. Comput. Sci.*, 348(2-3):207–216, 2005.

24 Sebastian Forster and Gramoz Goranci. Dynamic low-stretch trees via dynamic low-diameter decompositions. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019.*, pages 377–388, 2019.

25 Sebastian Forster and Danupon Nanongkai. A Faster Distributed Single-Source Shortest Paths Algorithm. In *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 686–697, 2018.

26 Stephan Friedrichs and Christoph Lenzen. Parallel Metric Tree Embedding Based on an Algebraic View on Moore-Bellman-Ford. *J. ACM*, 65(6):43:1–43:55, 2018.

27 Juan A Garay, Shay Kutten, and David Peleg. A sublinear time distributed algorithm for minimum-weight spanning trees. *SIAM Journal on Computing*, 27(1):302–316, 1998.

28 Mohsen Ghaffari and Bernhard Haeupler. Distributed Algorithms for Planar Networks II: Low-congestion Shortcuts, MST, and Min-Cut. In *Proceedings of the Twenty-seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 202–219, 2016.

29 Mohsen Ghaffari, Andreas Karrenbauer, Fabian Kuhn, Christoph Lenzen, and Boaz Patt-Shamir. Near-Optimal Distributed Maximum Flow. *SIAM Journal on Computing*, 47(6):2078–2117, 2018.

30 Mohsen Ghaffari, Fabian Kuhn, and Yannic Maus. On the Complexity of Local Distributed Graph Problems. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2017, pages 784–797, 2017.

31 Mohsen Ghaffari and Christoph Lenzen. Near-Optimal Distributed Tree Embedding. In *Distributed Computing - 28th International Symposium, DISC*, pages 197–211, 2014.

32 Monika Rauch Henzinger, Prabhakar Raghavan, and Sridhar Rajagopalan. Computing on data streams. In *External Memory Algorithms, Proceedings of a DIMACS Workshop, New Brunswick, New Jersey, USA, May 20-22, 1998*, pages 107–118, 1998.

33 Michael Kapralov and Rina Panigrahy. Spectral sparsification via random spanners. In *Innovations in Theoretical Computer Science 2012, Cambridge, MA, USA, January 8-10, 2012*, pages 393–398, 2012.

34 Maleq Khan, Fabian Kuhn, Dahlia Malkhi, Gopal Pandurangan, and Kunal Talwar. Efficient distributed approximation algorithms via probabilistic tree embeddings. *Distributed Computing*, 25(3):189–205, 2012.

35 Ioannis Koutis. Simple parallel and distributed algorithms for spectral graph sparsification. In *26th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA '14, Prague, Czech Republic - June 23 - 25, 2014*, pages 61–66, 2014.

36 Ioannis Koutis, Gary L. Miller, and Richard Peng. A Nearly-m log n Time Solver for SDD Linear Systems. In *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 590–598, 2011.

37 Ioannis Koutis, Gary L. Miller, and Richard Peng. Approaching Optimality for Solving SDD Linear Systems. *SIAM J. Comput.*, 43(1):337–354, 2014.

38 Shay Kutten and David Peleg. Fast Distributed Construction of Smallk-Dominating Sets and Applications. *Journal of Algorithms*, 28(1):40–66, 1998.

39 Christoph Lenzen, Boaz Patt-Shamir, and David Peleg. Distributed distance computation and routing with small messages. *Distributed Computing*, 2018.

40 Nathan Linial and Michael Saks. Low diameter graph decompositions. *Combinatorica*, 13(4):441–454, 1993.

41 Andrew McGregor. Graph stream algorithms: a survey. *SIGMOD Record*, 43(1):9–20, 2014.

42 Gary L. Miller, Richard Peng, Adrian Vladu, and Shen Chen Xu. Improved Parallel Algorithms for Spanners and Hopsets. In *Proceedings of the 27th ACM on Symposium on Parallelism in Algorithms and Architectures, SPAA 2015, Portland, OR, USA, June 13-15, 2015*, pages 192–201, 2015.

43 Gary L. Miller, Richard Peng, and Shen Chen Xu. Parallel graph decompositions using random shifts. In *25th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA '13, Montreal, QC, Canada - July 23 - 25, 2013*, pages 196–203, 2013.

44 D. Peleg. *Distributed Computing: A Locality-Sensitive Approach*. Discrete Mathematics and Applications. Society for Industrial and Applied Mathematics, 2000.

45 David Peleg and Alejandro A. Schäffer. Graph spanners. *Journal of Graph Theory*, 13(1):99–116, 1989.

46 Atish Das Sarma, Stephan Holzer, Liah Kor, Amos Korman, Danupon Nanongkai, Gopal Pandurangan, David Peleg, and Roger Wattenhofer. Distributed Verification and Hardness of Distributed Approximation. *SIAM J. Comput.*, 41(5):1235–1265, 2012.

47 Daniel A. Spielman and Shang-Hua Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004*, pages 81–90, 2004.

Generalized List Decoding

Yihan Zhang
Department of Information Engineering, The Chinese University of Hong Kong
https://sites.google.com/view/yihan/
zy417@ie.cuhk.edu.hk

Amitalok J. Budkuley
Department of Electronics and Electrical Communication Engineering,
Indian Institute of Technology Kharagpur, India
http://www.facweb.iitkgp.ac.in/~amitalok
amitalok@ece.iitkgp.ac.in

Sidharth Jaggi
Department of Information Engineering, The Chinese University of Hong Kong
https://scholar.google.com.hk/citations?user=AX7276AAAAAJ&sortby=pubdate
jaggi@ie.cuhk.edu.hk

Abstract

This paper concerns itself with the question of list decoding for *general adversarial channels*, e.g., bit-flip (XOR) channels, erasure channels, AND (Z-) channels, OR (Σ-) channels, real adder channels, noisy typewriter channels, etc. We precisely *characterize* when exponential-sized (or positive *rate*) $(L-1)$-*list decodable* codes (where the *list size* L is a universal constant) exist for such channels. Our criterion essentially asserts that:

> For any given general adversarial channel, it is possible to construct positive rate $(L-1)$-list decodable codes *if and only if* the set of *completely positive tensors* of order-L with admissible marginals is not entirely contained in the order-L *confusability set* associated to the channel.

The sufficiency is shown via random code construction (combined with expurgation or time-sharing). The necessity is shown by

1. extracting approximately equicoupled subcodes (generalization of equidistant codes) from *any* sequence of "large" codes using hypergraph Ramsey's theorem, and
2. significantly extending the classic *Plotkin bound* in coding theory to list decoding for general channels using duality between the completely positive tensor cone and the *copositive* tensor cone.

In the proof, we also obtain a new fact regarding asymmetry of joint distributions, which may be of independent interest.

Other results include

1. List decoding capacity with asymptotically large L for general adversarial channels;
2. A *tight* list size bound for *most constant composition* codes (generalization of constant weight codes);
3. Rederivation and demystification of Blinovsky's [9] characterization of the list decoding *Plotkin points* (threshold at which large codes are impossible) for bit-flip channels;
4. Evaluation of general bounds ([43]) for *unique decoding* in the error correction code setting.

2012 ACM Subject Classification Mathematics of computing → Coding theory; Mathematics of computing → Information theory

Keywords and phrases Generalized Plotkin bound, general adversarial channels, equicoupled codes, random coding, completely positive tensors, copositive tensors, hypergraph Ramsey theory

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.51

Related Version An extended version of the paper will be updated at https://arxiv.org/abs/1909.04264.

Acknowledgements We thank Andrej Bogdanov who provided an elegant reduction from general L to $L = 2$ for the proof of the asymmetric case of the converse (Lemma 68) and rederived Blinovsky's [9] characterization of the Plotkin point P_{L-1} for $(p, L-1)$-list decoding via a conceptually cleaner proof (Sec. 16), despite that he generously declined to co-author this paper. We also thank him for inspiring discussions in the early stage and helpful comments near the end of this work.
Part of this work was done while YZ was visiting the Simons Institute for the Theory of Computing for the Summer Cluster: Error-Correcting Codes and High-Dimensional Expansion, and AJB was at the Department of Information Engineering, the Chinese University of Hong Kong. This work was partially supported by GRF grants 14301519 and 14313116.

1 Warmup

In favour of motivating general problems, introducing general notions and stating our general theorems, we first go through concrete numerical examples that are special cases of our results.

Suppose Alice can transmit a length-n bit string (*codeword*) to Bob and an adversary James can flip np $(0 \leqslant p \leqslant 1)$ of these bits. Consider first the classic coding theory question.

1. **Error correction.** For what values of p, can one construct a *code* (collection of codewords) of *positive rate* (i.e., codebook size at least 2^{Rn} for some constant $1 \geqslant R > 0$) such that Bob can uniquely decode? The classic Plotkin bound [33] tells us that this is impossible for $p > 1/4$,[1] and the classic Gilbert–Varshamov (GV) bound [22, 42] tells us that this is possible for $p < 1/4$.
2. **List decoding.** For what values of p, can one construct a code of positive rate such that it is *3-list decodable* (i.e., regardless of which np bits James flips, Bob can always decode the received word to a *list* of at most 3 codewords, one of which is the codeword transmitted by Alice)?[2] Due to work by Blinovsky, it is known that this is possible if and only if $p \leqslant 5/16$.[3]

In this work, not only are we are able to rederive all the above thresholds, but are also able to derive the corresponding thresholds for a vast variety of *general adversarial channels* such as bit-flip channels, erasure channels, AND (Z-) channels, OR (Σ-) channels, adder channels, noisy typewriter channels, etc..

In this section, let us revisit the answers to questions 1 and 2 in the technical language we develop in this paper.

1. **Error correction.** Consider any pair of codewords $\underline{x}_1, \underline{x}_2$ that are resilient to np bit-flips. They must therefore be at a Hamming distance larger than $2np$. Said differently, the *joint type* (i.e., the 2×2 matrix whose (x_1, x_2)-th, $x_1, x_2 \in \{0, 1\}$, entry is the fraction of locations i of $(\underline{x}_1, \underline{x}_2)$ such that $\underline{x}_1(i) = x_1$ and $\underline{x}_2(i) = x_2$) $\tau_{\underline{x}_1, \underline{x}_2} = \begin{bmatrix} t(0,0) & t(0,1) \\ t(1,0) & t(1,1) \end{bmatrix}$ of these two codewords must satisfy the condition that

   C1 $t(0,1) + t(1,0) \geqslant 2p$.

[1] Actually for $p = 1/4$ this is still impossible.
[2] Note that a 1-list decodable code is exactly a uniquely decodable code (or more commonly called an *error correction code*).
[3] In fact Blinovsky identified the threshold p up to which positive rate $(p, L-1)$-list decodable codes exist for *any* integer $L \geqslant 2$. This, in particular, recovers the Plotkin bound.

a. In [9, 34, 2][4] and [43], it was shown that: if a code $\mathcal{C}$ of size 2^{Rn} exists, then there must exist a *positive rate* subcode $\mathcal{C}' \subset \mathcal{C}$ such that for *every* pair of codewords $\underline{x}_1, \underline{x}_2$ in $\mathcal{C}'$, their joint type is approximately the same (as, say, $P_{\mathbf{x}_1,\mathbf{x}_2}$).

b. In [43], it was shown that: it is possible to construct positive rate codes with joint types (close to) $P_{\mathbf{x}_1,\mathbf{x}_2}$ if and only if $P_{\mathbf{x}_1,\mathbf{x}_2}$ is a *completely positive (*CP*)* distribution, i.e., $P_{\mathbf{x}_1,\mathbf{x}_2}$ can be written as a convex combination of products of independent and identical distributions,

$$P_{\mathbf{x}_1,\mathbf{x}_2} = \sum_{i=1}^{k} \lambda_i P_{\mathbf{x}_i} P_{\mathbf{x}_i}^{\top},$$

for some positive integer k, convex combination coefficients $\{\lambda_i\}_{1\leqslant i\leqslant k}$ and probability vectors $\{P_{\mathbf{x}_i}\}_{1\leqslant i\leqslant k}$. For example,

$$\lambda \begin{bmatrix} 1/2 & 0 \\ 0 & 1/2 \end{bmatrix} + (1-\lambda) \begin{bmatrix} 1/4 & 1/4 \\ 1/4 & 1/4 \end{bmatrix} \tag{1}$$

is CP for $\lambda \in [0,1]$ since it can be written as $\frac{\lambda}{2}\begin{bmatrix}1\\0\end{bmatrix}\begin{bmatrix}1 & 0\end{bmatrix} + \frac{\lambda}{2}\begin{bmatrix}0\\1\end{bmatrix}\begin{bmatrix}0 & 1\end{bmatrix} + (1-\lambda)\begin{bmatrix}1/2\\1/2\end{bmatrix}\begin{bmatrix}1/2 & 1/2\end{bmatrix}$. One can check that for $\lambda < 0$, matrix (1) is not CP. For condition **C1** to be satisfied by some CP distribution, it must be the case that $2p \leqslant 2\cdot(1-\lambda)\cdot(1/4)$ for some $\lambda \in [0,1]$. This is impossible if $p > 1/4$. As a consequence, the classic Plotkin bound is recovered in this convex geometry language, since the non-CP matrices of the form (1) with *negative* λ correspond to codes with minimum pairwise fractional distance $\frac{1+|\lambda|}{2}$ (hence correspond to $p = \frac{1+|\lambda|}{4} > 1/4$), which, by the Plotkin bound, cannot have positive rate.

2. **List decoding.** Now let us move to the list decoding question in hands. For a code to be 3-list decodable, it must be the case that for any quadruple $\underline{x}_1, \underline{x}_2, \underline{x}_3, \underline{x}_4$, there is no $\underline{y}$ such that the Hamming distance from $\underline{x}_i$ to $\underline{y}$ is at most np for *every* $i \in \{1,2,3,4\}$. In this case, the appropriate object is therefore a $2\times2\times2\times2$ tensor (or a joint distribution of $(\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3,\mathbf{x}_4)$) $P_{\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3,\mathbf{x}_4}$ such that

C2 any of its *extension* $P_{\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3,\mathbf{x}_4,\mathbf{y}}$ (i.e., a coupling of $(\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3,\mathbf{x}_4)$ and $\mathbf{y}$, or a $2\times2\times2\times2\times2$ tensor such that $P_{\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3,\mathbf{x}_4} = P_{\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3,\mathbf{x}_4,\mathbf{y}=0} + P_{\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3,\mathbf{x}_4,\mathbf{y}=1}$) satisfies the condition that $P_{\mathbf{x}_i,\mathbf{y}}(0,1) + P_{\mathbf{x}_i,\mathbf{y}}(1,0) > p$ for at least one $i \in \{1,2,3,4\}$.

a. Again, by [9, 34, 2] and our work, we can restrict our attention to codes in which every 4-tuple of codewords has joint type close to some $P_{\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3,\mathbf{x}_4}$, since we can find such a subcode which is sufficiently large in *any* positive rate code.

b. Generalizing [43], we show that codes with order-4 joint types (close to) $P_{\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3,\mathbf{x}_4}$ exist if and only if $P_{\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3,\mathbf{x}_4}$ is a *completely positive tensor* of order-4, i.e., $P_{\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3,\mathbf{x}_4}$ can be written as a convex combination of products of independent and identical distributions,

$$P_{\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3,\mathbf{x}_4} = \sum_{i=1}^{k} \lambda_i P_{\mathbf{x}_i}^{\otimes 4}.$$

[4] Their and our work showed that it is also possible to find a positive rate subcode such that every L-tuple of codewords has joint type close to some $P_{\mathbf{x}_1,\cdots,\mathbf{x}_L}$. This, as we shall see momentarily, is useful for list decoding.

One can check that distributions of the form

$$\lambda \operatorname{diag}(1/2) + (1-\lambda)\begin{bmatrix}1/2\\1/2\end{bmatrix}^{\otimes 4} = \frac{\lambda}{2}\begin{bmatrix}1\\0\end{bmatrix}^{\otimes 4} + \frac{\lambda}{2}\begin{bmatrix}0\\1\end{bmatrix}^{\otimes 4} + (1-\lambda)\begin{bmatrix}1/2\\1/2\end{bmatrix}^{\otimes 4}$$

is CP if and only if $\lambda \in [0,1]$. On the other hand, for condition **C2** to be satisfied by some tensor like that, it turns out, as shown by Blinovsky [9] and us, that p has to be no larger than 5/16.

Of course, bit-flips are just one of the simplest models of corruption that may occur in real-world communication/storage systems. Perhaps, under certain circumstances, in the system of interest, we are allowed to transmit length-n codewords taking values from $\{0,1,2,3,4,5\}$, but each legitimate codeword $\underline{x}$ has to satisfy the following constraints inherently associated to this communication system

$$\begin{cases} \tau_{\underline{x}}(1) + 3\tau_{\underline{x}}(3) \leqslant 1.2 \\ \tau_{\underline{x}}(2) - \tau_{\underline{x}}(3) \geqslant 0.05 \\ \tau_{\underline{x}}(0) - \tau_{\underline{x}}(4) - 0.2\tau_{\underline{x}}(5) \leqslant 0.7 \end{cases},$$

where $\tau_{\underline{x}}(x)$ denotes the fraction of $x \in \{0,1,\cdots,5\}$ in $\underline{x} \in \{0,1,\cdots,5\}^n$. An adversary is allowed to change symbols in the transmitted codeword only from small values to large values; the cost he pays by changing every i to j $(0 \leqslant i < j \leqslant 5)$ is $j-i$ dollars, and he has a budget of $2.3n$ dollars in total. Among others, one of the fundamental questions we are able to answer in this paper is the following: is it possible for us to design exponentially large codes such that no matter which codeword is transmitted and no matter how an adversary corrupts it via a legitimate action, the decoder is always able to output a list of at most (say) 10 codewords which contains the correct one?

The answer to the above question is affirmative and can be stated in a similar manner: it is possible if and only if there is a CP tensor of order 11 and dimension 6 which does not lie inside the *confusability set* determined by the channel. In particular, the confusability set is the set of joint distributions which fail to meet the conditions similar to **C1** or **C2** that are determined by the channel.

Our results tell us that if one only aims to search for exponentially large $(L-1)$-list decodable codes (instead of optimizing its size) for a given general adversarial channel, then it is *sufficient* (and obviously necessary) to restrict our attention to codes that are *chunk-wise random-like*. Such codes correspond to some CP distribution $\sum_{i=1}^{k}\lambda_i P_{\mathbf{x}_i}^{\otimes L}$. If a random code of positive rate, where the $\lambda_i n$ $(1 \leqslant i \leqslant k)$ components in the i-th chunk of each codeword are sampled from distribution $P_{\mathbf{x}_i}$, does not "work" with high probability (w.h.p.), then we can never find positive rate codes of any other form that "work" for the underlying channel.

By setting the *list size* $L-1=1$, results in [43] are recovered by our work.

2 Introduction

While the main contribution of this work is to strictly generalize notions that have been primarily studied for "Hamming metric" channels, before we precisely define general channels, let us reprise what is known for Hamming metric channels in this section.

2.1 Error correction and the Plotkin bound

The theory of error correction codes is about protecting data from errors. In classic coding theory, a code, say $\mathcal{C}$, is just a collection of binary *codewords* (which are usually just binary length-n sequences, where n is called the *blocklength*). The most well-studied error model is *bit-flip*. When a certain codeword is transmitted, an adversary can arbitrarily flip at most np $(0 \leqslant p \leqslant 1)$ bits. It is easy to see that two codewords are not *confusable* if and only if their Hamming distance (number of locations where they differ, denoted $d_{\mathrm{H}}(\cdot,\cdot)$) is at least $2np+1$. Let

$$d_{\min}(\mathcal{C}) := \min_{\substack{\underline{x},\underline{x}'\in\mathcal{C}\\ \underline{x}\neq\underline{x}'}} d_{\mathrm{H}}\left(\underline{x},\underline{x}'\right)$$

denote the minimum pairwise distance of codewords in $\mathcal{C}$. The goal is to *pack* as many codewords as possible in the Hamming space $\mathbb{F}_2^n$ while ensuring that the minimum distance is at least $2np+1$. By a simple volume argument (Gilbert–Varshamov (GV) bound [22, 42]), it is known that *exponentially many* such vectors can be packed when $p < 1/4$. The fundamental quantity that coding theorists are seeking when faced with any communication model is the largest *achievable* rate, i.e., *capacity*. The *rate* $R(\mathcal{C})$ of a code $\mathcal{C}$ is its normalized cardinality, i.e., $R(\mathcal{C}) := \frac{\log|\mathcal{C}|}{n}$. The capacity C measures, asymptotically as the blocklength grows, the largest fraction of bits (out of n) that can be reliably transmitted despite np adversarial bit-flips. C is formally defined as[5]

$$C := \limsup_{n\to\infty} \max_{\mathcal{C}\subset\mathbb{F}_2^n:\, d_{\min}(\mathcal{C})>2np} R(\mathcal{C}).$$

For the aforementioned bit-flip model, as said, the problem of finding the capacity can be also cast as determining the *sphere packing density*. This problem is notoriously difficult and is still open to date. However, we do know that $p = 1/4$ is the threshold below which exponential-sized packing exists (as suggested by the GV bound) and above which it is impossible. The latter fact is the famous Plotkin bound. More formally,

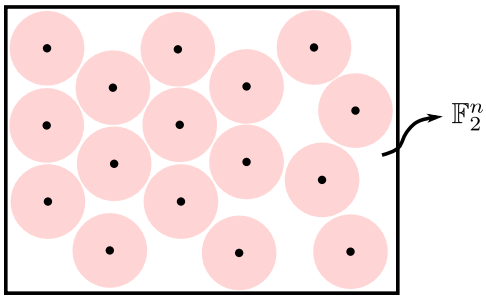
▶ **Theorem 1** (Plotkin bound [33]). *If $p = 1/4 + \epsilon$, $\epsilon > 0$, then any code $\mathcal{C}$ of distance larger than $2np$ has cardinality at most $1 + \frac{1}{4\epsilon}$ (and hence has zero rate).*

We will call the value of p at which the capacity hits zero, the *Plotkin point*. Note that the Plotkin bound actually tells us that above the Plotkin point, not only does every code/packing have a size $2^{o(n)}$ (and hence, rate zero), but also that its size should be at most a *constant* (independent of the blocklength n). Coupled with the achievability result given by the GV bound, the phase transition threshold for exponential-sized packing is thereby identified precisely.

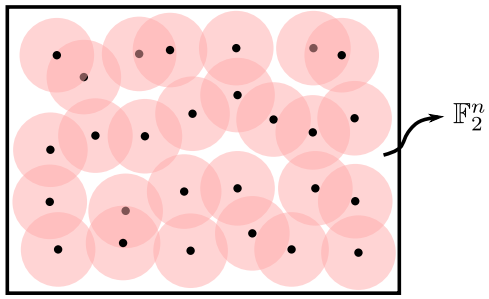
2.2 List decoding and the list decoding Plotkin bound

We now introduce another important notion: *list decoding*. List decodability still requires codewords to be separated out, but in a more relaxed sense; only a few codewords (instead of exactly one under unique decoding discussed earlier) can be captured by a ball of certain radius, no matter where it is located.

[5] It turns out that allowing vanishing probability of decoding error does not change the problem.



(a) An $(L-1)$-packing for $L=2$, i.e., disjoint packing.



(b) An $(L-1)$-packing for $L=3$, i.e., packing with multiplicity 2.

Figure 1 Packing (uniquely decodable codes) vs. multiple packing (list decodable codes). The geometry depicted in the above figures may be misleading compared with the truth in binary Hamming space.

▶ **Definition 2** (List decodability [21, 46]). *A code $\mathcal{C}$ is $(p, L-1)$-list decodable (or $(p, <L)$-list decodable) if for all $\underline{y} \in \mathbb{F}_2^n$, $\left|\mathcal{C} \cap \mathcal{B}_H\left(\underline{y}, np\right)\right| < L$, where $\mathcal{B}_H(\underline{y}, np)$ denotes a Hamming ball centered at $\underline{y}$ of radius np.*

Of course we want the *list size* L to be as small as possible. In particular, the problem is trivial when $L = |\mathcal{C}|$. (The decoder ignores the received word and outputs the whole code.) When $L = 2$, the problem precisely becomes packing. As the admissible L grows, the problem is expected to become easier.

Introduced by Elias [21], list decoding is an important and well-studied subject in coding theory. It is a natural mathematical question to pose towards understanding high-dimensional geometry in discrete spaces. It also serves as a primitive that is useful within and beyond the scope of coding theory. For instance, in many communication problems (e.g., [1, 13]), one proof technique is to let the decoder list decode to a short list (usually $\text{poly}(n)$-sized suffices) of candidate messages, then use other information to disambiguate the list and get a unique message. List decoding also finds applications in complexity theory, cryptography, etc. [25]. For instance, it is used for amplifying hardness and constructing extractors, pseudorandom generators and other pseudorandom objects [20]. The idea of relaxing the problem by asking the solver to just output a list (ideally as small as possible) of solutions that is guaranteed to contain the correct one, instead of insisting on a unique answer, is also adopted in many other fields in computer science [19, 35, 28]. In the context of high-dimensional geometry over finite fields, list decoding is equivalent to multiple packing, just like error correction codes are equivalent to sphere packing. Multiple packing is a natural generalization of the famous sphere packing problem in which, instead of insisting on disjoint alignment, overlap with bounded multiplicity is allowed.

▶ **Definition 3** (Multiple packing). *A subset $\mathcal{C} \subset \mathbb{F}_2^n$ is a $(p, L-1)$-multiple packing if when we put Hamming balls of radii np around each vector in $\mathcal{C}$, no point in the space simultaneously lies in the intersection of at least L balls.*

See Fig. 1 for examples of packing and multiple packing in the Hamming space.

Surprisingly, list decoding capacity is known if we allow L to be asymptotically large. In some sense, list decoding makes us information-theoretic since in many (though not all) cases the list decoding capacity coincides with the capacity of the corresponding Shannon channels in which the noise is random with the same "power" (e.g., in the bit-flip/bit-erasure case, each component of the random noise is independently and identically distributed (i.i.d.) according to a Bernoulli distribution with mean p).

▶ **Theorem 4** (List decoding capacity (folklore) [47]). *Given any $\delta > 0$, there exists an infinite sequence of $(p, \mathcal{O}(1/\delta))$-list decodable codes $\{\mathcal{C}_n\}_n$, each of rate $1 - H(p) - \delta$. Indeed, for any sufficiently large n, a random code (each codeword sampled uniformly at random from $\mathbb{F}_2^n$) of rate $1 - H(p) - \delta$ is $(p, \mathcal{O}(1/\delta))$-list decodable w.h.p..*

On the other hand, any infinite sequence of codes of rate $1 - H(p) + \delta$ is $\left(p, 2^{\Omega(n\delta)}\right)$-list decodable.

*We call $1 - H(p)$ the p-*list decoding capacity *(without specifying a specific L). In particular, the Plotkin point for p-list decoding when L is sufficiently large is $1/2$.*

Though the fundamental limit for the relaxed problem for large constant L is essentially understood, $(p, L-1)$-list decodability for small L (e.g., absolute constant, say $3, 8, 100$, etc.; or sublinear in $1/\delta$, say $(1/\delta)^{1/2}$, $(1/\delta)^{1/3}\log(1/\delta)$, $\log\log(1/\delta)$) is far from being understood. Indeed, it is believed (at least for absolute constant L) to be equivalently hard as the sphere packing problem. Formally, the question of understanding the role of L can be cast as follows. Note first that when $L = 2$, the (unknown) capacity lies somewhere between the Gilbert–Varshamov bound and the Linear Programming bound ([18, 30, 44, 31, 32]). When $L = \mathcal{O}(1/\delta)$, the list decoding capacity $1 - H(p)$ is much larger than the unique decoding capacity. As we increase L, the $(p, L-1)$-list decoding capacity should be gradually "lifted" and the corresponding Plotkin point (the value of p where the capacity is zero) should somehow move rightwards from $1/4$ to $1/2$. The principal goal is to completely understand the dynamics of this evolution.

▶ Remark 5. In this paper, we explicitly distinguish the list decoding capacity for large L and for small L. When we say that L is asymptotically large, we refer to $L = \Omega(1/\delta)$ which suffices to approach the p-list decoding capacity within gap δ. When we say that L is small without further specification, we refer to absolute constant L. For large L, the p-list decoding capacity, denoted by C (recall that we do not explicitly specify L for this regime, see Theorem 4), is fully characterized; however, the $(p, L-1)$-list decoding capacity for small L, denoted by C_{L-1}, is widely open.

Again, for any absolute constant L, the $(p, L-1)$-list decoding capacity is poorly understood. We only have non-matching lower and upper bounds. To our knowledge, the current best bounds are due to Blinovsky from the 80s [9, 10, 11], except for sporadic values of L in some regimes of p. Specifically, for $L = 3$, Ashikhmin–Barg–Litsyn [3] can uniformly improve Blinovsky's upper bound for all values of p. For *even* L's that are at least 4, Polyanskiy [34] can partially beat Blinovsky's upper bounds in the low rate regime.

Though how C_{L-1} approaches C as L increases is not exactly known, Blinovsky's bounds *do* resolve the dynamics of the Plotkin point evolution! Let P_{L-1} denote the Plotkin point for $(p, L-1)$-list decoding. Let $L = 2k$ or $2k+1$ ($k \geqslant 1$). Then Blinovsky's results imply that P_{L-1} is precisely given by the following formula

$$P_{L-1} = \sum_{i=1}^{k} \frac{\binom{2(i-1)}{i-1}}{i} 2^{-2i}.$$

Later, Alon–Bukh–Polyanskiy [2] recovered this result with a simpler-looking formula

$$P_{L-1} = \frac{1}{2} - 2^{-2k-1}\binom{2k}{k},$$

For instance, $P_1 = P_2 = 1/4$, $P_3 = P_4 = 5/16$, etc. As can be noted, the Plotkin point moves *periodically*! The fact that the above two formulas always evaluate to the same value is implicit in [2] and is formally established in Appendix D.

3 Our contributions

Our motivation comes from a well-known connection between list decodability and reliability of communication over adversarial channels. A binary code is $(p, L-1)$-list decodable if and only if it has zero error when used over the following *adversarial bit-flip channel* (Fig. 2).

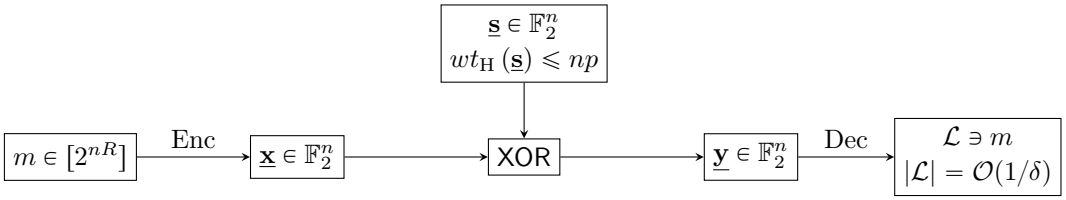


Figure 2 Adversarial bit-flip channels.

The above system depicts a one-way point-to-point communication scenario in which the encoder (Alice) randomly picks a message m from 2^{nR} of them and encodes it into an n-bit string. The adversary (James) stares at this entire codeword and maliciously flips at most np bits of it. Then, the decoder (Bob) receives the corrupted word and is required to output a short list of messages which is guaranteed to contain m with probability 1.

In the above model, the adversary is power constrained in the sense that he only has a budget of np bit-flips. But the encoder is not constrained – she can encode the message into any vector in $\mathbb{F}_2^n$. In some scenarios, codewords are also weight constrained. It makes sense to pose the same question (understanding the list decoding capacity) for input constrained channels. Indeed, this question was also studied in the literature [26].

Motivated by this connection, we significantly generalize the bit-flip model and define list decodability for *general adversarial channels*. We consider a large family of channels in which the encoder is allowed to encode the message into a length-n sequence $\underline{x}$ over *any* alphabet $\mathcal{X}$ of constant size, the adversary is allowed to design an adversarial noise pattern $\underline{s}$ over *any* alphabet $\mathcal{S}$ and the channel can be any *deterministic component-wise* function taking as input a pair of strings from $\mathcal{X}^n \times \mathcal{S}^n$, outputting a sequence $\underline{y}$ over *any* alphabet $\mathcal{Y}$ of the same length. The system designer can incorporate a large family of constraints on $\underline{x}$ and $\underline{s}$ in terms of their *types* (i.e., empirical distributions). The above family of adversarial channels we consider includes but is not limited to

1. The standard adversarial bit-flip channels and adversarial erasure channels;
2. Z-channels in which the adversary can only flip 1 to 0 but not the other way around;
3. Adder channels in which the output is the sum of inputs over the reals rather than modulo the input alphabet size;
4. Channels equipped with Lee distance instead of the Hamming metric.

Indeed, our framework covers most well-studied error models and more that potentially have not been studied in the literature.

However, since we require the channel transition function to act on each component of the inputs independently, a well-studied family of channels is excluded: the *adversarial deletion channels* (cf. [12]). In this model, the adversary can *delete* at most np entries of the transmitted codeword and the decoder receives a vector of smaller length (but at least $(1-p)n$) without knowing the original locations of the symbols he got[6]. Determining the

[6] We want to emphasize the difference between deletions and erasures. When symbols in the codeword are deleted, the rest of the symbols are concatenated and the receiver has no idea which symbols were

Plotkin point for this channel is a long standing open problem. It is known [12] that for binary channels the Plotkin point lies between $\sqrt{2}-1 \approx 0.414$ and 0.5; for q-ary channels, it lies between $1-\frac{2}{q+\sqrt{q}}$ and $1-\frac{1}{q}$. The capacity of this channel is even less understood.

For technical simplicity, we also assume that the channel transition function is *deterministic*, i.e., the output symbol y is a deterministic function of the codeword symbol x and the error symbol s.[7]

We can assume, without any loss of generality, that none of the encoder, decoder and adversary has private randomness to randomize their strategy. This is because there are reductions showing that, given stochastic encoder/decoder, we can construct a deterministic coding scheme with essentially the same rate. Similarly, given a stochastic adversarial error function, we can turn it into a deterministic one which is equivalently malicious in terms of rate. Therefore, for the encoder, it suffices to only consider deterministic codes where each message is mapped to a unique codeword with probability 1. For the adversary, we can assume the error pattern is a deterministic function of the transmitted codeword. Nevertheless, note that the error function does *not* have to be component-wise independent. The i-th component $\underline{s}(i)$ of the noise pattern $\underline{s}$ can depend on *every* entry of $\underline{x}$, not only on the corresponding $\underline{x}(i)$. Moreover, the decoder's decision on the estimated message given the received word can also be assumed to be deterministic. That is, we can require the decoder to output the correct message with *zero* error probability. Hence, the problem is purely combinatorial and all desirable events should happen with probability one.

In this work, we precisely *characterize* the Plotkin point for list decoding over any channel from the above family of general adversarial channels. That is, we essentially provide a criterion (sufficient and necessary condition) for the existence of positive rate $(L-1)$-list decodable codes for such channels.

In the context of high-dimensional geometry over finite fields, the result can be also cast as pinning down the location of the phase transition threshold for the optimal density of $(L-1)$-multiple packing using *general shapes* (not necessarily Hamming balls) corresponding to the defining constraints on codewords and errors of the channel. Above the threshold, exponential-sized multiple packing exists while below that, it is impossible to have such exponential-sized multiple packings.

This criterion can be summarized in one sentence:

> exponential-sized $(L-1)$-list decodable codes for general adversarial channels (or $(L-1)$-multiple packings using general shapes) essentially exist if and only if the *completely positive tensor cone* of order-L is not entirely contained in the *confusability set* of the channel for $(L-1)$-list decoding.

Jargon in the above informal statement will become understandable once we formalize the problem setup and present rigorous claims. The proof consists of the sufficiency part and the necessity part. At a very high level, the sufficiency part follows from a random coding argument and its generalization inspired by the time-sharing argument frequently used in Network Information Theory. The necessity part builds upon and significantly generalizes the classic Plotkin bound, which goes by first extracting an *equicoupled* subcode using Ramsey theory and then applying a generalized *double counting* trick.

---

deleted. However, when symbols are erased, they are replaced by erasure symbols erasure at the same locations and the receiver seeing them knows exactly which symbols were erased. Hence the erasure case is much simpler than the deletion case.

[7] The general case in which the channel law is given by a conditional distribution $W_{\mathbf{y}|\mathbf{x},\mathbf{s}}$ (with not necessarily only singleton atoms) is more technical and is left as one of our future directions.

Our other results include the following:

1. For any given general adversarial channel, we pin down the list decoding capacity for asymptotically large L. This generalizes the classic list decoding capacity in the bit-flip case. The lower bound is achieved by a purely random code. The upper bound follows from a volume packing argument.
2. For any given general adversarial channel, we determine the *exact* order (in terms of δ) of the list sizes of a large fraction (exponentially close to one) of constant composition codes (in which all codewords have the same type) achieving the list decoding capacity within gap δ. It turns out that if we pick a constant composition code from the set of all such codes uniformly at random, with high probability, it is exactly $\Theta(1/\delta)$-list decodable.
3. For any given general adversarial channel and any $L \geqslant 2$, we give a lower bound on the $(L-1)$-list decoding capacity. It coincides with the generalized Gilbert–Varshamov bound obtained by [43] when $L-1$ is equal to 1. Our bound follows from a random code construction assisted by expurgation, generalizing a classic construction for $(p, L-1)$-list decoding in the bit-flip case [24]. Note that this construction differs from [43]'s construction for unique decoding using greedy packing.
4. In the special case where $L = 2$, i.e., the unique decoding setting and under the bit-flip model, we evaluate the Gilbert–Varshamov-type bound and an achievable rate expression of cloud codes (codes constructed from CP distributions) obtained by [43]. In particular, we show that the Gilbert–Varshamov-type bound for general adversarial channels matches the classic GV bound in coding theory. We also provide an explicit convex program for evaluating achievable rates of cloud codes.
5. By evaluating our general criterion under the bit-flip model, we *numerically* recover Blinovsky's [9] characterization of the Plotkin points for $(p, L-1)$-list decoding. This boils down to checking the feasibility of an explicit linear program with structured coefficient matrix. Though the LP has size exponential in L, its feasibility can be checked in constant time since our results are tailored for constant L independent of the blocklength n (which needs to approach infinity for many of our results to hold).
6. By utilizing facts discovered in this paper, we *rigorously* recover Blinovsky's [9] characterization of the Plotkin points for $(p, L-1)$-list decoding. Our proof avoids the complicated calculations Blinovsky did and demystifies the formula by Blinovsky[8]. In particular, our lower bound on the Plotkin point explains why, in the low rate regime, *average-radius*[9] list decoding is equivalent to the classic notion of list decoding. We believe that this fact was first observed and rigorously justified by Blinovsky. It was later rediscovered many times and became one of the basic starting points of many papers, especially those regarding list decoding random q-ary linear codes. Our upper bound relates the Plotkin point P_{L-1} to the expected translation distance of a one-dimensional unbiased random walk after L steps. In summary, using connections between codes and random variables, we are able to re-interpret the results by Blinvosky [9] and Alon–Bukh–Polyanskiy [2] within the framework we established by providing a more intuitive formula which matches known results.

[8] In fact, he provided upper and lower bounds on the $(p, L-1)$-list decoding capacity which happen to vanish at the same value of p.

[9] $(p, L-1)$-average-radius list decodability requires that the *average* distance (instead of maximum distance required by the classic notion of $(p, L-1)$-list decodability) from any L-tuple of codewords to their centroid is larger than np. Average-radius list decodability is a more stringent requirement since it implies the classic list-decodability. However, it is easier to analyze since the problem is *linearized* from infinity norm to one norm. Indeed it plays a useful role in a long line of work towards understanding the list decodability of random linear codes [26, 45, 36, 37, 38].

4 Overview of techniques

Our paper is highly correlated to a sister paper [43] which a subset of the authors are involved in. That paper provides generalized Plotkin bound for *unique* decoding over general adversarial channels. The authors showed that exponential-sized *uniquely* decodable codes or hard packings exist if and only if the set of completely positive *matrices* is not entirely contained in the *confusability set* associated to the given channel. This answers the question we posed in the beginning of the paper for the $L = 2$ case. We generalize their results to *any universal constant* L. Almost all results in [43] can be recovered by setting $L = 2$ in our paper.

We give an overview of the techniques used in this paper and highlight the similarities and differences between [43][10] and our work.

1. The general adversarial channel models that both papers are concerned with belong to a larger family of channels known as *Arbitrarily Varying Channels (AVC)* in Information Theory community; these were first studied by Blackwell et al. [8] (see [29] for a detailed survey). We want to emphasize that the bulk of the literature on AVCs deals with *oblivious* adversary channels in which the adversary has to pick his malicious noise pattern *before* the codeword is chosen from the codebook (and hence, *oblivious* of the transmitted codeword) by the encoder. This makes the problem significantly easier and the capacity of such channels is precisely known (cf. [17]). The channels that [43] and we are considering are such that the adversary gets to design the error pattern with the complete knowledge of the transmitted codeword; these are called *omniscient* adversaries in [43]. This problem is much more difficult and the capacity is, again, widely open even for simple models such as the bit-flip channels. Indeed, the subclass of AVCs that [43] and we defined is motivated by the bit-flip channels and its various variants, e.g., q-ary channels, weight constrained channels, asymmetric channels, etc..
2. The connection between codes and random variables/distributions is classic in Theoretical Computer Science. The idea of realizing binary error correction codes using $\{-1, 1\}$-valued random variables or functions supported on the Boolean hypercube $\{-1, 1\}^n$ is spread out in the literature explicitly or in disguise. Such a trick allows one to borrow tools from other fields of Theoretical Computer Science, e.g., the theory of expander graphs, randomness extractors, small-bias distributions, discrete Fourier analysis, etc., (cf. [40, 5, 41, 7]) to understand, construct and analyze codes.
3. With respect to (w.r.t.) codes for general adversarial channels, the specific idea of collecting admissible types of good codes and studying the set of corresponding distributions was used in [43]. In particular, they defined similar notions of self-couplings and confusability sets which are submanifolds of *matrices* corresponding to joint distributions. Such objects only take care of *pairwise* interaction of codewords, which is insufficient for understanding list decoding. We generalize their notions to *tensors* which capture the (empirical) joint distributions of *lists* of codewords. While some properties in [43] continue to hold when objects in the matrix case are extended to their tensor versions, others fail to hold, as we will see in the rest of the paper. We also encounter issues which do not arise in the unique decoding setting. As is well-known, tensors are much more delicate [27] to handle compared to matrices.

[10] Though the work by Wang–Budkuley–Bogdanov–Jaggi [43] has been accepted to ISIT 2019, the conference version is limited to 5 pages and contains essentially no proof. At the time this paper is written, we do not have a publicly available *full* version of [43] and the following comparison is w.r.t. the current status of a draft of [43] that the authors kindly shared with us.

4. To prove *upper* bounds on capacity, it is also an old idea to extract structured subcodes from *any* infinite sequence of good codes. Depending on the applications, the nature of *structures* and techniques used to extract them may vary. To the best of our knowledge, in coding theory, the use of Ramsey theory for obtaining symmetric subcodes dates back to as least as early as Blinovsky [9]. His techniques were applied in a similar manner in followup works by Polyanskiy [34] and Alon–Bukh–Polyanskiy [2]. The work in [43] generalized this idea and managed to extract structured subcodes from arbitrary codes for *general* adversarial channels. Since they dealt with unique decoding, *pairwise* equicoupledness suffices. In our setup, we would like a sequence of subcodes which are *L-wise equicoupled* in the sense that the (empirical) joint distribution of any L-tuple of codewords from the extracted subcode is approximately the same and is close to some $\widehat{P}_{\mathbf{x}_1,\cdots,\mathbf{x}_L}$. This resembles but generalizes Polyanskiy's [34] techniques. One of the downsides of invoking Ramsey theory is that the reduction usually causes terrible detriment to the rate of the code, since the smallest size for a combinatorial object to contain abundant structures is generally poorly understood in combinatorics. However, we are fine to tolerate such a rate loss since we only care about the *positivity* of the $(L-1)$-list decoding capacity.
5. To show *lower* bounds on capacity, we use the random coding argument aided by *expurgation*. In the prior work [43], the achievability result is obtained by greedy packing. This is reminiscent of a classic technique in coding theory for proving the existence of good codes of certain size. Since in the unique decoding (hard packing) setting, goodness of a code relies merely on pairwise statistics, the size of a greedy packing can be lower bounded using a standard volume counting argument. Indeed, this idea can be implemented in the general setting by counting the volume of the "forbidden region" of any codeword [43]. However, in list decoding setting, the notion of *confusability* is defined for *tuples* of codewords and translates to bounded multiplicity of intersection of forbidden regions of codewords. It is thus not clear how to pack codewords in a greedy manner while ensuring non-existence of local dense clusters. Instead, our code construction is more information-theoretic. We apply ideas of random coding with expurgation which is commonly used in the study of error exponent in Information Theory. A random code may be mildly locally clustered, but this only occurs at rare locations in the space of all length-n input sequences. Indeed, we are able to show that, with high probability, a random code carefully massaged by shoveling off a small number of codewords attains a GV-type bound for general channels.
6. The most difficult part of our work is the converse.
   a. First assume that the distribution $\widehat{P}_{\mathbf{x}_1,\cdots,\mathbf{x}_L}$ associated to the subcode obtained by Ramsey reduction is *symmetric*. To show that no large $(L-1)$-list decodable code exists for general adversarial channels when $\widehat{P}_{\mathbf{x}_1,\cdots,\mathbf{x}_L}$ is not completely positive, we provide upper and lower bounds on the average (over all L-tuples in the equicoupled subcode) inner product between the empirical distribution of an L-tuple and a copositive witness of non-complete positivity of $\widehat{P}_{\mathbf{x}_1,\cdots,\mathbf{x}_L}$. The bounds contradict each other if the code size exceeds certain constant (independent of the blocklength). We review this *double counting trick* (for unique and list decoding under special settings that appeared in prior work) in Section 5. The $L = 2$ case is proved in [43]. The existence of the witness of non-complete positivity is guaranteed by the duality of certain matrix cones. We generalize calculations in [43] to joint distributions of > 2 random variables. Similar notions of complete positivity and copositivity for tensors exist in the literature and the duality continues to hold.

b. If $\widehat{P}_{\mathbf{x}_1,\cdots,\mathbf{x}_L}$ is *asymmetric*, we use a completely different argument. We reduce the claim, in a nontrivial way, to the $L = 2$ case which is known to be true [43]. The $L = 2$ case itself is proved [43] by viewing the task of constructing a long sequence of random variables with prescribed asymmetric pairwise marginals as a zero sum game and using discrete Fourier analysis to provide conflicting bounds on the value of the game, if the sequence is longer than certain constant (again independent of the blocklength).

5 Prior work

Among various ideas, our results are built upon prior work which applies a *double counting trick* to obtain upper bounds on code sizes. We first review this technique which can be found in the proof of the classic Plotkin bound and its generalizations.

5.1 Plotkin [33]

One way to prove Theorem 1 is by lower and upper bounding the expected pairwise distance of any given code $\mathcal{C}$ with minimum distance larger than $2np$ ($p = 1/4 + \epsilon$)

$$\mathop{\mathbb{E}}_{(\underline{x},\underline{x}')\sim\mathcal{C}\times\mathcal{C}}\left[d_{\mathrm{H}}\left(\underline{x},\underline{x}'\right)\right], \tag{2}$$

where $\underline{x}, \underline{x}'$ are uniformly and independently picked from $\mathcal{C}$. First note that pairs $\underline{x} = \underline{x}'$ do not contribute to the expectation. On the one hand, the expectation is clearly at least

$$\frac{|\mathcal{C}|(|\mathcal{C}|-1)}{|\mathcal{C}|^2} d_{\min} > |\mathcal{C}|^{-1}(|\mathcal{C}|-1)2np = |\mathcal{C}|^{-1}(|\mathcal{C}|-1)2n(1/4+\epsilon).$$

On the other hand, if we stack codewords into a $2^{nR} \times n$ matrix and let S_j denote the number of 1's in the j-th column, then from the column's perspective, the above expectation is at most

$$\frac{1}{|\mathcal{C}|^2}\sum_{j=1}^{n} 2S_j(|\mathcal{C}| - S_j).$$

The coefficient 2 is because we need to count $(\underline{x}, \underline{x}')$ and $(\underline{x}', \underline{x})$ separately. This bound is at most $n/2$ by concavity of the summands in S_j. Comparing the upper and lower bounds we have that $|\mathcal{C}| \leqslant 1 + \frac{1}{4\epsilon}$, as claimed in Theorem 1.

5.2 Blinovsky [9]

The above double counting argument can be generalized to the setting of list decoding. For the $(p, L-1)$-list decoding setup we introduced in Definition 2, the earliest work we are aware of following this idea is the one by Blinovsky [9].

Unlike Theorem 1, not only did Blinovsky show that any $(p, L-1)$-list decodable code has to be small as long as $p > P_{L-1}$, he even gave an upper bound (it is still essentially the best as far as we know) on the $(p, L-1)$-list decoding capacity for *any* L. We sketch his idea below but omit the complicated calculations.

First note that proving upper bounds on C_{L-1} for fixed p is equivalent to proving upper bounds on p for fixed rate R. We then define the following three quantities

$$r_{\mathrm{LD}} = \min_{\mathcal{L}\in\binom{\mathcal{C}}{L}} \min_{\underline{y}\in\mathbb{F}_2^n} \max_{\underline{x}\in\mathcal{L}} d_{\mathrm{H}}\left(\underline{y},\underline{x}\right), \tag{3}$$

$$r_{\mathrm{avg}} = \min_{\mathcal{L}\in\binom{\mathcal{C}}{L}} \min_{\underline{y}\in\mathbb{F}_2^n} \mathop{\mathbb{E}}_{\underline{x}\sim\mathcal{L}} \left[d_{\mathrm{H}}\left(\underline{y},\underline{x}\right)\right], \tag{4}$$

$$r_{\mathrm{DC}} = \mathop{\mathbb{E}}_{\mathcal{L}\sim\binom{\mathcal{C}}{L}} \min_{\underline{y}\in\mathbb{F}_2^n} \mathop{\mathbb{E}}_{\underline{x}\sim\mathcal{L}} \left[d_{\mathrm{H}}\left(\underline{y},\underline{x}\right)\right]. \tag{5}$$

All expectations are over uniform selections from the corresponding sets. Namely,

$$\mathop{\mathbb{E}}_{\mathcal{L}\sim\binom{\mathcal{C}}{L}}[\cdot] = \frac{1}{\binom{|\mathcal{C}|}{L}} \sum_{\mathcal{L}\in\binom{\mathcal{C}}{L}} [\cdot], \quad \mathop{\mathbb{E}}_{\underline{x}\sim\mathcal{L}}[\cdot] = \frac{1}{L}\sum_{\underline{x}\in\mathcal{L}}[\cdot].$$

Let us parse what these quantities are measuring.

1. r_{LD} is known as the *list decoding radius* of a given code $\mathcal{C}$. The minimax expression associated to a set $\mathcal{L}$ of vectors

   $$r_{\mathrm{Cheb}} \coloneqq \min_{\underline{y}\in\mathbb{F}_2^n} \max_{\underline{x}\in\mathcal{L}} d_{\mathrm{H}}\left(\underline{y},\underline{x}\right)$$

   is known as the *Chebyshev radius* of $\mathcal{L}$. It is the radius of the smallest circumscribed ball of $\mathcal{L}$. And

   $$p^*(R) \coloneqq \limsup_{n\to\infty} \max_{\mathcal{C}\subset\mathbb{F}_2^n\colon |\mathcal{C}|\geqslant 2^{nR}} r_{\mathrm{LD}}(\mathcal{C})$$

   is precisely the largest allowable p for $(p, L-1)$-list decodable codes of a fixed rate R to exist. Note that $p^*(0) = P_{L-1}$.
2. r_{avg} is known as the *average list decoding radius* and the min-average expression

   $$\min_{\underline{y}\in\mathbb{F}_2^n} \mathop{\mathbb{E}}_{\underline{x}\sim\mathcal{L}} \left[d_{\mathrm{H}}\left(\underline{y},\underline{x}\right)\right]$$

   is the *average radius* of a list. It is not hard to see that the average radius center of $\mathcal{L}$ is the component-wise majority of vectors in $\mathcal{L}$, i.e., the minimizer $\underline{y}^*$ has $\mathsf{MAJ}\left(\underline{x}(i)\colon \underline{x}\in\mathcal{L}\right)$ as its i-th component. Define *plurality* as

   $$\begin{array}{rccl} \mathsf{PLUR}\colon & \mathbb{F}_2^L & \to & [0,1] \\ & (x_1,\cdots,x_L) & \mapsto & \frac{1}{L}\left|\{i\in[L]\colon x_i = \mathsf{MAJ}(x_1,\cdots,x_L)\}\right|, \end{array}$$

   which is the fraction of the most frequent symbol. Then the average radius of $\mathcal{L}$ can be explicitly written as

   $$\min_{\underline{y}\in\mathbb{F}_2^n} \mathop{\mathbb{E}}_{\underline{x}\sim\mathcal{L}} \left[d_{\mathrm{H}}\left(\underline{y},\underline{x}\right)\right] = \sum_{j=1}^{n} \left(1 - \mathsf{PLUR}\left(\underline{x}(j)\colon \underline{x}\in\mathcal{L}\right)\right).$$

3. r_{DC} is a further variant of r_{LD} – the ultimate quantity we are looking for. It is the object that Blinovsky was really dealing with. Note that it is in the same spirit as the quantity (2) considered in the double counting argument in the proof of the classic Plotkin bound. Blinovsky used r_{DC} as a proxy to finally bound r_{LD}.

By extracting a constant weight subcode and applying the double counting trick (and using the convexity of a certain function), Blinovsky showed the following

▶ **Lemma 6.** *Let $\lambda\in[0,1/2]$ and fix $R = 1 - H(\lambda)$. Then*

$$r_{\mathrm{DC}} \leqslant \sum_{i=1}^{\lceil L/2\rceil} \frac{\binom{2i-2}{i-1}}{i} (\lambda(1-\lambda))^i.$$

Apparently, by definition, we have

$$r_{\mathrm{LD}} \geqslant r_{\mathrm{avg}}, \quad r_{\mathrm{DC}} \geqslant r_{\mathrm{avg}}.$$

So Lemma 6 automatically holds for r_{avg}. However, a priori the relation between r_{LD} and r_{DC} is unclear. Surprisingly, Blinovsky showed that it is "okay" to replace the first and third optimization in r_{LD} with averaging in the sense that the following holds.

▶ **Lemma 7.** *For any infinite sequence of codes $\{\mathcal{C}_n\}_n$, there exists an infinite sequence of subcodes $\mathcal{C}'_n \subseteq \mathcal{C}_n$ such that $r_{\mathrm{LD}}(\mathcal{C}') = r_{\mathrm{avg}}(\mathcal{C}') + o(n)$.*

The proof involves an *equidistant* subcode extraction step using Ramsey theory. Lemma 7 implies that the same bound in Lemma 6 holds for r_{LD} as well!

5.3 Cohen–Litsyn–Zémor [15]

Similar ideas were used to provide upper bounds on the erasure list decoding capacity. A binary code is said to be $(p, L-1)$-*erasure list decodable* if for any $\mathcal{T} \in \binom{[n]}{n(1-p)}$ and any $\underline{y} \in \mathbb{F}_2^{(1-p)n}$, $\left|\{\underline{x} \in \mathcal{C} \colon \underline{x}|_{\mathcal{T}} = \underline{y}\}\right| \leqslant L-1$, where $\underline{x}|_{\mathcal{T}}$ denotes the restriction of $\underline{x}$ to $\mathcal{T}$, i.e., a vector of length $|\mathcal{T}|$ only consisting of components from $\underline{x}$ indexed by elements in $\mathcal{T}$. The erasure list decoding radius $r_{\mathrm{LD,eras}}$ and the $(p, L-1)$-erasure list decoding capacity $C_{L-1,\mathrm{eras}}$ are defined in the same manner. Cohen–Litsyn–Zémor [15] showed that

▶ **Theorem 8** ([15])**.** *$C_{L,\mathrm{eras}} \leqslant 1 - H(\lambda)$, where λ is the unique root of the equation $\lambda^{L+1} + (1-\lambda)^{L+1} = 1 - p$ in $[0, 1/2]$.*

The idea is essentially still double counting. Here, it turns out that the right object to be counted is the *erasure radius* of a list $\mathcal{L}$,

$$r_{\mathrm{eras}} := \left|\{i \in [n] \colon \underline{x}(i) \text{ are the same } \forall \underline{x} \in \mathcal{L}\}\right|.$$

Extracting a subcode living on a sphere (followed by shifting out the center to get a constant weight subcode $\mathcal{C}'$) and conducting similar calculations on

$$\mathop{\mathbb{E}}_{\mathcal{L} \sim \binom{\mathcal{C}'}{L}} \left[r_{\mathrm{eras}}(\mathcal{L})\right],$$

allow the authors to conclude Theorem 8.

▶ Remark 9. The original paper [15] was stated for *generalized distance* which is equivalent to erasure list decoding radius via a well-known connection. The above version was presented in Guruswami's PhD thesis [24].

5.4 Wang–Budkuley–Bogdanov–Jaggi [43]

As mentioned, our work is a continuation of the prior work [43] which a subset of the authors were involved in. We refer the readers to the corresponding paragraphs in Sec. 1 and Sec. 3 for a review of their work along with a comparison with this work.

6 Organization of the paper

In Sec. 1 we have seen numeric examples that illustrate our results. In Sec. 2 we properly motivated the problem and introduced relevant background in coding theory. Our contributions in this paper were listed in details in Sec. 3. In Sec. 4 we reviewed various techniques used in this paper and highlighted our innovations. Prior works that our results build up on and push forward were surveyed in Sec. 5.

The rest of the paper is organized as follows. We fix our notational conventions in Sec. 7 and provide necessary preliminaries, especially *the method of types* in Information Theory, in Sec. 8. We develop basic notions that will be used throughout the paper in Sec. 9. In particular, *general adversarial channels* and objects associated to them will be introduced in this section. In Sec. 10 we prove the list decoding capacity theorem for general adversarial channels when L is asymptotically large. Furthermore, we obtain *tight* list size bounds for *most* capacity-achieving constant composition codes in Sec. 11. In Sec. 12 and Sec. 13 we show sufficiency and necessity, respectively, of the criterion for the existence of exponential-sized $(L-1)$-list decodable codes (where L is an arbitrary universal constant) for general adversarial channels. In Sec. 14 we make two remarks on the converse, which is technically the most challenging piece of our work. In Sec. 15 we verify the correctness of our characterization obtained in Sec. 12 and Sec. 13 by running it on the problem specialized to the bit-flip model which has been understood in prior works [9, 2]. In Sec. 16, utilizing tools developed and facts proved in this paper, we rigorously rederive Blinovsky's [9] results. We obtain more intuitive expressions and demystify his calculations. In Sec. 17 we evaluate bounds on the unique decoding capacity ($L = 2$) in [43] under the bit-flip model. We conclude the paper and list several open questions and future directions in Sec. 18. Some calculations and background knowledge are deferred to Appendices A, B, C and D.

7 Notation

Conventions. Sets are denoted by capital letters in calligraphic typeface, e.g., $\mathcal{C}, \mathcal{I}$, etc.. Random variables are denoted by lower case letters in boldface or capital letters in plain typeface, e.g., $\mathbf{m}, \mathbf{x}, \mathbf{s}, U, W$, etc.. Their realizations are denoted by corresponding lower case letters in plain typeface, e.g., m, x, s, u, w, etc.. Vectors (random or fixed) of length n, where n is the blocklength without further specification, are denoted by lower case letters with underlines, e.g., $\underline{\mathbf{x}}, \underline{\mathbf{s}}, \underline{x}, \underline{s}$, etc.. The i-th entry of a vector $\underline{x} \in \mathcal{X}^n$ is denoted by $\underline{x}(i)$ since we can alternatively think of $\underline{x}$ as a function from $[n]$ to $\mathcal{X}$. Same for a random vector $\underline{\mathbf{x}}$. Matrices are denoted by capital letters in boldface, e.g., $\mathbf{P}, \mathbf{\Sigma}$, etc.. Similarly, the (i,j)-th entry of a matrix $\mathbf{G} \in \mathbb{F}^{n\times m}$ is denoted by $\mathbf{G}(i,j)$. We sometimes write $\mathbf{G}_{n\times m}$ to explicitly specify its dimension. For square matrices, we write $\mathbf{G}_n$ for short. Letter $\mathbf{I}$ is reserved for identity matrix. Tensors are denoted by capital letters in plain typeface, e.g., T, P, etc..

Functions. We use the standard Bachmann–Landau (Big-Oh) notation for asymptotics of real-valued functions in positive integers.

For $x \in \mathbb{R}$, let $[x]^+ \coloneqq \max\{x, 0\}$.

For two real-valued functions f, g on the same domain Ω, let fg and f/g denote the functions obtained by multiplying and taking the ratio of the images of f and g point-wise, respectively. That is, for $\omega \in \Omega$,

$$(fg)(\omega) \coloneqq f(\omega)g(\omega), \quad (f/g)(\omega) \coloneqq f(\omega)/g(\omega).$$

In particular, for types or distributions, we can write $\tau_{\mathbf{x},\mathbf{y}} = \tau_{\mathbf{x}}\tau_{\mathbf{y}|\mathbf{x}}, \tau_{\mathbf{y}|\mathbf{x}} = \tau_{\mathbf{x},\mathbf{y}}/\tau_{\mathbf{x}}$, or $P_{\mathbf{x},\mathbf{y}} = P_{\mathbf{x}}P_{\mathbf{y}|\mathbf{x}}, P_{\mathbf{y}|\mathbf{x}} = P_{\mathbf{x},\mathbf{y}}/P_{\mathbf{x}}$ and so on.

For two real-valued functions $f(n), g(n)$ in positive integers, we say that $f(n)$ *asymptotically equals* $g(n)$, denoted $f(n) \asymp g(n)$, if

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 1.$$

For instance, $2^{n+\log n} \asymp 2^{n+\log n} + 2^n$, $2^{n+\log n} \not\asymp 2^n$. We write $f(n) \doteq g(n)$ (read $f(n)$ dot equals $g(n)$) if the coefficients of the dominant terms in the exponents of $f(n)$ and $g(n)$ match,

$$\lim_{n\to\infty} \frac{\log f(n)}{\log g(n)} = 1.$$

For instance, $2^{3n} \doteq 2^{3n+n^{1/4}}$, $2^{2^n} \neq 2^{2^{n+\log n}}$. Note that $f(n) \asymp g(n)$ implies $f(n) \doteq g(n)$, but the converse is not true.

For any $q \in \mathbb{R}_{>0}$, we write $\log_q(\cdot)$ for the logarithm to the base q. In particular, let $\log(\cdot)$ and $\ln(\cdot)$ denote logarithms to the base two and e, respectively.

Sets. For any two sets $\mathcal{A}$ and $\mathcal{B}$ with additive and multiplicative structures, let $\mathcal{A}+\mathcal{B}$ and $\mathcal{A}\cdot\mathcal{B}$ denote the Minkowski sum and Minkowski product of them which are defined as

$$\mathcal{A}+\mathcal{B} \coloneqq \{a+b\colon a\in\mathcal{A}, b\in\mathcal{B}\}, \quad \mathcal{A}\cdot\mathcal{B} \coloneqq \{a\cdot b\colon a\in\mathcal{A}, b\in\mathcal{B}\},$$

respectively. If $\mathcal{A}=\{x\}$ is a singleton set, we write $x+\mathcal{B}$ and $x\cdot\mathcal{B}$ for $\{x\}+\mathcal{B}$ and $\{x\}\cdot\mathcal{B}$.

For any finite set $\mathcal{X}$ and any integer $0 \leqslant k \leqslant |\mathcal{X}|$, we use $\binom{\mathcal{X}}{k}$ to denote the collection of all subsets of $\mathcal{X}$ of size k.

$$\binom{\mathcal{X}}{k} \coloneqq \{\mathcal{Y}\subseteq\mathcal{X}\colon\ |\mathcal{Y}|=k\}.$$

For $M\in\mathbb{Z}_{>0}$, we let $[M]$ denote the set of first M positive integers $\{1,2,\cdots,M\}$.

For any $\mathcal{A}\subseteq\Omega$, the indicator function of $\mathcal{A}$ is defined as, for any $x\in\Omega$,

$$\mathbb{1}_{\mathcal{A}}(x) \coloneqq \begin{cases} 1, & x\in\mathcal{A} \\ 0, & x\notin\mathcal{A} \end{cases}.$$

At times, we will slightly abuse notation by saying that $\mathbb{1}_{\mathsf{A}}$ is 1 when event A happens and 0 otherwise. Note that $\mathbb{1}_{\mathcal{A}}(\cdot) = \mathbb{1}_{\{\cdot\in\mathcal{A}\}}$.

Geometry. For any $\underline{x}\in\mathbb{F}_q^n$, let $wt_{\mathrm{H}}(\underline{x})$ denote the Hamming weight of $\underline{x}$, i.e., the number of nonzero entries of $\underline{x}$.

$$wt_{\mathrm{H}}(\underline{x}) \coloneqq |\{i\in[n]\colon \underline{x}(i)\neq 0\}|.$$

For any $\underline{x},\underline{y}\in\mathbb{F}_q^n$, let $d_{\mathrm{H}}(\underline{x},\underline{y})$ denote the Hamming distance between $\underline{x}$ and $\underline{y}$, i.e., the number of locations where they differ.

$$d_{\mathrm{H}}(\underline{x},\underline{y}) \coloneqq wt_{\mathrm{H}}(\underline{x}-\underline{y}) = \left|\left\{i\in[n]\colon \underline{x}(i)\neq\underline{y}(i)\right\}\right|.$$

Balls and spheres in $\mathbb{F}_q^n$ centered around some point $\underline{x}\in\mathbb{F}_q^n$ of certain radius $r\in\{0,1,\cdots,n\}$ w.r.t. the Hamming metric are defined as follows.

$$\mathcal{B}_{\mathrm{H}}^n(\underline{x},r) \coloneqq \left\{\underline{y}\in\mathbb{F}_q^n\colon d_{\mathrm{H}}(\underline{x},\underline{y})\leqslant r\right\}, \quad \mathcal{S}_{\mathrm{H}}^n(\underline{x},r) \coloneqq \left\{\underline{y}\in\mathbb{F}_q^n\colon d_{\mathrm{H}}(\underline{x},\underline{y})= r\right\}.$$

We will drop the subscript and superscript for the associated metric and dimension when they are clear from the context.

Probability. The probability mass function (p.m.f.) of a discrete random variable $\mathbf{x}$ or a random vector $\underline{\mathbf{x}}$ is denoted by $P_{\mathbf{x}}$ or $P_{\underline{\mathbf{x}}}$. Here we use the following shorthand notation to denote the probability that $\mathbf{x}$ or $\underline{\mathbf{x}}$ distributed according to $P_{\mathbf{x}}$ or $P_{\underline{\mathbf{x}}}$ takes a particular value.

$$P_{\mathbf{x}}(x) \coloneqq \Pr_{\mathbf{x}\sim P_{\mathbf{x}}}[\mathbf{x} = x], \quad P_{\underline{\mathbf{x}}}(\underline{x}) = \Pr_{\underline{\mathbf{x}}\sim P_{\underline{\mathbf{x}}}}[\underline{\mathbf{x}} = \underline{x}],$$

for any $x \in \mathcal{X}$ or $\underline{x} \in \mathcal{X}^n$. If every entry of $\underline{\mathbf{x}}$ is independently and identically distributed (i.i.d.) according to $P_{\mathbf{x}}$, then we write $\underline{\mathbf{x}} \sim P_{\mathbf{x}}^{\otimes n}$, where $P_{\mathbf{x}}^{\otimes n}$ is a product distribution defined as

$$P_{\underline{\mathbf{x}}}(\underline{x}) = P_{\mathbf{x}}^{\otimes n}(\underline{x}) \coloneqq \prod_{i=1}^{n} P_{\mathbf{x}}(\underline{x}(i)).$$

For a finite set $\mathcal{X}$, $\Delta(\mathcal{X})$ denotes the probability simplex on $\mathcal{X}$, i.e., the set of all probability distributions supported on $\mathcal{X}$,

$$\Delta(\mathcal{X}) \coloneqq \left\{ P_{\mathbf{x}} \in [0,1]^{|\mathcal{X}|} \colon \sum_{x\in\mathcal{X}} P_{\mathbf{x}}(x) = 1 \right\}.$$

Similarly, $\Delta(\mathcal{X}\times\mathcal{Y})$ denotes the probability simplex on $\mathcal{X}\times\mathcal{Y}$,

$$\Delta(\mathcal{X}\times\mathcal{Y}) \coloneqq \left\{ P_{\mathbf{x},\mathbf{y}} \in [0,1]^{|\mathcal{X}|\times|\mathcal{Y}|} \colon \sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} P_{\mathbf{x},\mathbf{y}}(x,y) = 1 \right\}.$$

Let $\Delta(\mathcal{Y}|\mathcal{X})$ denote the set of all conditional distributions,

$$\Delta(\mathcal{Y}|\mathcal{X}) \coloneqq \left\{ P_{\mathbf{y}|\mathbf{x}} \in \mathbb{R}^{|\mathcal{X}|\times|\mathcal{Y}|} \colon P_{\mathbf{y}|\mathbf{x}}(\cdot|x) \in \Delta(\mathcal{Y}),\ \forall x \in \mathcal{X} \right\}.$$

The general notion for multiple spaces is defined in the same manner.

Let $\mathrm{Unif}(\Omega)$ denote the uniform distribution on some probability space Ω.

For a joint distribution $P_{\mathbf{x},\mathbf{y}} \in \Delta(\mathcal{X}\times\mathcal{Y})$, let $[P_{\mathbf{x},\mathbf{y}}]_{\mathbf{x}} \in \Delta(\mathcal{X})$ denote the *marginalization* onto the variable $\mathbf{x}$, i.e., for $x \in \mathcal{X}$,

$$[P_{\mathbf{x},\mathbf{y}}]_{\mathbf{x}}(x) \coloneqq \sum_{y\in\mathcal{Y}} P_{\mathbf{x},\mathbf{y}}(x,y).$$

Sometimes we simply write it as $P_{\mathbf{x}}$ when the notation is not overloaded.

Algebra. Let $\|\cdot\|_p$ denote the standard ℓ^p-norm. Specifically, for any $\underline{x} \in \mathbb{R}^n$,

$$\|\underline{x}\|_p \coloneqq \left(\sum_{i=1}^{n} |\underline{x}(i)|^p \right)^{1/p}.$$

For brevity, we also write $\|\cdot\|$ for the ℓ^2-norm.

An order-k dimension-$(n_1,\cdots,n_k)$ tensor T is a multidimensional array. It can be thought as a function on the product space $[n_1]\times\cdots\times[n_k]$ which identifies the value of each of its entries.

$$\begin{array}{rccl} T\colon & [n_1]\times\cdots\times[n_k] & \to & \mathbb{R} \\ & (i_1,\cdots,i_k) & \mapsto & T(i_1,\cdots,i_k), \end{array}$$

where, as usual, we use $T(i_1,\cdots,i_k)$ to denote its $(i_1,\cdots,i_k)$-th entry.

Without specification, all matrices and tensors are over the real number field. The space of $n \times m$ matrices is denoted by

$$\mathsf{Mat}_{n\times m} := \left\{\mathbf{M} \in \mathbb{R}^{n\times m}\right\} \cong \mathbb{R}^{n\cdot m}.$$

When $n = m$, we write Mat_n for the space of square matrices of dimension n. The space of order-k dimension-$(n_1, \cdots, n_k)$ tensors is denoted by

$$\mathsf{Ten}^{\otimes k}_{n_1,\cdots,n_k} := \left\{T \in \mathbb{R}^{n_1\times\cdots\times n_k}\right\} \cong \mathbb{R}^{n_1\cdots n_k}.$$

If every dimension of T is the same, $n_1 = \cdots = n_k = n$, then we write $\mathsf{Ten}^{\otimes k}_n$ for the space of equilateral tensors of order k and dimension n. Definitions of the sets of *symmetric* (Sym), *non-negative* (NN), *doubly non-negative* (DNN), *positive semidefinite* (PSD), *completely positive* (CP), *copositive* (coP), etc., matrices and tensors are deferred to the corresponding sections where we need them. Note that $\mathsf{Mat}_{n,m} = \mathsf{Ten}^{\otimes 2}_{n,m}$. When the order of the tensors is $k = 2$, namely matrices, we drop the superscript $\otimes 2$.

For a tensor $T \in \mathsf{Ten}^{\otimes k}_{n_1,\cdots,n_k}$, we use $\|T\|_{\mathrm{F}}$ to denote the *Frobenius norm* of T, which is the ℓ^2 norm when T is vectorized into a length-$n_1\cdots n_k$ vector.

$$\|T\|_{\mathrm{F}} := \left(\sum_{(i_1,\cdots,i_k)\in[n_1]\times\cdots\times[n_k]} T(i_1,\cdots,i_k)^2\right)^{1/2}.$$

We use $\|T\|_{\mathrm{sav}}$ to denote the *sum-absolute-value norm* of T which is the ℓ^1 norm after vectorization.

$$\|T\|_{\mathrm{sav}} := \sum_{(i_1,\cdots,i_k)\in[n_1]\times\cdots\times[n_k]} |T(i_1,\cdots,i_k)|.$$

Similarly, define

$$\|T\|_{\mathrm{mav}} := \max_{(i_1,\cdots,i_k)\in[n_1]\times\cdots\times[n_k]} |T(i_1,\cdots,i_k)|$$

to be the *max-absolute-value* norm of T, which is the ℓ^∞ norm when viewed as a vector.

Note that the Frobenius norm, sum-absolute-value norm and max-absolute-value are different from the matrix/tensor 2-norm, 1-norm and ∞-norm. Though they do trivially coincide with the corresponding vector norm when the order of the tensor is one.

We endow the matrix/tensor space with an inner product. For tensors T_1 and T_2 both in $\mathsf{Ten}^{\otimes k}_{n_1,\cdots,n_k}$,

$$\langle T_1, T_2\rangle := \sum_{(i_1,\cdots,i_k)\in[n_1]\times\cdots\times[n_k]} T_1(i_1,\cdots,i_k)T_2(i_1,\cdots,i_k).$$

When T_1, T_2 are matrices, the above definition agrees with the *Frobenius inner product*, which is alternatively defined as $\mathrm{Tr}\left(T_1^\top T_2\right)$. When T_1, T_2 are vectors, this inner product becomes the standard inner product associated to $\mathbb{R}^n$ as a Hilbert space, which is denoted by the same notation without confusion.

Let S_n denote the *symmetric group* of degree n consisting of $n!$ permutations on $[n]$. Permutations are typically denoted by lower case Greek letters.

Information theory. We use $H(\cdot)$ to interchangeably denote the binary entropy function and the Shannon entropy; the exact meaning will be clear from the context. In particular, for any $p \in [0,1]$, $H(p)$ denotes the binary entropy

$$H(p) = p \log \frac{1}{p} + (1-p) \log \frac{1}{1-p}.$$

For a distribution $P \in \Delta(\mathcal{X})$ on a finite alphabet $\mathcal{X}$ or a random variable $\mathbf{x} \sim P$ distributed according to P, the Shannon entropy of P or $\mathbf{x}$ is defined similarly as

$$H(P) = H(\mathbf{x}) \coloneqq \sum_{x \in \mathcal{X}} P(x) \log \frac{1}{P(x)}.$$

For two distributions $P, Q \in \Delta(\mathcal{X})$ on the same alphabet $\mathcal{X}$, the *Kullback–Leibler (KL) divergence* between them is defined as

$$D(P \| Q) \coloneqq \sum_{x \in \mathcal{X}} P(x) \log \frac{P(x)}{Q(x)}.$$

If $\mathbf{x}, \mathbf{y}$ are jointly distributed according to $P_{\mathbf{x},\mathbf{y}} \in \Delta(\mathcal{X} \times \mathcal{Y})$, then their *joint entropy* is defined as

$$H(\mathbf{x}, \mathbf{y}) = H(P_{\mathbf{x},\mathbf{y}}) \coloneqq \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} P_{\mathbf{x},\mathbf{y}}(x,y) \log \frac{1}{P_{\mathbf{x},\mathbf{y}}(x,y)};$$

their *mutual information* is defined as

$$\begin{aligned} I(\mathbf{x}; \mathbf{y}) \coloneqq & D\left(P_{\mathbf{x},\mathbf{y}} \| P_{\mathbf{x}} P_{\mathbf{y}}\right) \\ = & \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} P_{\mathbf{x},\mathbf{y}}(x,y) \log \frac{P_{\mathbf{x},\mathbf{y}}(x,y)}{P_{\mathbf{x}}(x) P_{\mathbf{y}}(y)} \\ = & \sum_{y \in \mathcal{Y}} P_{\mathbf{y}}(y) \sum_{x \in \mathcal{X}} P_{\mathbf{x}|\mathbf{y}}(x|y) \log \frac{P_{\mathbf{x}|\mathbf{y}}(x|y)}{P_{\mathbf{x}}(x)}. \end{aligned}$$

If the conditional distribution of $\mathbf{y}$ given $\mathbf{x}$ is $P_{\mathbf{y}|\mathbf{x}} \in \Delta(\mathcal{Y}|\mathcal{X})$, then the *conditional entropy* of $\mathbf{y}$ given $\mathbf{x}$ is defined as

$$\begin{aligned} H(\mathbf{y}|\mathbf{x}) \coloneqq & \sum_{x \in \mathcal{X}} P_{\mathbf{x}}(x) H(\mathbf{y}|\mathbf{x} = x) \\ = & \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{X}} P_{\mathbf{x},\mathbf{y}}(x,y) \log \frac{P_{\mathbf{x}}(x)}{P_{\mathbf{x},\mathbf{y}}(x,y)}. \end{aligned}$$

It is easy to check that different definitions above for the same quantities are consisted with each other.

8 Preliminaries

▶ **Lemma 10** (Stirling's approximation). *For any $n \in \mathbb{Z}_{>0}$, $n! \asymp \sqrt{2\pi n}\,(n/e)^n$.*

▶ **Corollary 11** (Asymptotics of multinomials). *For any positive integers $n \geqslant q$ and any q-partition $(n_1, \cdots, n_q)$ of n ($n_1 + \cdots + n_q = n$, $n_i \geqslant 0$ for every i), $\binom{n}{n_1, \cdots, n_q} \doteq 2^{nH(P)}$, where $P \in \Delta([q])$ is an empirical distribution such that for $i \in [q]$, $P(i) = n_i/n$. More precisely, we have $\binom{n}{n_1, \cdots, n_q} \asymp \nu(n)^{-1} 2^{nH(P)}$, where $\nu(n)$ is a polynomial defined as*

$$\nu(n) \coloneqq (2\pi n)^{\frac{q-1}{2}} \left(\prod_{i=1}^{q} P(i) \right)^{\frac{1}{2}}.$$

▶ **Fact 12** (Approximation of binomials). *For any positive integers $n \geqslant k$,*

$$\left(\frac{n}{k}\right)^k \leqslant \binom{n}{k} \leqslant \left(\frac{en}{k}\right)^k, \tag{6}$$

$$(n-k)^k \leqslant (n-k+1)^k \leqslant \binom{n}{k} \leqslant n^k. \tag{7}$$

Without loss of generality, we write $\mathcal{X} = \{1, \cdots, |\mathcal{X}|\}$. For $\underline{x} \in \mathcal{X}^n$ and $x \in \mathcal{X}$, let

$$N_x(\underline{x}) := \left|\{i \in [n] \colon \underline{x}(i) = x\}\right|,$$

which counts the number of occurrences of a symbol x in a vector $\underline{x}$. Similarly, define

$$N_{x,y}\left(\underline{x}, \underline{y}\right) := \left|\left\{i \in [n] \colon \underline{x}(i) = x,\ \underline{y}(i) = y\right\}\right|.$$

▶ **Definition 13** (Types). *For a length-n vector $\underline{x}$ over a finite alphabet $\mathcal{X}$, the type $\tau_{\underline{x}}$ of $\underline{x}$ is a length-$|\mathcal{X}|$ (empirical) probability vector (or the histogram of $\underline{x}$), i.e., $\tau_{\underline{x}} \in [0,1]^{|\mathcal{X}|}$ has entries $\tau_{\underline{x}}(x) := N_x(\underline{x})/n$ for all $x \in \mathcal{X}$.*

▶ **Definition 14** (Joint types and conditional types). *The* joint type $\tau_{\underline{x},\underline{y}} \in [0,1]^{|\mathcal{X}|\times|\mathcal{Y}|}$ *of two vectors $\underline{x} \in \mathcal{X}^n$ and $\underline{y} \in \mathcal{Y}^n$ is defined as $\tau_{\underline{x},\underline{y}}(x,y) = N_{x,y}(\underline{x},\underline{y})/n$ for $x \in \mathcal{X}$ and $y \in \mathcal{Y}$.*

The conditional type $\tau_{\underline{y}|\underline{x}} \in [0,1]^{|\mathcal{X}|\times|\mathcal{Y}|}$ of a vector $\underline{y} \in \mathcal{Y}^n$ given another vector $\underline{x} \in \mathcal{X}^n$ is defined as $\tau_{\underline{y}|\underline{x}}(y|x) = N_{x,y}\left(\underline{x},\underline{y}\right)/N_x\left(\underline{x}\right)$.

▶ Remark 15 (Types vs. distributions). Types are empirical distributions of length-n vectors. They can only take rational values, in particular, a/n for $a \in \{0,1,\cdots,n\}$. For finite alphabets and a fixed n, there are only poly(n) many types. However, there are uncountably infinitely many distributions on any finite alphabets and they form a probability simplex.

▶ Remark 16. We will also write $\tau_{\mathbf{x}}, \tau_{\mathbf{x},\mathbf{y}}, \tau_{\mathbf{y}|\underline{x}}, \tau_{\mathbf{y}|\mathbf{x}}$, etc., for generic types that are taken from the corresponding sets of types even if they do not come from instantiated vectors. For instance, $\tau_{\mathbf{x}}$ is a type in $\mathcal{P}^{(n)}(\mathcal{X})$ corresponding to any $\underline{x} \in \mathcal{T}_{\underline{\mathbf{x}}}(\tau_{\mathbf{x}})$. The particular choice of $\underline{x}$ is not important and will not be specified. These notations are for explicitly distinguishing types from distributions.

▶ **Definition 17** (Set of types). *We use $\mathcal{P}^{(n)}(\mathcal{X})$ to denote the set of types of all length-n vectors over $\mathcal{X}$.*

$$\mathcal{P}^{(n)}(\mathcal{X}) = \left\{\tau_{\underline{x}} \colon \underline{x} \in \mathcal{X}^n\right\}.$$

Similarly, define

$$\mathcal{P}^{(n)}(\mathcal{X},\mathcal{Y}) = \left\{\tau_{\underline{x},\underline{y}} \colon \underline{x} \in \mathcal{X}^n,\ \underline{y} \in \mathcal{Y}^n\right\},$$
$$\mathcal{P}^{(n)}(\mathcal{Y}|\underline{x}) = \left\{\tau_{\underline{y}|\underline{x}} \colon \underline{y} \in \mathcal{Y}^n\right\},$$
$$\mathcal{P}^{(n)}(\mathcal{Y}|\mathcal{X}) = \left\{\tau_{\underline{y}|\underline{x}} \colon \underline{x} \in \mathcal{X}^n,\ \underline{y} \in \mathcal{Y}^n\right\}$$

to be

1. *the set of all joint types;*
2. *the set of all conditional types of $\underline{y}$ given a particular $\underline{x}$;*
3. *the set of all conditional types of $\underline{y}$ given some $\underline{x}$,*

respectively.

▶ **Lemma 18** (Types are dense in distributions). *The union of the sets of types of all possible blocklengths is dense in the set of distributions, i.e.,*

$$\bigcup_{n=1}^{\infty} \mathcal{P}^{(n)}(\mathcal{X})$$

is dense in $\Delta(\mathcal{X})$. This holds true for joint types and conditional types as well.

▶ **Lemma 19** (Number of types). *When alphabet sizes are constants, the number of types of length-n vectors is polynomial in n. To be precise, the number of types of length-n vectors over $\mathcal{X}$ is*

$$\left|\mathcal{P}^{(n)}(\mathcal{X})\right| = \binom{n+|\mathcal{X}|-1}{|\mathcal{X}|-1}. \tag{8}$$

For a vector $\underline{x} \in \mathcal{X}^n$ of type $\tau_{\underline{x}}$, the number of conditional types of length-n vectors over $\mathcal{Y}$ given $\underline{x}$ is

$$\left|\mathcal{P}^{(n)}(\mathcal{Y}|\underline{x})\right| = \prod_{x\in\mathcal{X}} \binom{\tau_{\underline{x}}(x)n+|\mathcal{Y}|-1}{|\mathcal{Y}|-1}. \tag{9}$$

The number of conditional types of $\mathcal{Y}$-valued vectors given some $\mathcal{X}$-valued vector is

$$\mathcal{P}^{(n)}(\mathcal{Y}|\mathcal{X}) = \sum_{\tau_{\mathbf{x}}\in\mathcal{P}^{(n)}(\mathcal{X})} \prod_{x\in\mathcal{X}} \binom{\tau_{\mathbf{x}}(x)n+|\mathcal{Y}|-1}{|\mathcal{Y}|-1}. \tag{10}$$

The following elementary bounds from [16] are sufficient for the purposes of this paper.

$$\left|\mathcal{P}^{(n)}(\mathcal{X})\right| \leqslant (n+1)^{|\mathcal{X}|},$$
$$\left|\mathcal{P}^{(n)}(\mathcal{Y}|\underline{x})\right| \leqslant \left|\mathcal{P}^{(n)}(\mathcal{Y}|\mathcal{X})\right| \leqslant (n+1)^{|\mathcal{X}|\cdot|\mathcal{Y}|}.$$

▶ **Definition 20** (Type classes). *Define type class $\mathcal{T}_{\underline{\mathbf{x}}}(\tau_{\mathbf{x}})$ w.r.t. a type $\tau_{\mathbf{x}} \in \mathcal{P}^{(n)}(\mathcal{X})$ as*

$$\mathcal{T}_{\underline{\mathbf{x}}}(\tau_{\mathbf{x}}) \coloneqq \left\{\underline{x} \in \mathcal{X}^n \colon \tau_{\underline{x}} = \tau_{\mathbf{x}}\right\}.$$

Similarly, the joint type class $\mathcal{T}_{\underline{\mathbf{x}},\underline{\mathbf{y}}}(\tau_{\mathbf{x},\mathbf{y}})$ w.r.t. a joint type $\tau_{\mathbf{x},\mathbf{y}} \in \mathcal{P}^{(n)}(\mathcal{X}\times\mathcal{Y})$ is defined as

$$\mathcal{T}_{\underline{\mathbf{x}},\underline{\mathbf{y}}}(\tau_{\mathbf{x},\mathbf{y}}) \coloneqq \left\{(\underline{x},\underline{y}) \in \mathcal{X}^n \times \mathcal{Y}^n \colon \tau_{\underline{x},\underline{y}} = \tau_{\mathbf{x},\mathbf{y}}\right\}.$$

The conditional type class $\mathcal{T}_{\underline{\mathbf{y}}|\underline{x}}(\tau_{\mathbf{y}|\underline{x}})$ w.r.t. a conditional type $\tau_{\mathbf{y}|\underline{x}} \in \mathcal{P}^{(n)}(\mathcal{Y}|\underline{x})$ given a vector $\underline{x} \in \mathcal{X}^n$ is defined as

$$\mathcal{T}_{\underline{\mathbf{y}}|\underline{x}}(\tau_{\mathbf{y}|\underline{x}}) \coloneqq \left\{\underline{y} \in \mathcal{Y}^n \colon \tau_{\underline{y}|\underline{x}} = \tau_{\mathbf{y}|\underline{x}}\right\}.$$

The conditional type class $\mathcal{T}_{\underline{\mathbf{y}}|\underline{\mathbf{x}}}(\tau_{\mathbf{y}|\mathbf{x}})$ w.r.t. a conditional type $\tau_{\mathbf{y}|\mathbf{x}} \in \mathcal{P}^{(n)}(\mathcal{Y}|\mathcal{X})$ is defined as

$$\mathcal{T}_{\underline{\mathbf{y}}|\underline{\mathbf{x}}}(\tau_{\mathbf{y}|\mathbf{x}}) \coloneqq \bigcup_{\tau_{\mathbf{x}}\in\mathcal{P}^{(n)}(\mathcal{X})} \mathcal{T}_{\underline{\mathbf{y}}|\underline{x}'}(\tau_{\mathbf{y}|\mathbf{x}}) \tag{11}$$

$$= \left\{\underline{y} \in \mathcal{Y}^n \colon \exists \underline{x}' \in \mathcal{X}^n,\ \tau_{\underline{y}|\underline{x}'} = \tau_{\mathbf{y}|\mathbf{x}}\right\}, \tag{12}$$

where in Eqn. (11) $\underline{x}'$ can be chosen arbitrarily from $\mathcal{T}_{\underline{\mathbf{x}}}(\tau_{\mathbf{x}})$.

▶ **Lemma 21** (Size of type classes)**.**

1. *For any type* $\tau_{\mathbf{x}} \in \mathcal{P}^{(n)}(\mathcal{X})$, $\left|\mathcal{T}_{\underline{\mathbf{x}}}(\tau_{\mathbf{x}})\right| \doteq 2^{nH(P_{\mathbf{x}})}$.
2. *For any vector* $\underline{x} \in \mathcal{X}^n$ *and any conditional type* $\tau_{\mathbf{y}|\underline{x}} \in \mathcal{P}^{(n)}(\mathcal{Y}|\underline{x})$, $\left|\mathcal{T}_{\underline{\mathbf{y}}|\underline{x}}\left(\tau_{\mathbf{y}|\underline{x}}\right)\right| \doteq 2^{nH(\mathbf{y}|\mathbf{x})}$, *where the conditional entropy is evaluated w.r.t. the joint type* $\tau_{\underline{x}}\tau_{\mathbf{y}|\underline{x}}$.
3. *For any conditional type* $\tau_{\mathbf{y}|\mathbf{x}} \in \mathcal{P}^{(n)}(\mathcal{Y}|\mathcal{X})$,

$$\left|\mathcal{T}_{\underline{\mathbf{y}}|\underline{\mathbf{x}}}\left(\tau_{\mathbf{y}|\mathbf{x}}\right)\right| \doteq 2^{n \max_{\tau_{\mathbf{x}} \in \mathcal{P}^{(n)}(\mathcal{X})} H(\mathbf{y}|\mathbf{x})},$$

where the conditional entropy is evaluated w.r.t. the joint type $\tau_{\mathbf{x}}\tau_{\mathbf{y}|\mathbf{x}}$.

Proof.

1. The number of sequences $\underline{x} \in \mathcal{X}^n$ of type $\tau_{\mathbf{x}}$ is precisely

$$\binom{n}{n\tau_{\mathbf{x}}(1), \cdots, n\tau_{\mathbf{x}}(|\mathcal{X}|)}$$

and the claim follows from Lemma 10.

2. Given $\underline{x} \in \mathcal{X}^n$, the number of sequences $\underline{y} \in \mathcal{Y}^n$ of conditional type $\tau_{\mathbf{y}|\underline{x}}$ is precisely

$$\prod_{x \in \mathcal{X}} \binom{n\tau_{\underline{x}}(x)}{n\tau_{\mathbf{y}|\underline{x}}(1|x), \cdots, n\tau_{\mathbf{y}|\underline{x}}(|\mathcal{Y}|\,|x)},$$

and the lemma follows from 10.

3. Note that

$$\mathcal{T}_{\underline{\mathbf{y}}|\underline{x}^*}\left(\tau_{\mathbf{y}|\underline{x}^*}\right) \leqslant \left|\mathcal{T}_{\underline{\mathbf{y}}|\underline{\mathbf{x}}}\left(\tau_{\mathbf{y}|\mathbf{x}}\right)\right| \leqslant \left|\mathcal{P}^{(n)}(\mathcal{X})\right| \mathcal{T}_{\underline{\mathbf{y}}|\underline{x}^*}\left(\tau_{\mathbf{y}|\underline{x}^*}\right),$$

where $\underline{x}^*$ is chosen arbitrarily from $\mathcal{T}_{\underline{\mathbf{x}}}\left(\tau^*_{\mathbf{x}}\right)$ and[11]

$$\tau^*_{\mathbf{x}} = \operatorname*{argmax}_{\tau_{\mathbf{x}} \in \mathcal{P}^{(n)}(\mathcal{X})} \left|\mathcal{T}_{\underline{\mathbf{y}}|\underline{x}}\left(\tau_{\underline{\mathbf{y}}|\underline{x}}\right)\right|.$$

The claim follows from Eqn. (8) and the previous claim.

◀

▶ **Lemma 22.** *If* $\underline{\mathbf{x}}$ *is generated using the product distribution* $P_{\mathbf{x}}^{\otimes n}$*, then for any* $\underline{x} \in \mathcal{T}_{\underline{\mathbf{x}}}(P_{\mathbf{x}})$,

$$\Pr\left[\underline{\mathbf{x}} = \underline{x}\right] = 2^{-nH(P_{\mathbf{x}})}.$$

Moreover,

$$\Pr\left[\underline{\mathbf{x}} \in \mathcal{T}_{\underline{\mathbf{x}}}(P_{\mathbf{x}})\right] \asymp \nu(n)^{-1}.$$

Proof. Both claims follow from elementary calculations. For the first one,

$$\begin{aligned}\Pr\left[\underline{\mathbf{x}} = \underline{x}\right] &= \prod_{x \in \mathcal{X}} P_{\mathbf{x}}(x)^{N_x(\underline{x})} \\ &= 2^{\sum_{x \in \mathcal{X}} N_x(\underline{x}) \log P_{\mathbf{x}}(x)} \\ &= 2^{n \sum_{x \in \mathcal{X}} P_{\mathbf{x}}(x) \log P_{\mathbf{x}}(x)} \qquad (13) \\ &= 2^{-nH(P_{\mathbf{x}})},\end{aligned}$$

where Eqn. (13) is because $\tau_{\underline{x}} = P_{\mathbf{x}}$ and hence $N_x(\underline{x})/n = P_{\mathbf{x}}(x)$ for any $x \in \mathcal{X}$.

[11] In the argmax, $\underline{x} \in \mathcal{T}_{\underline{\mathbf{x}}}\left(\tau_{\mathbf{x}}\right)$ is arbitrary as well.

For the second one,

$$\begin{aligned}\Pr\left[\underline{\mathbf{x}} \in \mathcal{T}_{\underline{\mathbf{x}}}(P_{\mathbf{x}})\right] =& \Pr\left[\tau_{\underline{\mathbf{x}}} = P_{\mathbf{x}}\right] \\ =& \binom{n}{nP_{\mathbf{x}}(1), \cdots, nP_{\mathbf{x}}(|\mathcal{X}|)} \prod_{x\in\mathcal{X}} P_{\mathbf{x}}(x)^{nP_{\mathbf{x}}(x)} \\ \asymp& \nu(n)^{-1} 2^{nH(P)} 2^{-nH(P)} \qquad (14) \\ =& \nu(n)^{-1},\end{aligned}$$

where Eqn. (14) is by Corollary 11. ◀

▶ **Lemma 23** (Markov). *For any non-negative random variable X and any positive number x,*

$$\Pr[X \geqslant x] \leqslant \frac{\mathbb{E}[X]}{x}.$$

▶ **Lemma 24** (Chernoff). *Let $X_1, \cdots, X_n$ be independent (not necessarily identically distributed) $\{0,1\}$-valued random variables. Let*

$$X := \sum_{i=1}^{n} X_i.$$

Then

$$\begin{aligned}\Pr[X \geqslant (1+\epsilon)\mathbb{E}[X]] \leqslant& e^{-\frac{\epsilon^2}{3}\mathbb{E}[X]}, \\ \Pr[X \leqslant (1-\epsilon)\mathbb{E}[X]] \leqslant& e^{-\frac{\epsilon^2}{2}\mathbb{E}[X]}, \\ \Pr[X \notin (1\pm\epsilon)\mathbb{E}[X]] \leqslant& 2e^{-\frac{\epsilon^2}{3}\mathbb{E}[X]}.\end{aligned}$$

▶ **Lemma 25** (Sanov). *Let $\mathcal{Q} \subset \Delta(\mathcal{X})$ be a subset of distributions such that it is equal to the closure of its interior. Let $\underline{\mathbf{x}} \sim P_{\mathbf{x}}^{\otimes n}$ be a random vector whose components are i.i.d. according to $P_{\mathbf{x}}$. Clearly $\underline{\mathbf{x}}$ is expected to have type $\mathbb{E}\left[\tau_{\underline{\mathbf{x}}}\right] = P_{\mathbf{x}}$. Sanov's theorem determines the first-order exponent of the probability that the vector empirically looks like coming from some distribution $Q \in \mathcal{Q}$.*

$$\Pr\left[\tau_{\underline{\mathbf{x}}} \in \mathcal{Q}\right] \doteq 2^{-n \inf_{Q\in\mathcal{Q}} D(Q\|P_{\mathbf{x}})}.$$

▶ Remark 26. One can view Sanov's theorem as a particular form of the Chernoff bound. Since $\underline{\mathbf{x}}(i)$'s are independent, it gives the *correct* exponent of $\Pr\left[\tau_{\underline{\mathbf{x}}} \in \mathcal{Q}\right]$ (up to lower order terms) rather than being merely a bound.

▶ **Lemma 27** (Anti-concentration). *Let X be a non-negative random variable. Then*

$$\Pr[X = 0] \leqslant \frac{Var[X]}{\mathbb{E}[X]^2}.$$

▶ **Fact 28** (Binomial identities). *For any non-negative integers n, K and $0 \leqslant k \leqslant n$, we have*

$$\binom{n}{k} = \binom{n}{n-k}, \qquad (15)$$

$$\binom{n}{k} = \frac{n}{k}\binom{n-1}{k-1}, \qquad (16)$$

$$\binom{n}{k} + \binom{n}{k+1} = \binom{n+1}{k+1}, \tag{17}$$

$$2^K = \sum_{i=0}^{K} \binom{n}{i}. \tag{18}$$

We list several basic (in)equalities concerning information measures that we will frequently refer to.

▶ **Fact 29** (Information (in)equalities). *The following inequalities hold for any random variables/distributions over finite sets.*

$$\begin{aligned} H(\mathbf{x},\mathbf{y}) =& H(\mathbf{x}) + H(\mathbf{y}|\mathbf{x}) \\ =& H(\mathbf{y}) + H(\mathbf{x}|\mathbf{y}) \\ =& H(\mathbf{x}|\mathbf{y}) + H(\mathbf{y}|\mathbf{x}) + I(\mathbf{x};\mathbf{y}) \\ =& H(\mathbf{x}) + H(\mathbf{y}) - I(\mathbf{x};\mathbf{y}), \\ I(\mathbf{x};\mathbf{y}) =& H(\mathbf{x}) - H(\mathbf{x}|\mathbf{y}) \\ =& H(\mathbf{y}) - H(\mathbf{y}|\mathbf{x}) \\ =& D\left(P_{\mathbf{x},\mathbf{y}} \| P_{\mathbf{x}} P_{\mathbf{y}}\right). \end{aligned}$$

9 Basic definitions

▶ **Definition 30** (Adversarial channels). *An adversarial channel* $\mathcal{A} = (\mathcal{X}, \lambda_{\mathbf{x}}, \mathcal{S}, \lambda_{\mathbf{s}}, \mathcal{Y}, W_{\mathbf{y}|\mathbf{x},\mathbf{s}})$ *(Fig. 3) is a sextuple consisting of*

1. *an input alphabet* $\mathcal{X}$*;*
2. *a set of input constraints* $\lambda_{\mathbf{x}} \subseteq \mathcal{P}^{(n)}(\mathcal{X})$*;*
3. *a noise alphabet* $\mathcal{S}$*;*
4. *a set of noise constraints* $\lambda_{\mathbf{s}} \subseteq \mathcal{P}^{(n)}(\mathcal{S})$*;*
5. *an output alphabet* $\mathcal{Y}$*;*
6. *a channel law given by a transition probability* $W_{\mathbf{y}|\mathbf{x},\mathbf{s}} \in \Delta(\mathcal{Y}|\mathcal{X} \times \mathcal{S})$.

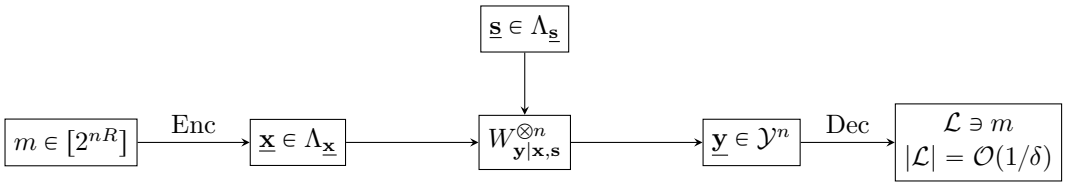


Figure 3 General adversarial channels.

▶ Remark 31. In this paper, we are only concerned with finite alphabets of constant size independent of the blocklength n.

Specifically,

- Though the alphabets $\mathcal{X}, \mathcal{S}$ and $\mathcal{Y}$ can be arbitrary finite sets, it is without loss of generality to realize them using the first $|\mathcal{X}|, |\mathcal{S}|$ and $|\mathcal{Y}|$ positive integers, i.e., $\mathcal{X} = [|\mathcal{X}|], \mathcal{S} = [|\mathcal{S}|]$ and $\mathcal{Y} = [|\mathcal{Y}|]$.[12]

[12] Under such realizations, these sets are *not* necessarily equipped with real arithmetic or modular arithmetic. The metric, if one cares, would be specified by the channel function.

- The input and noise constraint sets $\lambda_{\mathbf{x}}$ and $\lambda_{\mathbf{s}}$ are subsets of types $\mathcal{P}^{(n)}(\mathcal{X})$ and $\mathcal{P}^{(n)}(\mathcal{S})$ respectively. In this paper we assume they are *convex* sets. Since there are polynomially many types in total, we can also think of these collections of types as defined by intersections of hyperplanes or halfspaces, that is, types satisfying a certain finite number of linear (in the entries of the types) (in)equality constraints.
- In this paper, for technical simplicity, we assume that the channel transition function has only *singleton* mass. That is, for each $x \in \mathcal{X}, s \in \mathcal{S}$, $W_{\mathbf{y}|\mathbf{x},\mathbf{s}}(y|x,s) = 1$ only for one $y \in \mathcal{Y}$ and is zero for all other outputs. Equivalently, such degenerate distributions can be alternatively thought of as *deterministic* functions

$$\begin{array}{rccc} W\colon & \mathcal{X}\times\mathcal{S} & \to & \mathcal{Y} \\ & (x,s) & \mapsto & y, \end{array}$$

  where y is the *unique* output which is assigned the full probability, $W_{\mathbf{y}|\mathbf{x},\mathbf{s}}(y|x,s) = 1$. Here we slightly abuse the notation and use the same letter for the channel transition distribution and the channel transition function (when the distribution is degenerate). Moreover, we use $\underline{y} = W(\underline{x},\underline{s})$ (with the superscript $\otimes n$ being dropped) to denote the output of n uses of the channel, or equivalently, the n-letter output of the function which acts on $(\underline{x},\underline{s})$ component by component.
  It seems this is a severe restriction (and turns out indeed to be so). Nevertheless, it is still a very first and significant step towards understanding general adversarial channels in full generality. The case where $W_{\mathbf{y}|\mathbf{x},\mathbf{s}}$ is an arbitrary conditional distribution, or equivalently, the function W is *non-deterministic*, is interesting as well and is left as a future direction.
- For notational convenience, let

$$\begin{aligned} \Lambda_{\underline{\mathbf{x}}} &:= \left\{\underline{x} \in \mathcal{X}^n \colon \tau_{\underline{x}} \in \lambda_{\mathbf{x}}\right\} \\ &= \bigcup_{\tau_{\mathbf{x}} \in \lambda_{\mathbf{x}}} \mathcal{T}_{\underline{\mathbf{x}}}(\tau_{\mathbf{x}}), \\ \Lambda_{\underline{\mathbf{s}}} &:= \left\{\underline{s} \in \mathcal{S}^n \colon \tau_{\underline{s}} \in \lambda_{\mathbf{s}}\right\} \\ &= \bigcup_{\tau_{\mathbf{s}} \in \lambda_{\mathbf{s}}} \mathcal{T}_{\underline{\mathbf{s}}}(\tau_{\mathbf{s}}), \end{aligned}$$

  be sets of codewords and error patterns of admissible types.

▶ **Example 32.** Our framework covers a large family of channel models, including most of the popular and well-studied ones.

1. The standard bit-flip channels. $\mathcal{X} = \mathbb{F}_2, \lambda_{\mathbf{x}} = \mathcal{P}^{(n)}(\mathbb{F}_2), \mathcal{S} = \mathbb{F}_2$,

$$\lambda_{\mathbf{s}} = \left\{\tau_{\mathbf{s}} \in \mathcal{P}^{(n)}(\mathbb{F}_2) \colon \tau_{\mathbf{s}}(1) \leqslant p\right\},$$

   $\mathcal{Y} = \mathbb{F}_2, y = W(x,s) = x \,\mathsf{XOR}\, s$.
2. The standard q-ary channels. $\mathcal{X} = \mathbb{Z}_q$, $\lambda_{\mathbf{x}} = \mathcal{P}^{(n)}(\mathbb{Z}_q)$, $\mathcal{S} = \mathbb{Z}_q$,

$$\lambda_{\mathbf{s}} = \left\{\tau_{\mathbf{s}} \in \mathcal{P}^{(n)}(\mathbb{Z}_q) \colon \tau_{\mathbf{s}}(1) + \cdots + \tau_{\mathbf{s}}(q-1) \leqslant p\right\},$$

   $\mathcal{Y} = \mathbb{Z}_q, y = W(x,s) = x + s \mod q$.
3. The standard erasure channels. $\mathcal{S} = \mathbb{Z}_q, \lambda_{\mathbf{x}} = \mathcal{P}^{(n)}(\mathbb{Z}_q), \mathcal{S} = \mathbb{F}_2$,

$$\lambda_{\mathbf{s}} = \left\{\tau_{\mathbf{s}} \in \mathcal{P}^{(n)}(\mathbb{F}_2) \colon \tau_{\mathbf{s}}(1) \leqslant p\right\},$$

$\mathcal{Y} = \mathbb{Z}_q \cup \{\mathsf{erasure}\}$,

$$y = W(x,s) = \begin{cases} x, & s = 0 \\ \mathsf{erasure}, & s = 1 \end{cases}.$$

4. Weight constrained channels. Any of the above channels with

$$\lambda_{\mathbf{x}} = \left\{ \tau_{\mathbf{x}} \in \mathcal{P}^{(n)}(\mathcal{X}) \colon 1 - \tau_{\mathbf{x}}(0) \leqslant w \right\}.$$

5. Z-channels (or multiplier/AND channels). $\mathcal{X} = \mathbb{F}_2, \lambda_{\mathbf{x}} = \mathcal{P}^{(n)}(\mathbb{F}_2), \mathcal{S} = \mathbb{F}_2$,

$$\lambda_{\mathbf{s}} = \left\{ \tau_{\mathbf{s}} \in \mathcal{P}^{(n)}(\mathbb{F}_2) \colon \tau_{\mathbf{s}}(1) \leqslant p \right\},$$

$\mathcal{Y} = \mathbb{F}_2$,

$$y = W(x,s) = \begin{cases} 0, & s = 0 \text{ or } x = 0 \\ x, & s = 1 \text{ and } x = 1 \end{cases},$$

or equivalently $y = W(x,s) = x \,\mathsf{AND}\, s$.

6. Adder channels. $\mathcal{X} = \{0, 1, \cdots, q-1\}, \lambda_{\mathbf{x}} = \mathcal{P}^{(n)}(\mathcal{X}), \mathcal{S} = \{0, 1, \cdots, q-1\}$,

$$\lambda_{\mathbf{s}} = \left\{ \tau_{\mathbf{s}} \in \mathcal{P}^{(n)}(\mathcal{S}) \colon \tau_{\mathbf{s}}(1) + \cdots + \tau_{\mathbf{s}}(q-1) \leqslant p \right\},$$

$\mathcal{Y} = \{0, 1, \cdots, 2(q-1)\}, y = W(x,s) = x + s$, where the addition is over $\mathbb{R}$.

7. Noisy typewriter channels. $\mathcal{X} = \mathbb{Z}_q, \lambda_{\mathbf{x}} = \mathcal{P}^{(n)}(\mathbb{Z}_q), \mathcal{S} = \mathbb{F}_2, \lambda_{\mathbf{s}} = \mathcal{P}^{(n)}(\mathbb{F}_2), \mathcal{Y} = \mathbb{Z}_q, y = W(x,s) = x + s \mod q$.
8. OR channels (or $\mathfrak{T}$-channels). $\mathcal{X} = \mathbb{F}_2, \lambda_{\mathbf{x}} = \mathcal{P}^{(n)}(\mathbb{F}_2), \mathcal{S} = \mathbb{F}_2$,

$$\lambda_{\mathbf{s}} = \left\{ \tau_{\mathbf{s}} \in \mathcal{P}^{(n)}(\mathbb{F}_2) \colon \tau_{\mathbf{s}}(1) \leqslant p \right\},$$

$\mathcal{Y} = \mathbb{F}_2, y = W(x,s) = x \,\mathsf{OR}\, s$,

9. Channels under Lee distance. $\mathcal{X} = \mathbb{Z}_q, \lambda_{\mathbf{x}} = \mathcal{P}^{(n)}(\mathbb{Z}_q)$, $\mathcal{S} = \{-\lfloor \frac{q}{2} \rfloor, -\lfloor \frac{q}{2} \rfloor + 1, \cdots, \lfloor \frac{q}{2} \rfloor - 1, \lfloor \frac{q}{2} \rfloor\}$,

$$\lambda_{\mathbf{s}} = \left\{ \tau_{\mathbf{s}} \in \mathcal{P}^{(n)}(\mathcal{S}) \colon \sum_{s=1}^{\lfloor q/2 \rfloor} (\tau_{\mathbf{s}}(s) - \tau_{\mathbf{s}}(-s)) \cdot s \leqslant p \right\},$$

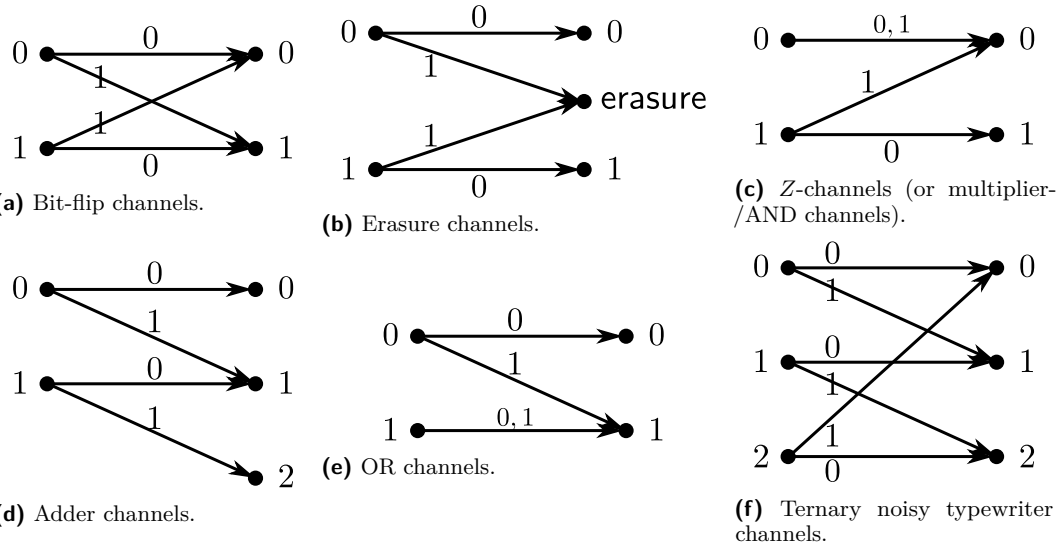
$\mathcal{Y} = \mathbb{Z}_q, y = W(x,s) = x + s$ over the reals.

10. Other more complicated channels, e.g., the one we defined in Sec. 1.

▶ **Definition 33** (Self-couplings). *A joint distribution $P_{\mathbf{x}_1, \cdots, \mathbf{x}_L} \in \Delta(\mathcal{X}^L)$ is said to be a* $(P_{\mathbf{x}}, L)$-self-coupling *for some $P_{\mathbf{x}} \in \Delta(\mathcal{X})$ if all of its marginals equal $P_{\mathbf{x}}$, i.e., $\left[P_{\mathbf{x}_1, \cdots, \mathbf{x}_L}\right]_{\mathbf{x}_i} = P_{\mathbf{x}}$ for all $i \in [L]$. The set of all $(P_{\mathbf{x}}, L)$-self-couplings is denoted by $\mathcal{J}^{\otimes L}(P_{\mathbf{x}})$.*

▶ **Definition 34** (Codes). *In general, a code $\mathcal{C}$ is a subset of $\mathcal{X}^n$. A code $\mathcal{C}$ for an adversarial channel $\mathcal{A} = (\mathcal{X}, \lambda_{\mathbf{x}}, \mathcal{S}, \lambda_{\mathbf{s}}, \mathcal{Y}, W_{\mathbf{y}|\mathbf{x},\mathbf{s}})$ is a subset of $\Lambda_{\underline{\mathbf{x}}}$; n is called the blocklength. Elements in $\mathcal{C}$ are called codewords. The rate $R(\mathcal{C})$ of $\mathcal{C}$ is defined as $R(\mathcal{C}) := (\log |\mathcal{C}|)/n$.*

▶ **Definition 35** (Constant composition codes). *A code $\mathcal{C} \subset \mathcal{X}^n$ is said to be $P_{\mathbf{x}}$-constant composition for some $P_{\mathbf{x}} \in \Delta(\mathcal{X})$ if the type of each codeword is $P_{\mathbf{x}}$, i.e., $\tau_{\underline{x}} = P_{\mathbf{x}}$ for every $\underline{x} \in \mathcal{C}$.*



(a) Bit-flip channels.

(b) Erasure channels.

(c) Z-channels (or multiplier-/AND channels).

(d) Adder channels.

(e) OR channels.

(f) Ternary noisy typewriter channels.

Figure 4 Examples of various well-studied channel models.

▶ **Lemma 36.** *For any code $\mathcal{C} \subset \mathcal{X}^n$ of rate R, there is a constant composition subcode $\mathcal{C}' \subseteq \mathcal{C}$ of asymptotically the same rate.*

Proof. Let $\mathcal{C}' = \mathcal{C} \cap \mathcal{T}_{\underline{\mathbf{x}}}(\tau^*_{\mathbf{x}})$, where

$$\tau^*_{\mathbf{x}} = \operatorname*{argmax}_{\tau_{\mathbf{x}} \in \mathcal{P}^{(n)}(\mathcal{X})} \left|\mathcal{C} \cap \mathcal{T}_{\underline{\mathbf{x}}}(\tau_{\mathbf{x}})\right|$$

is the most common type in $\mathcal{C}$. By Lemma 8 and Lemma 23,

$$\left|\mathcal{C}'\right| \geqslant \frac{|\mathcal{C}|}{(n+1)^{|\mathcal{X}|}} = 2^{nR + |\mathcal{X}| \log(n+1)},$$

which implies that $R(\mathcal{C}') \asymp R(\mathcal{C})$ as n grows. ◀

▶ **Definition 37** (Confusability of tuples of vectors)**.** *A list of L distinct codewords $\underline{x}_1, \cdots, \underline{x}_L \in \mathcal{X}^n$ is said to be* L-confusable *if there are $\underline{y} \in \mathcal{Y}^n$ and $\underline{s}_1, \cdots, \underline{s}_L \in \Lambda_{\underline{\mathbf{s}}}$ such that $W(\underline{x}_i, \underline{s}_i) = \underline{y}$ for all $i \in [L]$.*

▶ **Definition 38** (Confusability of joint distributions)**.** *A $(P_{\mathbf{x}}, L)$-self-coupling $P_{\mathbf{x}_1,\cdots,\mathbf{x}_L} \in \mathcal{J}^{\otimes L}(P_{\mathbf{x}})$ is said to be* L-confusable *if it has some extension given by $P_{\mathbf{x}_1,\cdots,\mathbf{x}_L,\mathbf{s}_1,\cdots,\mathbf{s}_L,\mathbf{y}} \in \Delta\left(\mathcal{X}^L \times \mathcal{S}^L \times \mathcal{Y}\right)$ such that*

1. $[P_{\mathbf{x},\cdots,\mathbf{x}_L,\mathbf{s}_1,\cdots,\mathbf{s}_L,\mathbf{y}}]_{\mathbf{x}_1,\cdots,\mathbf{x}_L} = P_{\mathbf{x}_1,\cdots,\mathbf{x}_L}$;
2. $P_{\mathbf{s}_i} \in \lambda_{\mathbf{s}}$ *for all* $i \in [L]$;
3. $P_{\mathbf{x}_i,\mathbf{s}_i,\mathbf{y}} = P_{\mathbf{x}} P_{\mathbf{s}_i|\mathbf{x}_i} W_{\mathbf{y}|\mathbf{x}_i,\mathbf{s}_i}$ *for all* $i \in [L]$.

▶ **Definition 39** (Confusability set)**.** *The $(P_{\mathbf{x}}, L)$-confusability set $\mathcal{K}^{\otimes L}(P_{\mathbf{x}})$ of a channel $\mathcal{A} = \left(\mathcal{X}, \lambda_{\mathbf{x}}, \mathcal{S}, \lambda_{\mathbf{s}}, \mathcal{Y}, W_{\mathbf{y}|\mathbf{x},\mathbf{s}}\right)$ is defined as*

$$\mathcal{K}^{\otimes L}(P_{\mathbf{x}}) \coloneqq \left\{P_{\mathbf{x}_1,\cdots,\mathbf{x}_L} \in \mathcal{J}^{\otimes L}(P_{\mathbf{x}}) \colon P_{\mathbf{x}_1,\cdots,\mathbf{x}_L} \text{ is } L\text{-confusable}\right\}.$$

▶ Remark 40. In the above definitions, we overload the notion of confusability for types and distributions.

$$\mathcal{K}^{\otimes L}(P_{\mathbf{x}}) = \bigcup_{n=1}^{\infty} \left\{\tau_{\underline{x}_1,\cdots,\underline{x}_L} \colon (\underline{x}_1, \cdots, \underline{x}_L) \text{ is } L\text{-confusable};\ \underline{x}_i \in \mathcal{T}_{\underline{\mathbf{x}}}(P_{\mathbf{x}}),\ \forall i \in [L]\right\}.$$

▶ **Definition 41** (List decodable codes). *A code $\mathcal{C} \subset \mathcal{X}^n$ is said to be $(L-1)$-*list decodable *if no size-L list is confusable, i.e., for any $\mathcal{L} \in \binom{\mathcal{C}}{L}$, $\mathcal{L}$ is non-L-confusable.*

▶ **Definition 42** (Achievable rate and list decoding capacity). *A rate R is said to be achievable under $(L-1)$-list decoding if there is an infinite sequence of $(L-1)$-list decodable codes $\{\mathcal{C}_i\}_{i\geqslant 1}$ of blocklength $n_i \in \mathbb{Z}_{>0}$ (such that $\{n_i\}$ is a non-vanishing sequence) and rate $R(\mathcal{C}) \geqslant R$.*

The $(L-1)$-list decoding capacity is defined as the maximal achievable rate.

$$C := \limsup_{n\to\infty} \max_{\substack{\mathcal{C}\subseteq\Lambda_{\underline{\mathbf{x}}} \\ (L-1)\text{-list decodable}}} R(\mathcal{C}).$$

10 List decoding capacity

▶ **Theorem 43** (List decoding capacity). *For any adversarial channel $\mathcal{A} = (\mathcal{X}, \lambda_{\mathbf{x}}, \mathcal{S}, \lambda_{\mathbf{s}}, \mathcal{Y}, W)$, let*[13]

$$C := \max_{P_{\mathbf{x}}\in\lambda_{\mathbf{x}}} \min_{P_{\mathbf{s}|\mathbf{x}}\in\lambda_{\mathbf{s}|\mathbf{x}}} I(\mathbf{x};\mathbf{y}), \tag{19}$$

which can be viewed as a generalized sphere-packing bound. The mutual information is evaluated w.r.t.

$$P_{\mathbf{x},\mathbf{y}} = \left[P_{\mathbf{x}} P_{\mathbf{s}|\mathbf{x}} W_{\mathbf{y}|\mathbf{x},\mathbf{s}}\right]_{\mathbf{x},\mathbf{y}}.$$

Then

1. *(Achievability) For any $\delta > 0$ and sufficiently large n, there exists $\mathcal{C}$ of rate $C - \delta$ such that it can be $\mathcal{O}(1/\delta)$ list decoded.*
2. *(Converse) For any $\mathcal{C}$ of rate $C + \delta$, $\mathcal{C}$ is $2^{\Omega(n\delta)}$-list decodable.*

Proof. We follow the idea used in the proof of list decoding theorem 4 under the standard bit-flip model but conduct the calculations under our generalized setting [39].

1. (Achievability) Let $R = C - \delta$. Fix $P_{\mathbf{x}}^* \in \lambda_{\mathbf{x}}$ to be a maximizer of expression (19). Generate a random code by sampling 2^{nR} codewords independently and uniformly from $\mathcal{T}_{\underline{\mathbf{x}}}(P_{\mathbf{x}}^*)$. We will actually show that

   ▶ **Lemma 44.** *For any $\delta > 0$ and sufficiently large n, a random $P_{\mathbf{x}}^*$-constant composition code of rate $R = C - \delta$ as defined above is $\left(\frac{1+\log|\mathcal{Y}|}{\delta} - 1\right)$-list decodable with probability at least $1 - 2^{-n(1-R)}$.*

   For every $\underline{y} \in \mathcal{Y}^n$, define *conditional typical set*

   $$\mathcal{A}_{\underline{\mathbf{x}}|\underline{y}} := \left\{\underline{x} \in \mathcal{T}_{\underline{\mathbf{x}}}(P_{\mathbf{x}}^*) : \exists \underline{s} \in \Lambda_{\underline{\mathbf{s}}},\ \underline{y} = W(\underline{x},\underline{s})\right\}$$

   to be the set of all $\underline{x}$ of type $P_{\mathbf{x}}^*$ that can reach $\underline{y}$ via allowable $\underline{s} \in \Lambda_{\underline{\mathbf{s}}}$. Note that $\mathcal{A}_{\underline{\mathbf{x}}|\underline{y}}$ is precisely the list of codewords around $\underline{y}$ whose size we would like to bound. In favour of proceeding calculations, we write $\mathcal{A}_{\underline{\mathbf{x}}|\underline{y}}$ in terms of types and estimate its size. We say that a type $\tau_{\mathbf{x},\mathbf{s},\mathbf{y}} \in \mathcal{P}^{(n)}(\mathcal{X}\times\mathcal{S}\times\mathcal{Y})$ is *valid* if

   a. $[\tau_{\mathbf{x},\mathbf{s},\mathbf{y}}]_{\mathbf{x}} = P_{\mathbf{x}}^*$;
   b. $[\tau_{\mathbf{x},\mathbf{s},\mathbf{y}}]_{\mathbf{s}} \in \lambda_{\mathbf{s}}$;
   c. $\tau_{\mathbf{x},\mathbf{s},\mathbf{y}} = P_{\mathbf{x}}^* \tau_{\mathbf{s}|\mathbf{x}} W_{\mathbf{y}|\mathbf{x},\mathbf{s}}$.

[13] It can be easily seen that the set $\lambda_{\mathbf{s}|\mathbf{x}}$ is immediately specified given $P_{\mathbf{x}}$, $\lambda_{\mathbf{x}}$ and $\lambda_{\mathbf{s}}$.

Then it is not hard to see that

$$\mathcal{A}_{\underline{\mathbf{x}}|\underline{y}} = \bigcup_{\tau_{\mathbf{x},\mathbf{s},\mathbf{y}} \text{ valid}} \mathcal{T}_{\underline{\mathbf{x}}|\underline{y}}\left(\tau_{\mathbf{x}|\underline{y}}\right),$$

where $\tau_{\mathbf{x}|\underline{y}}$ is obtained from $\tau_{\mathbf{x},\mathbf{s},\mathbf{y}}$. Note that there is only a polynomial number of types and the volume of each $\mathcal{T}_{\underline{\mathbf{x}}|\underline{y}}\left(\tau_{\mathbf{x}|\underline{y}}\right)$ is dot equal to $2^{nH(\mathbf{x}|\mathbf{y})}$, where $H(\mathbf{x}|\mathbf{y})$ is evaluated w.r.t. $\left[\tau_{\mathbf{x},\mathbf{s},\underline{y}}\right]_{\mathbf{x},\underline{y}} = \tau_{\underline{y}}\tau_{\mathbf{x}|\underline{y}}$. Hence the volume of $\mathcal{A}_{\underline{\mathbf{x}}|\underline{y}}$ is

$$\begin{aligned}
\frac{1}{n}\log\left|\mathcal{A}_{\underline{\mathbf{x}}|\underline{y}}\right| &\xrightarrow{n\to\infty} \max_{\tau_{\mathbf{x},\mathbf{s},\mathbf{y}} \text{ valid}} H(\mathbf{x}|\mathbf{y}) && (20)\\
&= \max_{\substack{P^*_{\mathbf{x}}\tau_{\mathbf{s}|\mathbf{x}}W_{\mathbf{y}|\mathbf{x},\mathbf{s}}:\\ \left[P^*_{\mathbf{x}}\tau_{\mathbf{s}|\mathbf{x}}W_{\mathbf{y}|\mathbf{x},\mathbf{s}}\right]_{\mathbf{s}}\in\lambda_{\mathbf{s}}}} H(\mathbf{x}|\mathbf{y}) && (21)\\
&\to \max_{P_{\mathbf{s}|\mathbf{x}}\in\lambda_{\mathbf{s}|\mathbf{x}}} H(\mathbf{x}|\mathbf{y}). && (22)
\end{aligned}$$

In Eqn. (20) and (21), the conditional entropy is evaluated w.r.t. $\left[\tau_{\mathbf{x},\mathbf{s},\mathbf{y}}\right]_{\mathbf{x},\mathbf{y}}$ and $\left[P^*_{\mathbf{x}}\tau_{\mathbf{s}|\mathbf{x}}W_{\mathbf{y}|\mathbf{x},\mathbf{s}}\right]_{\mathbf{x},\mathbf{y}}$, respectively. In Eqn. (22), the conditional entropy is evaluated w.r.t. $\left[P^*_{\mathbf{x}}P_{\mathbf{s}|\mathbf{x}}W_{\mathbf{y}|\mathbf{x},\mathbf{s}}\right]_{\mathbf{x},\mathbf{y}}$. This equality holds in the limit as n approaches infinity since types are asymptotically dense in distributions. Note that $\mathcal{A}_{\underline{\mathbf{x}}|\underline{y}} \subset \mathcal{T}_{\underline{\mathbf{x}}}(P^*_{\mathbf{x}})$. We have that the probability q that a random codeword $\underline{\mathbf{x}}$ is able to result in $\underline{y}$ via some admissible $\underline{s} \in \Lambda_{\mathbf{s}}$ is

$$\begin{aligned}
\frac{1}{n}\log q \coloneqq& \frac{1}{n}\log\Pr\left[\underline{\mathbf{x}} \in \mathcal{A}_{\underline{\mathbf{x}}|\underline{y}}\right]\\
=& \frac{1}{n}\log\frac{\left|\mathcal{A}_{\underline{\mathbf{x}}|\underline{y}}\right|}{\left|\mathcal{T}_{\underline{\mathbf{x}}}(P^*_{\mathbf{x}})\right|} && (23)\\
\xrightarrow{n\to\infty}& \max_{P_{\mathbf{s}|\mathbf{x}}\in\lambda_{\mathbf{s}|\mathbf{x}}} H(\mathbf{x}|\mathbf{y}) - H(\mathbf{x}) && (24)\\
=& -\max_{P_{\mathbf{x}}\in\lambda_{\mathbf{x}}}\min_{P_{\mathbf{s}|\mathbf{x}}\in\lambda_{\mathbf{s}|\mathbf{x}}} I(\mathbf{x};\mathbf{y}) && (25)\\
=& -C.
\end{aligned}$$

Eqn. (23) follows since codewords are picked uniformly from $\mathcal{T}_{\underline{\mathbf{x}}}(P^*_{\mathbf{x}})$. Eqn. (24) is by Eqn. (22) and Eqn. (21). Eqn. (25) is by the choice of $P^*_{\mathbf{x}}$. The probability that there is a large list clustered around $\underline{y}$ is given by

$$\Pr_{\mathcal{C}}\left[\left|\mathcal{A}_{\underline{\mathbf{x}}|\underline{y}} \cap \mathcal{C}\right| \geqslant L\right] \doteq \sum_{i=L}^{2^{nR}} \binom{2^{nR}}{i} q^i (1-q)^{2^{nR}-i}.$$

Let S_i denote the summand

$$S_i \coloneqq \binom{2^{nR}}{i} q^i (1-q)^{2^{nR}-i}.$$

Note that

$$\begin{aligned}
\frac{S_i}{S_{i+1}} =& \frac{i+1}{2^{nR}-i}\frac{1-q}{q}\\
\geqslant& \frac{2}{2^{n(C-\delta)}}\frac{1-2^{-nC}}{2^{-nC}} && (26)
\end{aligned}$$

$$
\begin{aligned}
&=2\cdot\frac{1}{2}\cdot 2^{n\delta} && (27)\\
&>1,
\end{aligned}
$$

where Eqn. (26) follows since $i \geqslant L \geqslant 1$ and Eqn. (27) follows since $1-2^{-nC} \geqslant \frac{1}{2}$ when $n \geqslant \frac{1}{C}$. The largest summand is the first term. Therefore we can bound the error probability by replacing each term with the first one.

$$
\begin{aligned}
\Pr_{\mathcal{C}}\left[\left|\mathcal{A}_{\underline{\mathbf{x}}|\underline{y}}\cap\mathcal{C}\right|\geqslant L\right] \leqslant& 2^{nR}\binom{2^{nR}}{L}q^{L}(1-q)^{2^{nR}-L}\\
\leqslant& 2^{nR}2^{nRL}2^{-nCL}\\
=& 2^{-n((L+1)\delta-C)}.
\end{aligned}
$$

Finally taking a union bound over all $\underline{y}\in\mathcal{Y}^n$, we know that the probability of list decoding error is at most

$$
\begin{aligned}
\Pr\left[\exists\underline{y}\in\mathcal{Y}^n,\ \left|\mathcal{A}_{\underline{\mathbf{x}}|\underline{y}}\cap\mathcal{C}\right|\geqslant L\right] \leqslant& |\mathcal{Y}|^n\, 2^{-n((L+1)\delta-C)}\\
=& 2^{-n((L+1)\delta-C-\log|\mathcal{Y}|)},
\end{aligned}
$$

which is $2^{-\Omega(n)}$ if $L > \frac{1+\log|\mathcal{Y}|}{\delta}-1$. Specifically, taking $L=\frac{1+\log|\mathcal{Y}|}{\delta}$, we have that the list decoding error probability is at most $2^{-n(1+\delta-C)}=2^{-n(1-R)}$, as desired.

2. (Converse) Given any code $\mathcal{C}$ of rate $C+\delta$, choose the $\tau^*_{\mathbf{x}}\in\mathcal{P}^{(n)}(\mathcal{X})$ such that $|\mathcal{C}\cap\mathcal{T}_{\underline{\mathbf{x}}}(\tau^*_{\mathbf{x}})|$ is maximized. By Lemma 36, $R(\mathcal{C}')\asymp R(\mathcal{C})$. For this $\tau^*_{\mathbf{x}}$, choose legitimate $\tau^*_{\mathbf{s}|\mathbf{x}}\in\lambda_{\mathbf{s}|\mathbf{x}}$ such that

$$
\tau^*_{\mathbf{s}|\mathbf{x}} := \operatorname*{argmin}_{\tau_{\mathbf{s}|\mathbf{x}}\in\lambda_{\mathbf{s}|\mathbf{x}}} I(\mathbf{x};\mathbf{y}),
$$

where $I(\mathbf{x};\mathbf{y})$ is evaluated according to $\left[\tau^*_{\mathbf{x}}\tau_{\mathbf{s}|\mathbf{x}}W_{\mathbf{y}|\mathbf{x},\mathbf{s}}\right]_{\mathbf{x},\mathbf{y}}$. Now define $\tau^*_{\mathbf{x},\mathbf{s},\mathbf{y}}:=\tau^*_{\mathbf{x}}\tau^*_{\mathbf{s}|\mathbf{x}}W_{\mathbf{y}|\mathbf{x},\mathbf{s}}$, $\tau^*_{\mathbf{x},\mathbf{y}}:=\left[\tau^*_{\mathbf{x},\mathbf{s},\mathbf{y}}\right]_{\mathbf{x},\mathbf{y}}$ and $\tau^*_{\mathbf{y}}:=\left[\tau^*_{\mathbf{x},\mathbf{y}}\right]_{\mathbf{y}}$. Over the randomness of selecting $\underline{\mathbf{y}}$ uniformly from $\mathcal{T}_{\underline{\mathbf{y}}}\left(\tau^*_{\mathbf{y}}\right)$, the average number of codewords in $\mathcal{A}_{\underline{\mathbf{x}}|\underline{\mathbf{y}}}$ is dot equal to

$$
\begin{aligned}
\mathop{\mathbb{E}}_{\underline{\mathbf{y}}}\left[\left|\mathcal{A}_{\underline{\mathbf{x}}|\underline{\mathbf{y}}}\cap\mathcal{C}'\right|\right] &= \mathop{\mathbb{E}}_{\underline{\mathbf{y}}}\left[\sum_{\underline{x}\in\mathcal{C}'}\mathbb{1}_{\left\{\mathcal{A}_{\underline{\mathbf{x}}|\underline{\mathbf{y}}}\ni\underline{x}\right\}}\right]\\
&= \sum_{\underline{x}\in\mathcal{C}'}\Pr_{\underline{\mathbf{y}}}\left[\mathcal{A}_{\underline{\mathbf{x}}|\underline{\mathbf{y}}}\ni\underline{x}\right] && (28)\\
&= \sum_{\underline{x}\in\mathcal{C}'}\Pr_{\underline{\mathbf{y}}}\left[\mathcal{T}_{\underline{\mathbf{x}}|\underline{\mathbf{y}}}\left(\tau^*_{\mathbf{x}|\mathbf{y}}\right)\ni\underline{x}\right] && (29)\\
&= \sum_{\underline{x}\in\mathcal{C}'}\Pr_{\underline{\mathbf{y}}}\left[\tau_{\underline{x}|\underline{\mathbf{y}}}=\tau^*_{\mathbf{x}|\mathbf{y}}\right] && (30)\\
&= \sum_{\underline{x}\in\mathcal{C}'}\frac{1}{\left|\mathcal{T}_{\underline{\mathbf{y}}}\left(\tau^*_{\mathbf{y}}\right)\right|}\prod_{x\in\mathcal{X}}\binom{\tau^*_{\mathbf{x}}(x)n}{\tau^*_{\mathbf{y}}(1)n\cdot\tau^*_{\mathbf{x}|\mathbf{y}}(x|1),\cdots,\tau^*_{\mathbf{y}}(|\mathcal{Y}|)n\cdot\tau^*_{\mathbf{x}|\mathbf{y}}(x||\mathcal{Y}|)}. && (31)
\end{aligned}
$$

Eqn. (28) is linearity of expectation. Note that by our choice of $\tau^*_{\mathbf{x}}$ and $\tau^*_{\mathbf{s}|\mathbf{x}}$ (hence $\tau^*_{\mathbf{x},\mathbf{s},\mathbf{y}}$ and $\tau^*_{\mathbf{x},\mathbf{y}}$), $\mathcal{A}_{\underline{\mathbf{x}}|\underline{y}}$ only contains one type class $\mathcal{T}_{\underline{\mathbf{x}}|\underline{y}}\left(\tau^*_{\mathbf{x}|\mathbf{y}}\right)$, where $\tau^*_{\mathbf{x}|\mathbf{y}}$ is computed from $\tau^*_{\mathbf{x},\mathbf{y}}$. Eqn. (29) then follows. Eqn. (30) follows from the definition of type classes

(Definition 20). Eqn. (31) is by analyzing the sampling procedure from the first principle. The product is exactly, given $\underline{x} \in \mathcal{C}'$, the number of ways to pick $\underline{y}$ from $\mathcal{T}_{\underline{\mathbf{x}}|\underline{y}}\left(\tau^*_{\mathbf{y}}\right)$ such that $\tau_{\underline{x}|\underline{y}} = \tau^*_{\mathbf{x}|\mathbf{y}}$. We compute the exponent of the above expectation.

$$\begin{aligned}
\frac{1}{n}\log \mathop{\mathbb{E}}_{\underline{\mathbf{y}}}\left[\left|\mathcal{A}_{\underline{\mathbf{x}}|\underline{y}} \cap \mathcal{C}'\right|\right] \xrightarrow{n\to\infty} & R' - H\left(\tau^*_{\mathbf{y}}\right) \\
& + \sum_{x\in\mathcal{X}} \tau^*_{\mathbf{x}}(x) \sum_{y\in\mathcal{Y}} \frac{\tau^*_{\mathbf{y}}(y)\tau^*_{\mathbf{x}|\mathbf{y}}(x|y)}{\tau^*_{\mathbf{x}}(x)} \log \frac{\tau^*_{\mathbf{x}}(x)}{\tau^*_{\mathbf{y}}(y)\tau^*_{\mathbf{x}|\mathbf{y}}(x|y)} && (32)\\
=& R - H\left(\tau^*_{\mathbf{y}}\right) + \sum_{x\in\mathcal{X}} \tau^*_{\mathbf{x}}(x) H(\mathbf{y}|\mathbf{x}=x) && (33)\\
=& R - H(\mathbf{y}) + H(\mathbf{y}|\mathbf{x}) && (34)\\
=& R - I(\mathbf{x};\mathbf{y}) \\
\geqslant& R - C && (35)\\
=& \delta.
\end{aligned}$$

Since codewords in the subcode $\mathcal{C}'$ are $\tau^*_{\mathbf{x}}$-constant composition, the summand in Eqn. (31) is independent of particular choices of $\underline{x}$. Eqn. (32) then follows from Stirling's approximation (Lemma 10). In Eqn. (33), $H(\mathbf{y}|\mathbf{x}=x)$ is drawn according to the conditional type

$$\tau^*_{\mathbf{y}|\mathbf{x}}(\cdot|x) = \frac{\tau^*_{\mathbf{y}}(\cdot)\tau^*_{\mathbf{x}|\mathbf{y}}(x|\cdot)}{\tau^*_{\mathbf{x}}(x)}.$$

In Eqn. (34), we pass types to distributions by the fact that types are dense in distributions asymptotically in n. $H(\mathbf{y})$ and $H(\mathbf{y}|\mathbf{x})$ are evaluated using distribution $\left[\tau^*_{\mathbf{x}} P^*_{\mathbf{s}|\mathbf{x}} W_{\mathbf{y}|\mathbf{x},\mathbf{s}}\right]_{\mathbf{x},\mathbf{y}}$, where

$$P^*_{\mathbf{s}|\mathbf{x}} := \mathop{\operatorname{argmin}}_{P_{\mathbf{s}|\mathbf{x}}\in\lambda_{\mathbf{s}|\mathbf{x}}} I(\mathbf{x};\mathbf{y}),$$

and the objective function $I(\mathbf{x};\mathbf{y})$ is evaluated using $\left[\tau^*_{\mathbf{x}} P_{\mathbf{s}|\mathbf{x}} W_{\mathbf{y}|\mathbf{x}}\right]_{\mathbf{x},\mathbf{y}}$. Eqn. (35) is by the definition of C (Eqn. (19)). $\tau^*_{\mathbf{x}}$ always gives rise to mutual information no larger than the maximizer in C.

Therefore, we have shown that there exists at least one $\underline{y} \in \mathcal{Y}^n$ such that the corresponding list around $\underline{y}$ has size at least $2^{n(\delta - o(1))}$. ◀

11 List sizes of random codes

In this section, we show that, if L has order lower than $1/\delta$, then the code used in the proof of achievability (part 1) of the list decoding capacity theorem (Theorem 43) is list decodable with vanishingly small probability. This coupled with Theorem 43 implies that, for the majority (an exponentially close to 1 fraction) of random constant composition capacity-achieving (within gap δ) codes, $\Theta(1/\delta)$ is actually the *correct* order of their list sizes.

▶ **Corollary 45.** *For $\delta > 0$ and sufficiently large n, at least a $1 - 2^{-n(1-R)} - 2^{-n\delta + \frac{2}{\delta}\log\frac{1}{\delta}}$ fraction of $P^*_{\mathbf{x}}$-constant composition codes ($P^*_{\mathbf{x}}$ as defined in Eqn. (36)) of rate $R = C - \delta$ is $(L-1)$-list decodable, where $L = \Theta(1/\delta)$ lies within the following range*

$$L \in \left[\frac{C}{\delta}, \frac{1+\log|\mathcal{Y}|}{\delta}\right].$$

▶ **Theorem 46.** *For an adversarial channel* $\mathcal{A} = \left(\mathcal{X}, \lambda_{\mathbf{x}}, \mathcal{S}, \lambda_{\mathbf{s}}, , \mathcal{Y}, W_{\mathbf{y}|\mathbf{x},\mathbf{s}}\right)$*, take an optimizing input distribution* $P_{\mathbf{x}}$ *which attains the list decoding capacity* C*,*

$$P_{\mathbf{x}}^{*} := \underset{P_{\mathbf{x}} \in \lambda_{\mathbf{x}}}{\operatorname{argmax}} \min_{P_{\mathbf{s}|\mathbf{x}} \in \lambda_{\mathbf{s}|\mathbf{x}}} I(\mathbf{x}; \mathbf{y}). \tag{36}$$

For any $\delta > 0$*, for each sufficiently large blocklength* n*, sample a random code* $\mathcal{C}$ *of rate* $R = C - \delta$ *whose codewords are selected independently and uniformly from* $\mathcal{T}_{\underline{\mathbf{x}}}\left(P_{\mathbf{x}}^{*}\right)$*. Then* $\mathcal{C}$ *is* $< (C/\delta - 1)$*-list decodable with probability at most* $2^{-n\delta + \frac{2}{\delta}\log\frac{1}{\delta}}$*.*

The theorem follows from second moment calculations and generalizes similar theorems for list decodability of random error/erasure correction codes over $\mathbb{F}_q$ [26].

Proof. Let $M := 2^{nR}$. Define *typical set*

$$\mathcal{A}_{\underline{\mathbf{y}}} := \left\{W\left(\underline{x}, \underline{s}\right) \in \mathcal{Y}^n : \underline{x} \in \mathcal{T}_{\underline{\mathbf{x}}}\left(P_{\mathbf{x}}^{*}\right),\ \underline{s} \in \Lambda_{\underline{\mathbf{s}}}\right\}.$$

Put in the language of types, it can also be written as

$$\mathcal{A}_{\underline{\mathbf{y}}} = \bigcup_{\tau_{\mathbf{x},\mathbf{s},\mathbf{y}} \text{ valid}} \mathcal{T}_{\underline{\mathbf{y}}}\left(\tau_{\mathbf{y}}\right),$$

where $\tau_{\mathbf{y}} = \left[\tau_{\mathbf{x},\mathbf{s},\mathbf{y}}\right]_{\mathbf{y}}$. Define random variable W as a witness for non-list decodability of $\mathcal{C}$

$$W := \sum_{\underline{y} \in \mathcal{A}_{\underline{\mathbf{y}}}} \sum_{\{m_1, \cdots, m_L\} \in \binom{[M]}{L}} \mathbb{1}_{\left\{\left\{\underline{\mathbf{x}}_{m_1}, \cdots, \underline{\mathbf{x}}_{m_L}\right\} \subset \mathcal{A}_{\underline{\mathbf{x}}|\underline{y}}\right\}}.$$

Then by Chebyshev's inequality,

$$\begin{aligned}
\Pr\left[\mathcal{C} \text{ is } (L-1)\text{-list decodable}\right] &= \Pr\left[\bigcap_{\underline{y} \in \mathcal{Y}^n} \left\{\left|\mathcal{A}_{\underline{\mathbf{x}}|\underline{y}} \cap \mathcal{C}\right| < L\right\}\right] \qquad (37)\\
&\leqslant \Pr\left[\bigcap_{\underline{y} \in \mathcal{A}_{\underline{\mathbf{y}}}(P_{\mathbf{y}})} \left\{\left|\mathcal{A}_{\underline{\mathbf{x}}|\underline{y}} \cap \mathcal{C}\right| < L\right\}\right]\\
&= \Pr\left[\left(\bigcup_{\underline{y} \in \mathcal{A}_{\underline{\mathbf{y}}}} \left\{\left|\mathcal{A}_{\underline{\mathbf{x}}|\underline{y}} \cap \mathcal{C}\right| \geqslant L\right\}\right)^{c}\right]\\
&= \Pr\left[W = 0\right] \qquad (38)\\
&\leqslant \frac{\operatorname{Var}\left[W\right]}{\mathbb{E}\left[W\right]^2},
\end{aligned}$$

where Eqn. (38) follows since $W = 0$ if and only if none of the events $\left\{\left|\mathcal{A}_{\underline{\mathbf{x}}|\underline{y}} \cap \mathcal{C}\right| \geqslant L\right\}$ $(\underline{y} \in \mathcal{A}_{\underline{\mathbf{y}}})$ happens. In what follows, we will obtain an upper bound on $\operatorname{Var}\left[W\right]$ and a lower bound on $\mathbb{E}\left[W\right]$, and hence an upper bound on the probability (37).

Lower bounding $\mathbb{E}\left[W\right]$. We can get a lower bound on the expected value of W from a straightforward calculation.

$$\begin{aligned}
\mathbb{E}\left[W\right] &= \sum_{\underline{y} \in \mathcal{A}_{\underline{\mathbf{y}}}} \sum_{\{m_1, \cdots, m_L\} \in \binom{[M]}{L}} \Pr\left[\left\{\underline{\mathbf{x}}_{m_1}, \cdots, \underline{\mathbf{x}}_{m_L}\right\} \subset \mathcal{A}_{\underline{\mathbf{x}}|\underline{y}}\right]\\
&= \sum_{\underline{y} \in \mathcal{A}_{\underline{\mathbf{y}}}} \sum_{\{m_1, \cdots, m_L\} \in \binom{[M]}{L}} \Pr\left[\underline{\mathbf{x}} \in \mathcal{A}_{\underline{\mathbf{x}}|\underline{y}}\right]^{L} \qquad (39)
\end{aligned}$$

$$\begin{aligned}
&\doteq \left|\mathcal{A}_{\underline{\mathbf{y}}}\right| \binom{M}{L} 2^{-nCL} && (40)\\
&\geqslant \left|\mathcal{A}_{\underline{\mathbf{y}}}\right| \left(\frac{M}{L}\right)^{L} 2^{-nCL}\\
&= \left|\mathcal{A}_{\underline{\mathbf{y}}}\right| 2^{-n\delta L - L\log L}.
\end{aligned}$$

Eqn. (39) follows since codewords are independent. Eqn. (40) is by Eqn. (25).

Upper bounding Var $[W]$. Define, for any $\underline{y} \in \mathcal{Y}^n$ and $\mathcal{L} \in \binom{[M]}{L}$,

$$\begin{aligned}
\mathbb{I}\left(\underline{y}, \mathcal{L}\right) :=& \mathbb{1}_{\left\{\{\underline{\mathbf{x}}_m\}_{m\in\mathcal{L}} \subset \mathcal{A}_{\underline{\mathbf{x}}|\underline{y}}\right\}}\\
=& \prod_{m\in\mathcal{L}} \mathbb{1}_{\left\{\underline{\mathbf{x}}_m \in \mathcal{A}_{\underline{\mathbf{x}}|\underline{y}}\right\}},
\end{aligned}$$

as the indicator function of the event $\bigcap_{m\in\mathcal{L}} \left\{\underline{\mathbf{x}}_m \in \mathcal{A}_{\underline{\mathbf{x}}|\underline{y}}\right\}$ that the list $\mathcal{L}$ is L-confusable w.r.t. $\underline{y}$.

Now the variance of W can be upper bounded as follows.

$$\begin{aligned}
\mathrm{Var}\left[W\right] =& \mathbb{E}\left[W^2\right] - \mathbb{E}\left[W\right]^2 && (41)\\
=& \sum_{\underline{y}_1,\underline{y}_2 \in \mathcal{A}_{\underline{\mathbf{y}}}} \sum_{\mathcal{L}_1,\mathcal{L}_2 \in \binom{[M]}{L}} \mathbb{E}\left[\mathbb{I}\left(\underline{y}_1, \mathcal{L}_1\right)\mathbb{I}\left(\underline{y}_2, \mathcal{L}_2\right)\right] - \mathbb{E}\left[\mathbb{I}\left(\underline{y}_1, \mathcal{L}_1\right)\right]\mathbb{E}\left[\mathbb{I}\left(\underline{y}_2, \mathcal{L}_2\right)\right] && (42)\\
\leqslant& \sum_{\substack{\mathcal{L}_1,\mathcal{L}_2 \in \binom{[M]}{L}\\ \mathcal{L}_1\cap\mathcal{L}_2 \neq \varnothing}} \sum_{\underline{y}_1,\underline{y}_2 \in \mathcal{A}_{\underline{\mathbf{y}}}} \mathbb{E}\left[\mathbb{I}\left(\underline{y}_1, \mathcal{L}_1\right)\mathbb{I}\left(\underline{y}_2, \mathcal{L}_2\right)\right] && (43)\\
=& \left|\mathcal{A}_{\underline{\mathbf{y}}}\right|^2 \sum_{\ell=1}^{L} \sum_{|\mathcal{L}_1\cap\mathcal{L}_2|=\ell} \Pr_{\underline{\mathbf{y}}_1,\underline{\mathbf{y}}_2,\mathcal{C}}\left[\mathcal{E}\right]. && (44)
\end{aligned}$$

Eqn. (41) follows from the definition of variance and Eqn. (42) follows from linearity of expectation. Note that $\mathbb{I}\left(\underline{y}_1, \mathcal{L}_1\right)$ and $\underline{I}\left(\underline{y}_2, \mathcal{L}_2\right)$ are independent if and only if $\mathcal{L}_1 \cap \mathcal{L}_2 = \varnothing$. When they are independent, the first expectation factors and the summand vanishes. The inequality (43) follows by dropping the negative term in the summand. In Eqn. (44), we rewrite the summation by randomizing the centers $\underline{y}_1, \underline{y}_2$ of the lists $\mathcal{L}_1, \mathcal{L}_2$. The probability is taken over $\underline{\mathbf{y}}_1$ and $\underline{\mathbf{y}}_2$ chosen uniformly at random from $\mathcal{A}_{\underline{\mathbf{y}}}$ and over the random code sampling procedure. We use $\mathcal{E}$ to denote the event that the lists $\mathcal{L}_1$ and $\mathcal{L}_2$ are simultaneously L-confusable w.r.t. $\underline{\mathbf{y}}_1$ and $\underline{\mathbf{y}}_2$, respectively,

$$\mathcal{E} := \bigcap_{m_1\in\mathcal{L}_1} \left\{\underline{\mathbf{x}}_{m_1} \in \mathcal{A}_{\underline{\mathbf{x}}|\underline{\mathbf{y}}_1}\right\} \cap \bigcap_{m_2\in\mathcal{L}_2} \left\{\underline{\mathbf{x}}_{m_2} \in \mathcal{A}_{\underline{\mathbf{x}}|\underline{\mathbf{y}}_2}\right\}.$$

It then suffices to bound $\Pr\left[\mathcal{E}\right]$. To this end, first define conditional typical set, for $\underline{x} \in \mathcal{X}^n$,

$$\begin{aligned}
\mathcal{A}_{\underline{\mathbf{y}}|\underline{x}} :=& \left\{W\left(\underline{x}, \underline{s}\right) \in \mathcal{Y}^n \colon \underline{s} \in \Lambda_{\underline{\mathbf{s}}}\right\}\\
=& \bigcup_{\tau_{\mathbf{x},\mathbf{s},\mathbf{y}} \text{ valid}} \mathcal{T}_{\underline{\mathbf{y}}}\left(\tau_{\mathbf{y}|\underline{x}}\right),
\end{aligned}$$

where $\tau_{\mathbf{y}|\underline{x}}$ is computed from $\tau_{\mathbf{x},\mathbf{s},\mathbf{y}}$ and $\tau_{\underline{x}}$, $\tau_{\mathbf{y}|\underline{x}} = \left[\tau_{\mathbf{x},\mathbf{s},\mathbf{y}}\right]_{\mathbf{x},\mathbf{y}} / \tau_{\underline{x}}$. Then define the following events in favour of bounding $\Pr\left[\mathcal{E}\right]$.

$$\mathcal{E}_1 := \left\{\underline{\mathbf{y}}_1 \in \mathcal{A}_{\underline{\mathbf{y}}|\underline{\mathbf{x}}_m}\right\} \cap \left\{\underline{\mathbf{y}}_2 \in \mathcal{A}_{\underline{\mathbf{y}}|\underline{\mathbf{x}}_m}\right\},$$

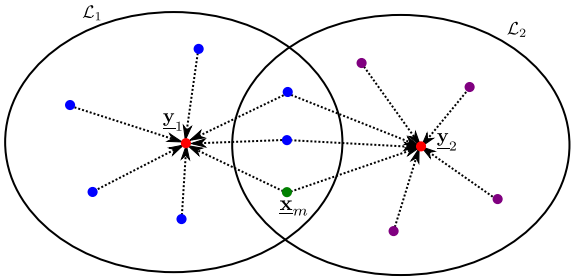


Figure 5 $\mathcal{E} \subset \mathcal{E}_1 \cap \mathcal{E}_2 \cap \mathcal{E}_3$. We upper bound $\Pr[\mathcal{E}]$ by neglecting the fact that codewords $\underline{\mathbf{x}}_i$ for $i \in (\mathcal{L}_1 \cap \mathcal{L}_2)\setminus\{m\}$ are simultaneously $\underline{\mathbf{y}}_1$-confusable and $\underline{\mathbf{y}}_2$-confusable, or equivalently, neglecting that $\underline{\mathbf{y}}_1, \underline{\mathbf{y}}_2$ should simultaneously belong to $\mathcal{A}_{\underline{\mathbf{y}}|\underline{\mathbf{x}}_{m'}}$ for all $m' \in \mathcal{L}_1 \cap \mathcal{L}_2$, not only the particular m we have chosen.

$$\mathcal{E}_2 := \bigcap_{m_1 \in \mathcal{L}_1 \setminus \{m\}} \left\{ \underline{\mathbf{x}}_{m_1} \in \mathcal{A}_{\underline{\mathbf{x}}|\underline{\mathbf{y}}_1} \right\},$$
$$\mathcal{E}_3 := \bigcap_{m_2 \in \mathcal{L}_2 \setminus \mathcal{L}_1} \left\{ \underline{\mathbf{x}}_{m_2} \in \mathcal{A}_{\underline{\mathbf{x}}|\underline{\mathbf{y}}_2} \right\},$$

where $m \in \mathcal{L}_1 \cap \mathcal{L}_2$ is any message that appears in both $\mathcal{L}_1$ and $\mathcal{L}_2$. It is easy to verify that $\mathcal{E} \subset \mathcal{E}_1 \cap \mathcal{E}_2 \cap \mathcal{E}_3$ (see Fig. 5). Note that $\mathcal{E}_2$ and $\mathcal{E}_3$ are independent conditioned on $\mathcal{E}_1$ since $\mathcal{L}_1\setminus\{m\}$ and $\mathcal{L}_2\setminus\mathcal{L}_1$ are disjoint. The probabilities of the above events can be computed precisely.

$$\Pr[\mathcal{E}_1] = \Pr\left[\underline{\mathbf{y}} \in \mathcal{A}_{\underline{\mathbf{y}}|\underline{\mathbf{x}}_m}\right]^2 \tag{45}$$
$$= \left(\frac{\left|\mathcal{A}_{\underline{\mathbf{y}}|\underline{\mathbf{x}}_m}\right|}{\left|\mathcal{A}_{\underline{\mathbf{y}}}\right|}\right)^2, \tag{46}$$

where Eqn. (45) is because $\underline{\mathbf{y}}_1$ and $\underline{\mathbf{y}}_2$ are independent, and Eqn. (46) follows since $\underline{\mathbf{y}}$ is chosen uniformly from $\mathcal{A}_{\underline{\mathbf{y}}}$. We now compute the exponent of $\Pr[\mathcal{E}]$.

$$\frac{1}{n}\log\left|\mathcal{A}_{\underline{\mathbf{y}}}\right| \xrightarrow{n\to\infty} \max_{\tau_{\mathbf{x},\mathbf{s},\mathbf{y}} \text{ valid}} H(\mathbf{y}) \tag{47}$$
$$= \max_{P_{\mathbf{s}|\mathbf{x}} \in \lambda_{\mathbf{s}|\mathbf{x}}} H(\mathbf{y}), \tag{48}$$

where in Eqn. (47) the entropy is computed w.r.t. $\tau_{\mathbf{y}} = [\tau_{\mathbf{x},\mathbf{s},\mathbf{y}}]_{\mathbf{y}}$; Eqn. (48) follows from similar calculations as done for $\mathcal{A}_{\mathbf{x}|\underline{y}}$ (Eqn. (20)) and the entropy is evaluated using $\left[P^*_{\mathbf{x}} P_{\mathbf{s}|\mathbf{x}} W_{\mathbf{y}|\mathbf{x},\mathbf{s}}\right]_{\mathbf{y}}$.

Similarly,

$$\frac{1}{n}\log\left|\mathcal{A}_{\underline{\mathbf{y}}|\underline{\mathbf{x}}_m}\right| \xrightarrow{n\to\infty} \max_{\tau_{\mathbf{x},\mathbf{s},\mathbf{y}} \text{ valid}} H(\mathbf{y}|\mathbf{x}) \tag{49}$$
$$= \max_{P_{\mathbf{s}|\mathbf{x}} \in \lambda_{\mathbf{s}|\mathbf{x}}} H(\mathbf{y}|\mathbf{x}), \tag{50}$$

where the conditional entropies in Eqn. (49) and (50) are evaluated w.r.t. $\tau_{\underline{x}}\tau_{\mathbf{y}|\underline{x}}$ and $\left[P^*_{\mathbf{x}} P_{\mathbf{s}|\mathbf{x}} W_{\mathbf{y}|\mathbf{x},\mathbf{s}}\right]_{\mathbf{x},\mathbf{y}}$ (since $\tau_{\underline{x}} \to P^*_{\mathbf{x}}$ as n approaches infinity), respectively. Continuing with

Eqn. (46), putting Eqn. (48) and Eqn. (50) together, we have

$$\begin{aligned}\Pr[\mathcal{E}_1] &\doteq \left(2^{n\max_{P_{\mathbf{s}|\mathbf{x}}\in\lambda_{\mathbf{s}|\mathbf{x}}} H(\mathbf{y}|\mathbf{x})-H(\mathbf{y})}\right)^2 \\ &= 2^{-2n\min_{P_{\mathbf{s}|\mathbf{x}}\in\lambda_{\mathbf{s}|\mathbf{x}}} I(\mathbf{x};\mathbf{y})} \\ &= 2^{-2nC}, \end{aligned} \tag{51}$$

where Eqn. (51) is by the choice of $P^*_{\mathbf{x}}$ (Eqn. (36)).

We also have

$$\Pr[\mathcal{E}_2|\mathcal{E}_1] = \Pr\left[\underline{\mathbf{x}} \in \mathcal{A}_{\underline{\mathbf{x}}|\underline{\mathbf{y}}_1} \middle| \mathcal{E}_1\right]^{L-1} \doteq 2^{-nC(L-1)}, \tag{52}$$

$$\Pr[\mathcal{E}_3|\mathcal{E}_1] = \Pr\left[\underline{\mathbf{x}} \in \mathcal{A}_{\underline{\mathbf{x}}|\underline{\mathbf{y}}_1} \middle| \mathcal{E}_1\right]^{L-\ell} \doteq 2^{-nC(L-\ell)}, \tag{53}$$

where Eqn. (52) and Eqn. (53) follow since $|\mathcal{L}_1| = |\mathcal{L}_2| = L$ and $|\mathcal{L}_1 \cap \mathcal{L}_2| = \ell$. We thus have, from Eqn. (51), (52) and (53), that

$$\begin{aligned}\Pr[\mathcal{E}] &\leqslant \Pr[\mathcal{E}_1 \cap \mathcal{E}_2 \cap \mathcal{E}_3] \\ &= \Pr[\mathcal{E}_1]\Pr[\mathcal{E}_2|\mathcal{E}_1]\Pr[\mathcal{E}_3|\mathcal{E}_1] \\ &\doteq 2^{-nC(2L-\ell+1)}. \end{aligned} \tag{54}$$

Note that the number of pairs of lists $\mathcal{L}_1$ and $\mathcal{L}_2$ with intersection size ℓ is

$$\begin{aligned}\binom{M}{\ell}\binom{M-\ell}{L-\ell}\binom{M-\ell}{L-\ell} &\leqslant M^\ell M^{L-\ell} M^{L-\ell} \\ &\leqslant M^{2L-\ell}. \end{aligned} \tag{55}$$

Therefore, the variance of W can be bounded as follows.

$$\mathrm{Var}[W] \leqslant \left|\mathcal{A}_{\underline{\mathbf{y}}}\right|^2 \sum_{1\leqslant\ell\leqslant L} M^{2L-\ell} 2^{-nC(2L-\ell+1)} \tag{56}$$

$$= \left|\mathcal{A}_{\underline{\mathbf{y}}}\right|^2 2^{-nC} \sum_{1\leqslant\ell\leqslant L} 2^{-n\delta(2L-\ell)} \tag{57}$$

$$\leqslant \left|\mathcal{A}_{\underline{\mathbf{y}}}\right|^2 2^{-nC} 2^{-n\delta(2L-\ell)+\log L}, \tag{58}$$

where Eqn. (56) is by Eqn. (44), (55) and (54); Eqn. (57) is by the definition of M and the choice of R; Eqn. (58) is by replacing each term with the largest one in the summation.

Putting them together.

$$\begin{aligned}\Pr[\mathcal{C} \text{ is } (L-1)\text{-list decodable}] &\leqslant \frac{\mathrm{Var}[W]}{\mathbb{E}[W]^2} \\ &\leqslant 2^{-nC+n\delta L+(2L+1)\log L}. \end{aligned}$$

The above probability vanishes in n if $L < C/\delta$. Say $L = C/\delta - 1$, then it is at most

$$2^{-n\delta+(2(C/\delta-1)+1)\log(C/\delta-1)} \leqslant 2^{-n\delta+\frac{2}{\delta}\log\frac{1}{\delta}}.$$

◀

12 Achievability

In this section, we are going to show, via concrete random code constructions, that as long as some completely positive $(P_{\mathbf{x}}, L)$-self-coupling of order L lies outside the order-L confusability set of the channel, the $(L-1)$-list decoding capacity is positive.

Let $\mathsf{CP}^{\otimes L}_{|\mathcal{X}|}(P_{\mathbf{x}}) := \mathsf{CP}^{\otimes L}_{|\mathcal{X}|} \cap \mathcal{J}^{\otimes L}(P_{\mathbf{x}})$.

▶ **Theorem 47** (Achievability). *For any given general adversarial channel $\mathcal{A} = (\mathcal{X}, \lambda_{\mathbf{x}}, \mathcal{S}, \lambda_{\mathbf{s}}, \mathcal{Y}, W_{\mathbf{y}|\mathbf{x},\mathbf{s}})$, its $(L-1)$-list decoding capacity is positive if there is a completely positive $(P_{\mathbf{x}}, L)$-self-coupling $P_{\mathbf{x}_1,\cdots,\mathbf{x}_L} \in \mathsf{CP}^{\otimes L}_{|\mathcal{X}|}(P_{\mathbf{x}})$ outside $\mathcal{K}^{\otimes L}(P_{\mathbf{x}})$ for some $P_{\mathbf{x}} \in \lambda_{\mathbf{x}}$.*

We first state a lemma concerning the rate of a random constant composition code.

▶ **Lemma 48** (Constant composition codes). *Let $\mathcal{C} = \{\underline{\mathbf{x}}_i\}_{i=1}^{2^{nR}}$ be a random code of rate R in which each codeword is selected according to product distribution $P_{\mathbf{x}}^{\otimes n}$ independently. Let $\mathcal{C}'$ be the $P_{\mathbf{x}}$-constant composition subcode of $\mathcal{C}$, $\mathcal{C}' = \mathcal{C} \cap \mathcal{T}_{\underline{\mathbf{x}}}(P_{\mathbf{x}})$. Then*

$$\Pr\left[\left|\mathcal{C}'\right| \notin (1 \pm 1/2)\frac{2^{nR}}{\nu(n)}\right] \leqslant 2\exp\left(-\frac{2^{nR}}{12\nu(n)}\right).$$

Proof. The lemma is a simple consequence of concentration of measure (Lemma 24).

$$\begin{aligned}\Pr\left[\left|\mathcal{C}'\right| \notin (1 \pm 1/2)\frac{2^{nR}}{\nu(n)}\right] &= \Pr\left[\sum_{i=1}^{2^{nR}} \mathbb{1}_{\{\tau_{\underline{\mathbf{x}}_i} = P_{\mathbf{x}}\}} \notin (1 \pm 1/2)\frac{2^{nR}}{\nu(n)}\right] \\ &\leqslant 2\exp\left(-\frac{(1/2)^2}{3}\mu\right) \qquad (59) \\ &= 2\exp\left(-\frac{2^{nR}}{12\nu(n)}\right).\end{aligned}$$

where in Eqn. (59), we note that

$$\begin{aligned}\mathbb{E}\left[\sum_{i=1}^{2^{nR}} \mathbb{1}_{\{\tau_{\underline{\mathbf{x}}_i} = P_{\mathbf{x}}\}}\right] &= 2^{nR}\Pr\left[\underline{\mathbf{x}} \in \mathcal{T}_{\underline{\mathbf{x}}}(P_{\mathbf{x}})\right] \\ &= \frac{2^{nR}}{\nu(n)} \\ &=: \mu.\end{aligned}$$

◀

12.1 Low rate codes

Let us proceed gently. We first show that a purely random code with each entry i.i.d. w.r.t. some distribution $P_{\mathbf{x}}$ is $(L-1)$-list decodable w.h.p. as long as $P_{\mathbf{x}}^{\otimes L}$ is not L-confusable.

▶ **Lemma 49.** *For any general adversarial channel $\mathcal{A} = (\mathcal{X}, \lambda_{\mathbf{x}}, \mathcal{S}, \lambda_{\mathbf{s}}, \mathcal{Y}, W_{\mathbf{y}|\mathbf{x},\mathbf{s}})$, if there exists a legitimate input distribution $P_{\mathbf{x}} \in \lambda_{\mathbf{x}}$ such that $P_{\mathbf{x}}^{\otimes L} \notin \mathcal{K}^{\otimes L}(P_{\mathbf{x}})$, then the $(L-1)$-list decoding capacity of $\mathcal{A}$ is positive.*

Proof. Let $M = 2^{nR}$ for some rate R to be specified momentarily. Sample a code $\mathcal{C} = \{\underline{\mathbf{x}}_1, \cdots, \underline{\mathbf{x}}_M\}$ where each $\underline{\mathbf{x}}_i \overset{\text{i.i.d.}}{\sim} P_{\mathbf{x}}^{\otimes n}$. The expected joint type $\tau_{\underline{\mathbf{x}}_{i_1},\cdots,\underline{\mathbf{x}}_{i_L}}$ $(1 \leqslant i_1 < \cdots < i_L \leqslant M)$ of any list $\underline{\mathbf{x}}_{i_1}, \cdots, \underline{\mathbf{x}}_{i_L}$ is $P_{\mathbf{x}}^{\otimes L}$. (See Fig. 6.)

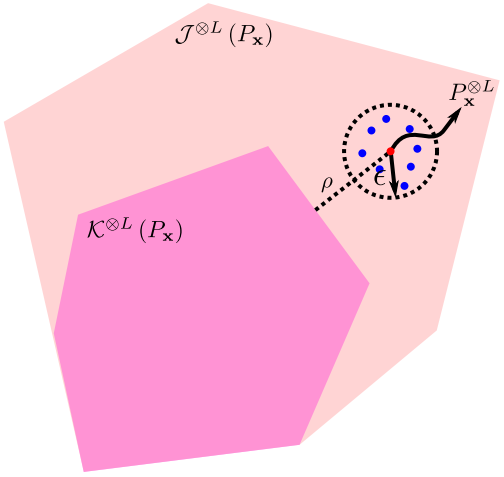


Figure 6 Low rate codes from product distribution. If the product distribution $P_{\mathbf{x}}^{\otimes L}$ is strictly separated away from $\mathcal{K}^{\otimes L}(P_{\mathbf{x}})$, then we could hope for a positive rate achieved by a random code with each entry sampled from $P_{\mathbf{x}}$. This is because w.h.p. the joint types of all (ordered) lists are contained in a $\|\cdot\|_{\text{mav}}$-ball which is completely outside the confusability set.

Let $\mathcal{C}' = \mathcal{C} \cap \mathcal{T}_{\underline{\mathbf{x}}}(P_{\mathbf{x}})$ be the $P_{\mathbf{x}}$-constant composition subcode of $\mathcal{C}$. Let

$$\rho \coloneqq \inf_{P_{\mathbf{x}_1,\cdots,\mathbf{x}_L} \in \mathcal{K}^{\otimes L}(P_{\mathbf{x}})} \left\| P_{\mathbf{x}}^{\otimes L} - P_{\mathbf{x}_1,\cdots,\mathbf{x}_L} \right\|_{\text{mav}}$$

be the max-absolute-value tensor distance from the product distribution to the confusability set. Let $R = \frac{\log e}{12}\frac{\rho^2}{L} - \delta$ for some small constant $\delta > 0$. We will show that

▶ **Lemma 50.** *The random $P_{\mathbf{x}}$-constant composition code $\mathcal{C}'$ as constructed above has rate $R = \frac{\log e}{12}\frac{\rho^2}{L} - \delta$ and is $(L-1)$-list decodable with probability at least $1 - 2\exp\left(-2^{nR}/\nu(n)\right) - 2^{-n\delta + L\log|\mathcal{X}|+1}$.*

Let $\epsilon \coloneqq \rho/2$. Define error events

$$\mathcal{E}_1 \coloneqq \left\{ \left|\mathcal{C}'\right| \notin (1 \pm 1/2)\frac{2^{nR}}{\nu(n)} \right\},$$
$$\mathcal{E}_2 \coloneqq \left\{ \mathcal{C}' \text{ is not } (L-1)\text{-list decodable} \right\}.$$

By Lemma 48,

$$\Pr\left[\mathcal{E}_1\right] \leqslant 2\exp\left(-\frac{2^{nR}}{\nu(n)}\right).$$

Hence the rate R' of $\mathcal{C}'$ is asymptotically equal to R w.h.p.

By Chernoff bound,

$$\begin{aligned}
&\Pr\left[\left\|\tau_{\underline{\mathbf{x}}_{i_1},\cdots,\underline{\mathbf{x}}_{i_L}} - P_{\mathbf{x}}^{\otimes L}\right\|_{\text{mav}} \geqslant \epsilon\right] \\
=&\Pr\left[\exists (x_1,\cdots,x_L) \in \mathcal{X}^L,\ \left|\tau_{\underline{\mathbf{x}}_{i_1},\cdots,\underline{\mathbf{x}}_{i_L}}(x_1,\cdots,x_L) - P_{\mathbf{x}}(x_1)\cdots P_{\mathbf{x}}(x_L)\right| \geqslant \epsilon\right] \qquad (60)\\
\leqslant&|\mathcal{X}|^L \Pr\left[\left|\sum_{j=1}^{n} \mathbb{1}_{\left\{\left(\underline{\mathbf{x}}_{i_1}(j),\cdots,\underline{\mathbf{x}}_{i_L}(j)\right)=(x_1,\cdots,x_L)\right\}} - nP_{\mathbf{x}}(x_1)\cdots P_{\mathbf{x}}(x_L)\right| \geqslant n\epsilon\right] \qquad (61)
\end{aligned}$$

$$
\begin{aligned}
&= |\mathcal{X}|^L \Pr\left[\sum_{j=1}^{n} \mathbb{1}_{\left\{\left(\underline{\mathbf{x}}_{i_1}(j),\cdots,\underline{\mathbf{x}}_{i_L}(j)\right)=(x_1,\cdots,x_L)\right\}} \notin \left(1 \pm \frac{n\epsilon}{\mu}\right)\mu\right] && (62)\\
&\leqslant |\mathcal{X}|^L \cdot 2\exp\left(-\frac{1}{3}\left(\frac{n\epsilon}{\mu}\right)^2 \mu\right) && (63)\\
&= |\mathcal{X}|^L \cdot 2\exp\left(-\frac{n\epsilon^2}{3P_{\mathbf{x}}^{\otimes L}(x_1,\cdots,x_L)}\right) && (64)\\
&\leqslant |\mathcal{X}|^L \cdot 2\exp\left(-\frac{n}{3}\left(\frac{\rho}{2}\right)^2\right) && (65)\\
&= 2\cdot|\mathcal{X}|^L \cdot \exp\left(-\frac{\rho^2}{12}n\right).
\end{aligned}
$$

Eqn. (60) follows from the definition of max-absolute-value norm. Eqn. (61) is obtained by taking a union bound and expanding the type using definition. In Eqn. (62), we define

$$\mu \coloneqq nP_{\mathbf{x}}^{\otimes L}(x_1,\cdots,x_L),$$

which equals

$$\mathbb{E}\left[\sum_{j=1}^{n} \mathbb{1}_{\left\{\left(\underline{\mathbf{x}}_{i_1}(j),\cdots,\underline{\mathbf{x}}_{i_L}(j)\right)=(x_1,\cdots,x_L)\right\}}\right].$$

Eqn. (63) is by Chernoff bound (Lemma 24). Eqn. (64) is by the definition of μ. Eqn. (65) is by the choice of ϵ and that $P_{\mathbf{x}}^{\otimes L}(x_1,\cdots,x_L) \leqslant 1$ for any $(x_1,\cdots,x_L) \in \mathcal{X}^L$. Taking a union bound over all lists $(i_1,\cdots,i_L) \in \binom{\mathcal{M}}{L}$,

$$
\begin{aligned}
&\Pr\left[\exists\,(i_1,\cdots,i_L) \in \binom{\mathcal{M}}{L},\ \left\|\tau_{\underline{\mathbf{x}}_{i_1},\cdots,\underline{\mathbf{x}}_{i_L}} - P_{\mathbf{x}}^{\otimes L}\right\|_\infty \geqslant \epsilon\right]\\
&\leqslant \binom{M}{L} 2\cdot|\mathcal{X}|^L \cdot \exp\left(-\frac{\rho^2}{12}n\right)\\
&\leqslant 2^{-n\left(\frac{\rho^2\log e}{12} - RL\right) + L\log|\mathcal{X}|+1}.
\end{aligned}
$$

We therefore get that $\mathcal{C}$ is $(L-1)$-list decodable with probability at least $1 - 2^{-n\delta + L\log|\mathcal{X}|+1}$ as long as

$$R = \frac{\log e}{12}\frac{\rho^2}{L} - \delta.$$

Overall, we have that

$$
\begin{aligned}
\Pr[\mathcal{E}_1 \cup \mathcal{E}_2] &\leqslant \Pr[\mathcal{E}_1] + \Pr[\mathcal{E}_2]\\
&\leqslant 2\exp\left(-\frac{2^{nR}}{\nu(n)}\right) + \Pr[\mathcal{C} \text{ is not } (L-1)\text{-list decodable}]\\
&\leqslant 2\exp\left(-\frac{2^{nR}}{\nu(n)}\right) + 2^{-n\delta + L\log|\mathcal{X}|+1}.
\end{aligned}
$$

◀

12.2 Random codes with expurgation

In the previous section, we only got an $(L-1)$-list decodable code of positive rate without making the effort to optimize the rate. In this section, we provide a lower bound on the $(L-1)$-list decoding capacity. It is achieved by a different code construction (random code with expurgation). However, we can only show the *existence* of such codes instead of showing that they attain the following bound w.h.p.

▶ **Lemma 51.** *The $(L-1)$-list decoding capacity of a channel $\mathcal{A}$ is at least*

$$C_{L-1} \geqslant \max_{P_{\mathbf{x}} \in \lambda_{\mathbf{x}}} \min_{P_{\mathbf{x}_1,\cdots,\mathbf{x}_L} \in \mathcal{K}^{\otimes L}(P_{\mathbf{x}})} \frac{1}{L-1} D\left(P_{\mathbf{x}_1,\cdots,\mathbf{x}_L} \| P_{\mathbf{x}}^{\otimes L}\right). \tag{66}$$

Proof. Fix any $P_{\mathbf{x}} \in \lambda_{\mathbf{x}}$ to be the maximizer of Eqn. (66). Let $M = 2^{nR}$ for some rate R to be determined. Generate a random code $\mathcal{C}$ of size $2M$ by sampling each entry of the codebook independently from $P_{\mathbf{x}}$.

For any $\underline{\mathbf{x}} \in \mathcal{C}$, by Lemma 22,

$$\Pr\left[\tau_{\underline{\mathbf{x}}} = P_{\mathbf{x}}\right] = 1/\nu(n).$$

Hence the expected number of codewords with type $P_{\mathbf{x}}$ is $2M/\nu(n)$.

For any $(\underline{\mathbf{x}}_1, \cdots, \underline{\mathbf{x}}_L) \in \binom{\mathcal{C}}{L}$,

$$\Pr\left[\tau_{\underline{\mathbf{x}}_1,\cdots,\underline{\mathbf{x}}_L} \in \mathcal{K}^{\otimes L}(P_{\mathbf{x}})\right] \doteq \sup_{P_{\mathbf{x}_1,\cdots,\mathbf{x}_L} \in \mathcal{K}^{\otimes L}(P_{\mathbf{x}})} 2^{-nD\left(P_{\mathbf{x}_1,\cdots,\mathbf{x}_L} \| P_{\mathbf{x}}^{\otimes L}\right)},$$

by Sanov's theorem 25. Let $P^* \in \mathcal{K}^{\otimes L}(P_{\mathbf{x}})$ be the extremizer for the above supremum. Hence the expected number of confusable lists is at most

$$\binom{2M}{L} 2^{-nD\left(P^* \| P_{\mathbf{x}}^{\otimes L}\right)} \leqslant (2M)^L 2^{-nD\left(P^* \| P_{\mathbf{x}}^{\otimes L}\right)}.$$

Pick M such that

$$(2M)^L 2^{-nD\left(P^* \| P_{\mathbf{x}}^{\otimes L}\right)} \leqslant M/\nu(n),$$

i.e.,

$$L + nRL - nD\left(P^* \| P_{\mathbf{x}}^{\otimes L}\right) \leqslant nR - \log \nu(n).$$

That is, R can be taken arbitrarily close to $\frac{1}{L-1} D\left(P^* \| P_{\mathbf{x}}^{\otimes L}\right)$.

$$\begin{aligned} R \leqslant & \frac{D\left(P^* \| P_{\mathbf{x}}^{\otimes L}\right)}{L-1} - \frac{\log \nu(n)}{(L-1)n} - \frac{L}{(L-1)n} \\ \overset{n\to\infty}{\to} & \frac{D\left(P^* \| P_{\mathbf{x}}^{\otimes L}\right)}{L-1}. \end{aligned}$$

Now, we remove all codewords of types different from $P_{\mathbf{x}}$. We also remove one codeword from each of the confusable lists. In expectation, this process reduces the size of the code by at most $2M - 2M/\nu(n)$ (due to the first expurgation) plus $(2M)^L 2^{-nD\left(P^* \| P_{\mathbf{x}}^{\otimes L}\right)} \leqslant M/\nu(n)$ (due to the second expurgation). After expurgation, we get an $(L-1)$-list decodable $P_{\mathbf{x}}$-constant composition code $\mathcal{C}'$ of size at least

$$2M - (2M/\nu(n) - 2M/\nu(n)) - M/\nu(n) = M/\nu(n).$$

The rate R' of $\mathcal{C}'$ is asymptotically the same as R.

$$\begin{aligned} R' = & R - \frac{\log \nu(n)}{n} \\ \overset{n\to\infty}{\to} & R. \end{aligned}$$

This finishes the proof. ◀

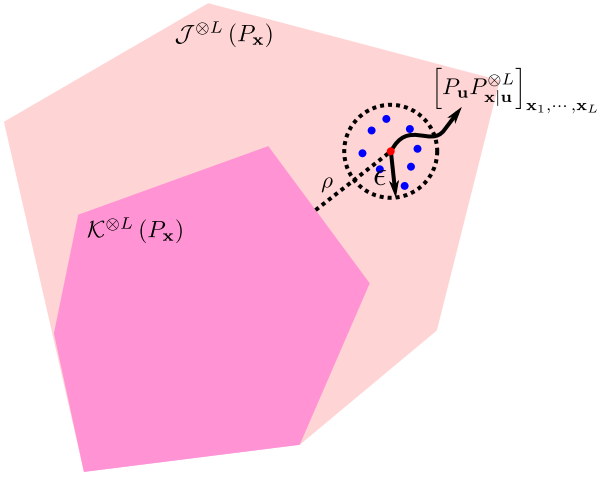


Figure 7 Low rate codes from CP distribution. If there is a CP distribution strictly outside $\mathcal{K}^{\otimes L}(P_{\mathbf{x}})$, then we can get a positive rate from random code using time-sharing. The only variation is that we divide codebook into chunks according to $P_{\mathbf{u}}$ and construct random codes of shorter length for each chunk u using distribution $P_{\mathbf{x}|\mathbf{u}=u}$.

12.3 Cloud codes

▶ **Lemma 52.** *If there is a $(P_{\mathbf{x}}, L)$-self-coupling ($P_{\mathbf{x}} \in \lambda_{\mathbf{x}}$) $P_{\mathbf{x}_1,\cdots,\mathbf{x}_L} \in \mathcal{J}^{\otimes L}(P_{\mathbf{x}}) \backslash \mathcal{K}^{\otimes L}(P_{\mathbf{x}})$ which can be decomposed into*

$$\begin{aligned} & P_{\mathbf{x}_1,\cdots,\mathbf{x}_L}(\underline{x}_1,\cdots,\underline{x}_L) \\ =& \sum_{u\in\mathcal{U}} P_{\mathbf{u}}(u)\, P_{\mathbf{x}|\mathbf{u}}^{\otimes L}(\underline{x}_1,\cdots,\underline{x}_L|u) \\ =& \sum_{u\in\mathcal{U}} P_{\mathbf{u}}(u) \prod_{i=1}^{L} P_{\mathbf{x}|\mathbf{u}}(\underline{x}_i|u)\,. \end{aligned}$$

for some distributions $P_{\mathbf{u}} \in \Delta(\mathcal{U})$ of finite support $|\mathcal{U}|$ and $P_{\mathbf{x}|\mathbf{u}} \in \Delta(\mathcal{X}|\mathcal{U})$ (see Fig. 7), then there exist positive rate $(L-1)$-list decodable codes.

Proof. The proof follows from a time-sharing argument combined with the previous low rate code construction (Lemma 49).

Fix R to be determined later. Sample 2^{nR} codewords in $\mathcal{C}$ independently from the following distribution. Divide each length-n codeword into $|\mathcal{U}|$ chunks $1,\cdots,|\mathcal{U}|$. For the u-th ($u \in \mathcal{U}$) chunk, sample $P_{\mathbf{u}}(u)n$ components in the chunk independently using distribution $P_{\mathbf{x}|\mathbf{u}=u}$. Let $P_{\mathbf{u},\mathbf{x}} = P_{\mathbf{u}}P_{\mathbf{x}|\mathbf{u}}$ and $P_{\mathbf{x}} = [P_{\mathbf{u},\mathbf{x}}]_{\mathbf{x}}$. Let $\mathcal{C}'$ be all codewords in $\mathcal{C}$ of type $P_{\mathbf{x}}$. (See Fig. 8.) Define

$$\rho := \inf_{P'_{\mathbf{x}_1,\cdots,\mathbf{x}_L} \in \mathcal{K}^{\otimes L}(P_{\mathbf{x}})} \left\| P_{\mathbf{x}_1,\cdots,\mathbf{x}_L} - P'_{\mathbf{x},\cdots,\mathbf{x}_L} \right\|_{\text{mav}}.$$

Let

$$u^* := \operatorname*{argmin}_{u\in\mathcal{U}} P_{\mathbf{u}}(u).$$

Note that $P_{\mathbf{u}}(u^*) > 0$ since $|\mathcal{U}|$ is the *support* of $P_{\mathbf{u}}$. Let $R = \frac{P_{\mathbf{u}}(u^*)\log e}{12}\frac{\rho^2}{L} - \delta$. We will show that

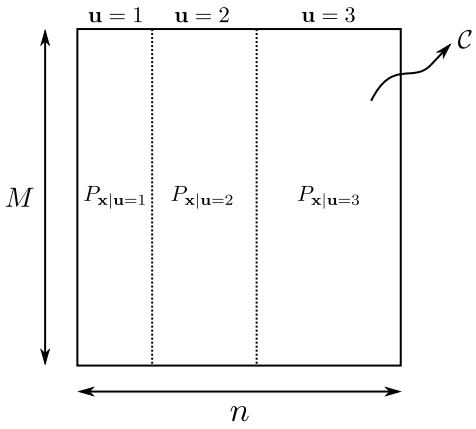


Figure 8 An example of cloud code construction in which $\mathcal{U} = \{1, 2, 3\}$. The codebook is divided into 3 chunks and symbols in the i-th chunk are sampled independently from $P_{\mathbf{x}|\mathbf{u}=i}$ $(i = 1, 2, 3)$.

▶ **Lemma 53.** *A random $P_{\mathbf{x}}$-constant composition cloud code as constructed above has rate $R = \frac{P_{\mathbf{u}}(u^*)\log e}{12}\frac{\rho^2}{L} - \delta$ and is $(L-1)$-list decodable with probability at least*

$$1 - 2\exp\left(-\frac{2^{nR}}{12\prod_{u\in\mathcal{U}}\nu(P_{\mathbf{u}}(u)n)}\right) - 2^{-n\delta + L\log|\mathcal{X}| + \log|\mathcal{U}| + 1}.$$

We write a length-n codeword as the concatenation of $|\mathcal{U}|$ chunks,

$$\underline{\mathbf{x}} = \left(\underline{\mathbf{x}}^{(1)}, \cdots, \underline{\mathbf{x}}^{(|\mathcal{U}|)}\right).$$

First we argue that w.h.p. the code $\mathcal{C}$ is almost $P_{\mathbf{x}}$-constant composition. The expected size of $\mathcal{C}'$ is

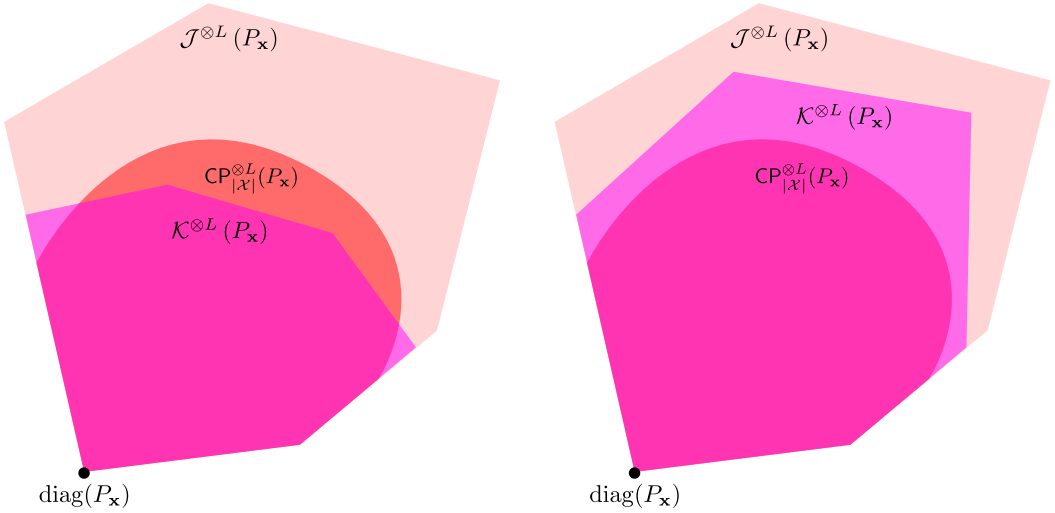
$$\begin{aligned}
\mathbb{E}\left[\left|\mathcal{C}'\right|\right] =& \mathbb{E}\left[\left|\mathcal{C} \cap \mathcal{T}_{\underline{\mathbf{x}}}(P_{\mathbf{x}|\mathbf{u}})\right|\right] \\
=& \sum_{i\in[M]} \Pr\left[\underline{\mathbf{x}}_i \in \mathcal{T}_{\underline{\mathbf{x}}}(P_{\mathbf{x}|\mathbf{u}})\right] && (67)\\
=& \sum_{i\in[M]} \Pr\left[\bigcap_{u\in\mathcal{U}} \left\{\underline{\mathbf{x}}_i^{(u)} \in \mathcal{T}_{\underline{\mathbf{x}}^{(u)}}(P_{\mathbf{x}|\mathbf{u}=u})\right\}\right] \\
=& \sum_{i\in[M]} \prod_{u\in\mathcal{U}} \Pr\left[\underline{\mathbf{x}}^{(u)} \in \mathcal{T}_{\underline{\mathbf{x}}^{(u)}}(P_{\mathbf{x}|\mathbf{u}=u})\right] && (68)\\
\asymp& M\prod_{u\in\mathcal{U}} \nu(P_{\mathbf{u}}(u)n)^{-1}, && (69)
\end{aligned}$$

where Eqn. (67) is by linearity of expectation; Eqn. (68) follows since different chunks are independent; Eqn. (69) follows from Lemma 22. Then by Lemma 48

$$\Pr\left[\left|\mathcal{C}'\right| \notin (1 \pm 1/2)\mathbb{E}\left[\left|\mathcal{C}'\right|\right]\right] \leqslant 2\exp\left(-\frac{2^{nR}}{12\prod_{u\in\mathcal{U}}\nu(P_{\mathbf{u}}(u)n)}\right).$$

Secondly, for any list $1 \leqslant i_1 < \cdots < i_L \leqslant M$ of distinct ordered messages,

$$\begin{aligned}
\Pr\left[\exists u \in \mathcal{U},\ \left\|\tau_{\underline{\mathbf{x}}_{i_1}^{(u)},\cdots,\underline{\mathbf{x}}_{i_L}^{(u)}} - P_{\mathbf{x}|\mathbf{u}=u}^{\otimes L}\right\|_{\max} \geqslant \epsilon\right] &\leqslant \sum_{u\in\mathcal{U}} 2\cdot|\mathcal{X}|^L \cdot \exp\left(-\frac{\rho^2}{12}nP_{\mathbf{u}}(u)\right) && (70)\\
&\leqslant 2|\mathcal{U}||\mathcal{X}|^L \exp\left(-\frac{\rho^2}{12}nP_{\mathbf{u}}(u^*)\right), && (71)
\end{aligned}$$



(a) "Below Plotkin point", positive $(L-1)$-list decoding rate is possible. In this case, for some input distribution $P_{\mathbf{x}} \in \lambda_{\mathbf{x}}$, the slice of $P_{\mathbf{x}}$-self-coupling CP tensors is not entirely contained in the confusability set $\mathcal{K}^{\otimes L}(P_{\mathbf{x}})$.

(b) "Above Plotkin point", no positive rate for $(L-1)$-list decoding is achievable. In this case, for every input distribution $P_{\mathbf{x}} \in \lambda_{\mathbf{x}}$, the slice of $P_{\mathbf{x}}$-self-coupling CP tensors is entirely contained in the confusability set $\mathcal{K}^{\otimes L}(P_{\mathbf{x}})$.

Figure 9 A characterization of when positive rate generalized list decodable codes exist.

where the first inequality (70) follows from a union bound and same calculations as in Lemma 49. The second inequality (71) follows from the definition of u^*.

Finally, by taking another union bound over lists $\mathcal{L} \in \binom{[M]}{L}$, we get

$$\Pr\left[\exists (i_1,\cdots,i_L) \in \binom{\mathcal{M}}{L},\ \exists u \in \mathcal{U},\ \left\| \tau_{\underline{\mathbf{x}}^{(u)}_{i_1},\cdots,\underline{\mathbf{x}}^{(u)}_{i_L}} - P^{\otimes L}_{\mathbf{x}|\mathbf{u}=u} \right\|_{\mathrm{mav}} \geqslant \epsilon \right]$$
$$\leqslant 2^{-n\left(\frac{\rho^2 \log e P_{\mathbf{u}}(u^*)}{12} - RL\right) + L\log|\mathcal{X}| + \log|\mathcal{U}| + 1}.$$

Therefore, we have that the probability that the random $P_{\mathbf{x}}$-constant composition cloud code $\mathcal{C}'$ constructed above has rate $R = \frac{P_{\mathbf{u}}(u^*)\log e}{12}\frac{\rho^2}{L} - \delta$ and is $(L-1)$-list decodable with probability at least

$$1 - 2\exp\left(-\frac{2^{nR}}{12\prod_{u\in\mathcal{U}}\nu(P_{\mathbf{u}}(u)n)}\right) - 2^{-n\delta + L\log|\mathcal{X}| + \log|\mathcal{U}| + 1},$$

which completes the proof. ◀

The above lemma apparently implies Theorem 47.

13 Converse

Let $\mathsf{CP}^{\otimes L}_{|\mathcal{X}|}(P_{\mathbf{x}}) := \mathsf{CP}^{\otimes L}_{|\mathcal{X}|} \cap \mathcal{J}^{\otimes L}(P_{\mathbf{x}})$ and $\mathsf{Sym}^{\otimes L}_{|\mathcal{X}|}(P_{\mathbf{x}}) := \mathsf{Sym}^{\otimes L}_{|\mathcal{X}|} \cap \mathcal{J}^{\otimes L}(P_{\mathbf{x}})$.

We have shown in the previous section that if $\mathsf{CP}^{\otimes L}_{|\mathcal{X}|}(P_{\mathbf{x}}) \cap \mathcal{K}^{\otimes L}(P_{\mathbf{x}})^c \neq \varnothing$, then the $(L-1)$-list decoding capacity is positive. In this section we are going to prove the converse. That is, such a condition is also necessary for positive rate being possible. Indeed, we will show that

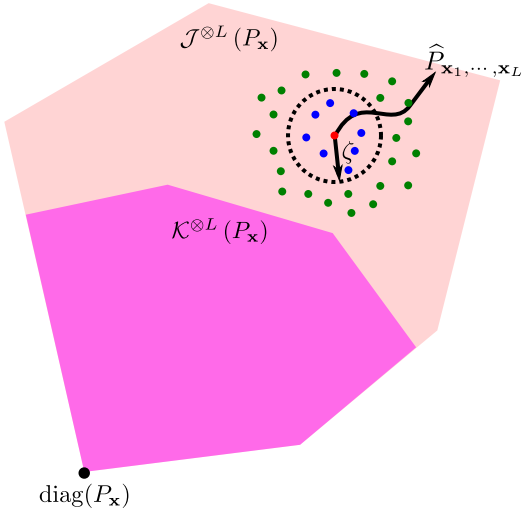


Figure 10 Equicoupled subcode extraction using hypergraph Ramsey's theorem. The union of green and blue dots represents the set of all joint types of ordered L-lists in $\mathcal{C}$. The blue dots correspond to joint types of its subcode $\mathcal{C}'$. (Note that they are all non-confusable.) They are clustered within a small ball (w.r.t. sum-absolute-value norm) centered at some distribution $\widehat{P}_{\mathbf{x}_1,\cdots,\mathbf{x}_L}$. Since the hypergraph Ramsey number is finite, there exists such $\mathcal{C}'$ which is suitably large.

▶ **Theorem 54** (Converse). *Given a general adversarial channel* $\mathcal{A} = \left(\mathcal{X}, \lambda_{\mathbf{x}}, \mathcal{S}, \lambda_{\mathbf{s}}, \mathcal{Y}, W_{\mathbf{y}|\mathbf{x}}\right)$, *if for every admissible input distribution* $P_{\mathbf{x}} \in \lambda_{\mathbf{x}}$, $\mathsf{CP}^{\otimes L}_{|\mathcal{X}|}(P_{\mathbf{x}}) \subseteq \mathcal{K}^{\otimes L}(P_{\mathbf{x}})$, *then the* $(L-1)$*-list decoding capacity of* $\mathcal{A}$ *is zero.*

13.1 Equicoupled subcode extraction

▶ **Definition 55** (Equicoupledness and ϵ-equicoupledness). *A code* $\mathcal{C}$ *is said to be* $P_{\mathbf{x}_1,\cdots,\mathbf{x}_L}$-equicoupled *if for all* ordered *lists* $(\underline{x}_{i_1},\cdots,\underline{x}_{i_L}) \in \binom{\mathcal{C}}{L}$ *where* $1 \leqslant i_1 < \cdots < i_L \leqslant |\mathcal{C}|$, $\tau_{\underline{x}_{i_1},\cdots,\underline{x}_{i_L}} = P_{\mathbf{x}_1,\cdots,\mathbf{x}_L}$. *A code* $\mathcal{C}$ *is said to be* $(\zeta, P_{\mathbf{x}_1,\cdots,\mathbf{x}_L})$-equicoupled *if for all ordered lists* $(\underline{x}_{i_1},\cdots,\underline{x}_{i_L}) \in \binom{\mathcal{C}}{L}$, *where* $1 \leqslant i_1 < \cdots < i_L \leqslant |\mathcal{C}|$, $\left\|\tau_{\underline{x}_1,\cdots,\underline{x}_{i_L}} - P_{\mathbf{x}_1,\cdots,\mathbf{x}_L}\right\|_{\mathrm{sav}} \leqslant \epsilon$.

▶ Remark 56. The above definition can also be overloaded for sequences of random variables or their joint distributions. We say a sequence of random variables $\mathbf{w}_1,\cdots,\mathbf{w}_M$ or the joint distribution $P_{\mathbf{w}_1,\cdots,\mathbf{w}_M}$ is $P_{\mathbf{x}_1,\cdots,\mathbf{x}_L}$-equicoupled (or $(\zeta, P_{\mathbf{x}_1,\cdots,\mathbf{x}_L})$-equicoupled) if every order-L marginal $P_{\mathbf{w}_{i_1},\cdots,\mathbf{w}_{i_L}}$ $(1 \leqslant i_1 < \cdots < i_L \leqslant M)$ equals (or is ζ-close to in $\|\cdot\|_{\mathrm{sav}}$) $P_{\mathbf{x}_1,\cdots,\mathbf{x}_L}$.

Using the hypergraph Ramsey's theorem, we first show that any infinite sequence of codes of positive rate has an infinite sequence of subcodes which are ζ-equicoupled.

▶ **Lemma 57** (Equicoupled subcode extraction). *For any infinite sequence of codes* $\{\mathcal{C}_i\}_{i\geqslant 1}$ *of blocklengths* n_i*'s and positive rate, where* $\{n_i\}_{i\geqslant 1}$ *is an infinite increasing integer sequence, for any* $\zeta > 0$ *and any* $M \in \mathbb{Z}_{>0}$, *there is an* $N \in \mathbb{Z}_{>0}$ *such that if* $|\mathcal{C}_i| \geqslant N$ *then* $\mathcal{C}'$ *contains a subcode* $\mathcal{C}'_i$ *satisfying that*

- $|\mathcal{C}'_i| \geqslant M$;
- $\mathcal{C}'_i$ *is* $(\zeta, P_{\mathbf{x}_1,\cdots,\mathbf{x}_L})$*-equicoupled for some* $P_{\mathbf{x}_1,\cdots,\mathbf{x}_L}$.

See Fig. 10.

Again, this lemma is a consequence of the hypergraph Ramsey's theorem. Let us denote by $R_c^{(m)}(n_1, \cdots, n_c)$ the smallest integer n such that the complete m-uniform hypergraph on n vertices with any c-colouring of hyperedges contains at least one of a clique of colour 1 and size n_1, ..., a clique of colour c and size n_c. It is known that $R_c^{(m)}(n_1, \cdots, n_c)$ is finite (Lemma 101), i.e., independent of the size n of the hypergraph.

Proof of Lemma 57. Recall that we assume $\mathsf{CP}_{|\mathcal{X}|}^{\otimes L}(P_{\mathbf{x}}) \cap \mathcal{K}^{\otimes L}(P_{\mathbf{x}})^c = \varnothing$. Let ρ be the gap between $\mathsf{CP}_{|\mathcal{X}|}^{\otimes L}(P_{\mathbf{x}})$ and $\mathcal{K}^{\otimes L}(P_{\mathbf{x}})$,

$$\rho := \inf_{\substack{P \in \mathsf{CP}_{|\mathcal{X}|}^{\otimes L}(P_{\mathbf{x}}) \\ P' \in \mathcal{J}^{\otimes L}(P_{\mathbf{x}}) \backslash \mathcal{K}^{\otimes L}(P_{\mathbf{x}})}} \left\| P - P' \right\|_{\mathrm{sav}}.$$

▶ **Definition 58** (ϵ-net)**.** *For a metric space $(\mathcal{X}, d)$, an ϵ-net $\mathcal{N} \subset \mathcal{X}$ is a subset which is a discrete ϵ-approximation of $\mathcal{X}$ in the sense that for any $x \in \mathcal{X}$, there is an $x' \in \mathcal{N}$ such that $d(x, x') \leqslant \epsilon$.*

We claim that

▶ **Lemma 59** (Bound on size of ϵ-net)**.** *There is an ϵ-net $\mathcal{N}$ of $\mathcal{J}^{\otimes L}(P_{\mathbf{x}}) \backslash \mathcal{K}^{\otimes L}(P_{\mathbf{x}})$ equipped with ℓ^1 metric of size at most $\left(\frac{|\mathcal{X}|^L}{2\epsilon} + 1\right)^{|\mathcal{X}|^L}$.*

Proof. The following construction is by no means optimal, but its size has a *finite* upper bound which is enough for our purposes. Indeed, it suffices to take $\mathcal{N}$ to be the coordinate-quantization net of $\mathcal{J}^{\otimes L}(P_{\mathbf{x}}) \backslash \mathcal{K}^{\otimes L}(P_{\mathbf{x}})$. Note that for any $P \in \mathcal{J}^{\otimes L}(P_{\mathbf{x}})$, each entry of P lies in $[0, 1]$. Take $\delta := \frac{2\epsilon}{|\mathcal{X}|^L}$. Divide $[0, 1]$ into sub-intervals of length δ (possibly except the last sub-interval that may have length less than δ). For each entry of P, there are at most $\frac{1}{\delta} + 1$ sub-intervals. Quantize each component of P to the nearest middle point of these sub-intervals. The set of all representatives whose components take values from the set of middle points of the sub-intervals form a net $\mathcal{N}$. In total, there are at most $\left(\frac{1}{\delta} + 1\right)^{|\mathcal{X}|^L}$ such representatives. For any $P \in \mathcal{J}^{\otimes L}(P_{\mathbf{x}}) \backslash \mathcal{K}^{\otimes L}(P_{\mathbf{x}})$, let $Q_{\mathcal{N}}(P)$ denote the quantization of P using $\mathcal{N}$, i.e.,

$$Q_{\mathcal{N}}(P) := \operatorname*{argmin}_{P' \in \mathcal{N}} \left\| P - P' \right\|_{\mathrm{sav}}.$$

The quantization error is at most

$$\begin{aligned} \left\| P - Q_{\mathcal{N}}(P) \right\|_{\mathrm{sav}} &\leqslant \sum_{(x_1, \cdots, x_L) \in \mathcal{X}^L} \left| P(x_1, \cdots, x_L) - Q_{\mathcal{N}}(P)(x_1, \cdots, x_L) \right| \\ &\leqslant |\mathcal{X}|^L \frac{\delta}{2} \\ &\leqslant \epsilon. \end{aligned}$$

We thus have shown that $\mathcal{N}$ constructed as above is an ϵ-quantizer of small cardinality. ◀

Let

$$\lambda := - \sup_{\widehat{P} \in (\mathcal{J}^{\otimes L}(P_{\mathbf{x}}) \backslash \mathcal{K}^{\otimes L}(P_{\mathbf{x}})) \cap \mathsf{Sym}_{|\mathcal{X}|}^{\otimes L}(P_{\mathbf{x}})} \inf_{Q \in \mathsf{coP}_{|\mathcal{X}|}^{\otimes L}} \left\langle \widehat{P}, Q \right\rangle. \tag{72}$$

We know that CP cone and coP cone are dual (Theorem 96) in the space of symmetric tensor cone. Thus, for any non-CP *symmetric* tensor $\widehat{P} \in \mathsf{Sym}^{\otimes L}_{|\mathcal{X}|}(P_{\mathbf{x}}) \backslash \mathsf{CP}^{\otimes L}_{|\mathcal{X}|}(P_{\mathbf{x}})$, there must be a witness Q with strictly negative inner product with $\widehat{P}$. The infimum

$$\inf_{Q \in \mathsf{coP}^{\otimes L}_{|\mathcal{X}|}} \left\langle \widehat{P}, Q \right\rangle < 0.$$

λ is the absolute value of the smallest inner product among all symmetric non-CP tensors. We know that $\lambda > 0$, since $\mathsf{CP}^{\otimes L}_{|\mathcal{X}|}(P_{\mathbf{x}})$ is *strictly* contained in $\mathcal{K}^{\otimes L}(P_{\mathbf{x}})$.

Let

$$\zeta := \frac{1}{2} \min \left\{ \rho, \frac{\lambda}{|\mathcal{X}|^L} \right\}. \tag{73}$$

Take a ζ-net of $\left(\Delta\left(\mathcal{X}^L\right), \ell^1\right)$ as constructed in Lemma 59. Such a net has cardinality at most $K := \left(\frac{|\mathcal{X}|^L}{\rho} + 1\right)^{|\mathcal{X}|^L}$.

Build an L-uniform complete hypergraph $\mathcal{H} = (\mathcal{C}, \mathcal{E})$ on $\mathcal{C}$. The vertices of $\mathcal{H}$ are codewords in $\mathcal{C}$. For every tuple $\left(\underline{x}_{i_1}, \cdots, \underline{x}_{i_L}\right) \in \binom{\mathcal{C}}{L}$ (where the indices $1 \leqslant i_1 < \cdots < i_L \leqslant |\mathcal{C}|$ are sorted in ascending order) of distinct codewords, there is a hyperedge connecting them. There are totally $\binom{|\mathcal{C}|}{L}$ hyperedges in $\mathcal{E}$. We now label hyperedges using distributions in $\mathcal{N}$. For each hyperedge $\left(\underline{x}_{i_1}, \cdots, \underline{x}_{i_L}\right) \in \mathcal{E}$, label it using the unique element $Q_{\mathcal{N}}\left(\tau_{\underline{x}_{i_1}, \cdots, \underline{x}_{i_L}}\right)$ from $\mathcal{N}$. This can be viewed as an edge colouring of $\mathcal{H}$ using at most K colours.

By hypergraph Ramsey's theorem (Theorem 101), there is a constant N such that if the size $|\mathcal{C}|$ of the hypergraph is at least N, then there is a monochromatic (each hyperedge in the sub-hypergraph has the same colour) clique $\mathcal{C}' \subset \mathcal{C}$ of size at least M. Indeed, we can take N to be the hypergraph Ramsey number $N = R^{(L)}_K(M, \cdots, M)$. By Theorem 102, there is a constant $c' > 0$ such that $N < t_L(c' \cdot K \log K)$, where $t_L(\cdot)$ is the tower function of height L. Put in another way, there exists a subcode $\mathcal{C}' \subset \mathcal{C}$ of size at least M such that for some distribution $\widehat{P}_{\mathbf{x}_1, \cdots, \mathbf{x}_L} \in \mathcal{N}$, the joint type of every ordered tuple of L distinct codewords in $\mathcal{C}'$ is ζ-close to $\widehat{P}_{\mathbf{x}_1, \cdots, \mathbf{x}_L}$. I.e., for every $\mathcal{L} = (\underline{x}_1, \cdots, \underline{x}_L) \in \binom{\mathcal{C}'}{L}$,

$$\left\| \tau_{\underline{x}_1, \cdots, \underline{x}_L} - \widehat{P}_{\mathbf{x}_1, \cdots, \mathbf{x}_L} \right\|_{\mathrm{sav}} \leqslant \zeta.$$

This completes the proof of Lemma 57. ◀

Before proceeding with the proof of converse, we first list several corollaries that directly follow from the above lemma. They are concerned with basic properties of $(\zeta, P_{\mathbf{x}_1, \cdots, \mathbf{x}_L})$-equicoupled codes.

▶ **Corollary 60.** *Any two lists of L (ordered) codewords from $\mathcal{C}'$ have joint types 2ζ close to each other in sum-absolute-value distance.*

Proof. For any $\mathcal{L}_1 = \left(\underline{x}_{i_1}, \cdots, \underline{x}_{i_L}\right)$ and $\mathcal{L}_2 = \left(\underline{x}_{j_1}, \cdots, \underline{x}_{j_L}\right)$ in $\binom{\mathcal{C}'}{L}$,

$$\begin{aligned} \left\| \tau_{\underline{x}_{i_1}, \cdots, \underline{x}_{i_L}} - \tau_{\underline{x}_{j_1}, \cdots, \underline{x}_{j_L}} \right\|_{\mathrm{sav}} &\leqslant \left\| \tau_{\underline{x}_{i_1}, \cdots, \underline{x}_{i_L}} - \widehat{P}_{\mathbf{x}_1, \cdots, \mathbf{x}_L} \right\|_{\mathrm{sav}} + \left\| \widehat{P}_{\mathbf{x}_1, \cdots, \mathbf{x}_L} - \tau_{\underline{x}_{j_1}, \cdots, \underline{x}_{j_L}} \right\|_{\mathrm{sav}} \\ &\leqslant \zeta + \zeta \\ &= 2\zeta. \end{aligned} \tag{74}$$

◀

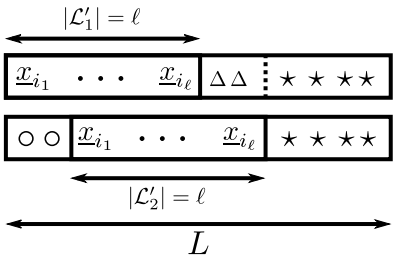


Figure 11 Two ways to complete the size-ℓ list $i_1, \cdots, i_\ell$ to size-L lists $\mathcal{L}_1, \mathcal{L}_2$, respectively. Triangles Δ, circles $\circ$ and stars $\star$ represent indices j's, k's and l's, respectively.

▶ **Corollary 61.** *Any two size-ℓ ($1 \leqslant \ell \leqslant L$) lists in $\mathcal{C}'$ have joint type 2ζ close to each other in sum-absolute-value distance, provided $|\mathcal{C}'| > 2L$.*

Proof. For any $\mathcal{L}'_1 = (\underline{x}_{i_1}, \cdots \underline{x}_{i_{L-1}})$ and $\mathcal{L}'_2 = (\underline{x}_{j_1}, \cdots, \underline{x}_{j_{L-1}})$ in $\binom{\mathcal{C}'}{L-1}$, take $\underline{x}_\iota \in \mathcal{C}' \backslash (\mathcal{L}'_1 \cup \mathcal{L}'_2)$. (This can be done as long as $|\mathcal{C}'| > 2L$.) Without loss of generality, assume $\iota > \max\{i_{L-1}, j_{L-1}\}$. Let $\mathcal{L}_1 := \mathcal{L}'_1 \cup \{\underline{x}_\iota\}, \mathcal{L}_2 := \mathcal{L}'_2 \cup \{\underline{x}_\iota\}$. We know that

$$
\begin{aligned}
2\zeta \geqslant & \left\| \tau_{\underline{x}_{i_1}, \cdots, \underline{x}_{i_{L-1}}, \underline{x}_\iota} - \tau_{\underline{x}_{j_1}, \cdots, \underline{x}_{j_{L-1}}, \underline{x}_\iota} \right\|_{\text{sav}} \\
= & \sum_{(x_1, \cdots, x_{L-1}, x) \in \mathcal{X}^L} \left| \tau_{\underline{x}_{i_1}, \cdots, \underline{x}_{i_{L-1}}, \underline{x}_\iota}(x_1, \cdots, x_{L-1}, x) - \tau_{\underline{x}_{j_1}, \cdots, \underline{x}_{j_{L-1}}, \underline{x}_\iota}(x_1, \cdots, x_{L-1}, x) \right| \\
\geqslant & \sum_{(x_1, \cdots, x_{L-1}) \in \mathcal{X}^{L-1}} \\
& \qquad \left| \sum_{x \in \mathcal{X}} \left(\tau_{\underline{x}_{i_1}, \cdots, \underline{x}_{i_{L-1}}, \underline{x}_\iota}(x_1, \cdots, x_{L-1}, x) - \tau_{\underline{x}_{j_1}, \cdots, \underline{x}_{j_{L-1}}, \underline{x}_\iota}(x_1, \cdots, x_{L-1}, x) \right) \right| \\
= & \sum_{(x_1, \cdots, x_{L-1}) \in \mathcal{X}^{L-1}} \left| \tau_{\underline{x}_{i_1}, \cdots, \underline{x}_{i_{L-1}}}(x_1, \cdots, x_{L-1}) - \tau_{\underline{x}_{j_1}, \cdots, \underline{x}_{j_{L-1}}}(x_1, \cdots, x_{L-1}) \right| \\
= & \left\| \tau_{\underline{x}_{i_1}, \cdots, \underline{x}_{i_{L-1}}} - \tau_{\underline{x}_{j_1}, \cdots, \underline{x}_{j_{L-1}}} \right\|_{\text{sav}}.
\end{aligned}
$$

Similarly we can see that Eqn. (74) holds also for size-ℓ ($\ell \leqslant L$) lists. ◀

For a subset $\mathcal{B} \subset [n]$, we let $P_{\mathbf{x}_\mathcal{B}}$ denote the marginalization of $P_{\mathbf{x}_1, \cdots, \mathbf{x}_L}$ onto the random variables indexed by elements in $\mathcal{B}$, $[P_{\mathbf{x}_1, \cdots, \mathbf{x}_L}]_{\{\mathbf{x}_i \,:\, i \in \mathcal{B}\}}$.

▶ **Corollary 62.** *For any $1 \leqslant \ell < L$ and any subsets $\mathcal{L}'_1, \mathcal{L}'_2 \in \binom{[n]}{\ell}$, $P_{\mathbf{x}_{\mathcal{L}'_1}}$ and $P_{\mathbf{x}_{\mathcal{L}'_2}}$ are 3ζ close to each other in sum-absolute-value distance, given $|\mathcal{C}'| > 2L$.*

Proof. Given two subsets $\mathcal{L}'_1, \mathcal{L}'_2 \subset [n]$ both of cardinality $\ell < L$, as long as the code size M is larger than $2L$, we can always find a tuple $1 \leqslant i_1 < \cdots < i_\ell \leqslant M$ such that it can be completed to L-tuples $\mathcal{L}_1, \mathcal{L}_2$ in two different ways

$$
\begin{aligned}
\mathcal{L}_1 = (&i_1, \cdots, i_{\ell-\ell'}, i_{\ell-\ell'+1}, \cdots, i_\ell, j_1, \cdots, j_{\ell-\ell'}, l_1, \cdots, l_{L-(2\ell-\ell')}), \\
\mathcal{L}_2 = (&k_1, \cdots, k_{\ell-\ell'}, i_1, \cdots, i'_\ell, i_{\ell'+1}, \cdots, i_\ell, l_1, \cdots, l_{L-(2\ell-\ell')}),
\end{aligned}
$$

for some $1 \leqslant k_1 < \cdots < k_{\ell-\ell'} < i_1 < \cdots < i_\ell < j_1 < \cdots < j_{\ell-\ell'} < l_1 < \cdots < l_{L-(2\ell-\ell')} \leqslant M$, where $\ell' = |\mathcal{L}'_1 \cap \mathcal{L}'_2|$. See Fig. 11. We know that

$$
\begin{aligned}
\|\tau_{\mathcal{L}_1} - P_{\mathbf{x}_1, \cdots, \mathbf{x}_L}\|_{\text{sav}} \leqslant & \zeta, \\
\|\tau_{\mathcal{L}_2} - P_{\mathbf{x}_1, \cdots, \mathbf{x}_L}\|_{\text{sav}} \leqslant & \zeta.
\end{aligned}
$$

Note that

$$\begin{aligned}
\zeta \geqslant & \left\| \tau_{\underline{x}_{\mathcal{L}_1}} - P_{\mathbf{x}_1,\cdots,\mathbf{x}_L} \right\|_{\text{sav}} \\
= & \sum_{\mathcal{L}_1 \in \{0,1\}^L} \left| \tau_{\underline{x}_{\mathcal{L}_1}}(\mathcal{L}_1) - P_{\mathbf{x}_1,\cdots,\mathbf{x}_L}(\mathcal{L}_1) \right| \\
\geqslant & \sum_{i_1,\cdots,i_\ell} \Bigg| \sum_{\mathcal{L}_1 \backslash \{i_1,\cdots,i_\ell\} \in \{0,1\}^{L-\ell}} \tau_{\underline{x}_{\mathcal{L}_1}}(i_1,\cdots,i_\ell,\mathcal{L}_1 \backslash \{i_1,\cdots,i_\ell\}) \\
& \qquad - P_{\mathbf{x}_1,\cdots,\mathbf{x}_L}(i_1,\cdots,i_\ell,\mathcal{L}_1 \backslash \{i_1,\cdots,i_\ell\}) \Bigg| \\
\leqslant & \sum_{i_1,\cdots,i_\ell} \left| \tau_{\underline{x}_{i_1},\cdots,\underline{x}_{i_\ell}}(i_1,\cdots,i_\ell) - P_{\mathbf{x}_{\mathcal{L}'_1}}(i_1,\cdots,i_\ell) \right| \\
= & \left\| \tau_{\underline{x}_{i_1},\cdots,\underline{x}_{i_\ell}} - P_{\mathbf{x}_{\mathcal{L}'_1}} \right\|_{\text{sav}}.
\end{aligned}$$

Similarly,

$$\left\| \tau_{\underline{x}_{i_1},\cdots,\underline{x}_{i_\ell}} - P_{\mathbf{x}_{\mathcal{L}'_2}} \right\|_{\text{sav}} \leqslant \zeta.$$

By triangle inequality,

$$\begin{aligned}
\left\| P_{\mathbf{x}_{\mathcal{L}'_1}} - P_{\mathbf{x}_{\mathcal{L}'_2}} \right\|_{\text{sav}} \leqslant & \left\| P_{\mathbf{x}_{\mathcal{L}'_1}} - \tau_{\underline{x}_{i_1},\cdots,\underline{x}_{i_\ell}} \right\|_{\text{sav}} + \left\| \tau_{\underline{x}_{i_1},\cdots,\underline{x}_{i_\ell}} - P_{\mathbf{x}_{\mathcal{L}'_2}} \right\|_{\text{sav}} \\
\leqslant & 2\zeta.
\end{aligned}$$

◀

▶ **Corollary 63.** *A $(\zeta, P_{\mathbf{x}_1,\cdots,\mathbf{x}_L})$-equicoupled code $\mathcal{C}'$ is $(3\zeta, P_{\mathbf{x}_1,\cdots,\mathbf{x}_\ell})$-equicoupled for any $1 \leqslant \ell \leqslant L$, as long as $|\mathcal{C}'| > 2L$.*

Proof. For any list of codewords $\underline{x}_{i_1}, \cdots, \underline{x}_{i_\ell}$, we can always find a completion of $(i_1, \cdots, i_\ell)$ to an L-tuple. Let $\mathcal{T}$ denote the set of locations of $i_1, \cdots, i_\ell$ in the completion. We know that

$$\left\| \tau_{\underline{x}_{i_1},\cdots,\underline{x}_{i_\ell}} - P_{\mathbf{x}_\mathcal{T}} \right\|_{\text{sav}} \leqslant \zeta.$$

By the previous corollary,

$$\begin{aligned}
\left\| \tau_{\underline{x}_{i_1},\cdots,\underline{x}_{i_\ell}} - P_{\mathbf{x}_1,\cdots,\mathbf{x}_\ell} \right\|_{\text{sav}} \leqslant & \left\| \tau_{\underline{x}_{i_1},\cdots,\underline{x}_{i_\ell}} - P_{\mathbf{x}_\mathcal{T}} \right\|_{\text{sav}} + \left\| P_{\mathbf{x}_\mathcal{T}} - P_{\mathbf{x}_1,\cdots,\mathbf{x}_\ell} \right\|_{\text{sav}} \\
\leqslant & \zeta + 2\zeta \\
= & 3\zeta.
\end{aligned}$$

◀

Now we apply the double counting trick used in the Plotkin-type bound for list decoding. We want to show that if $\widehat{P}_{\mathbf{x}_1,\cdots,\mathbf{x}_L}$ is not completely positive, then any $(L-1)$-list decodable code cannot be large.

▶ **Definition 64** (Symmetry of tensors). *A tensor $T \in \mathsf{Ten}_n^{\otimes m}$ is said to be* symmetric *if its components are invariant under permutation of indices, i.e., for any $\sigma \in S_m$ and any $(t_1, \cdots, t_m) \in [n]^m$,*

$$T(t_1, \cdots, t_m) = T\left(t_{\sigma(1)}, \cdots, t_{\sigma(m)}\right).$$

The set of dimension-n order-m symmetric tensors is denoted by $\mathsf{Sym}_n^{\otimes m}$.

13.2 Symmetric case

In this subsection, assume $\widehat{P}_{\mathbf{x}_1,\cdots,\mathbf{x}_L}$ is symmetric as a dimension-$|\mathcal{X}|$ order-L tensor. We are going to show that

▶ **Lemma 65** (Converse, symmetric case). *For a general adversarial channel given by $\mathcal{A} = \left(\mathcal{X}, \lambda_{\mathbf{x}}, \mathcal{S}, \lambda_{\mathbf{s}}, \mathcal{Y}, W_{\mathbf{y}|\mathbf{x},\mathbf{s}}\right)$ and an admissible input distribution $P_{\mathbf{x}} \in \lambda_{\mathbf{x}}$, if $\mathsf{CP}^{\otimes L}_{|\mathcal{X}|}(P_{\mathbf{x}}) \subseteq \mathcal{K}^{\otimes L}(P_{\mathbf{x}})$, the any $(\zeta, P_{\mathbf{x}_1,\cdots,\mathbf{x}_L})$-equicoupled $(L-1)$-list decodable code $\mathcal{C}'$ has size at most*

$$|\mathcal{C}'| \leqslant \max\left\{2(L-1), \frac{2^{L+1}L!}{\lambda}\right\},$$

where $\widehat{P}_{\mathbf{x}_1,\cdots,\mathbf{x}_L} \in \mathsf{Sym}^{\otimes L}_{|\mathcal{X}|}(P_{\mathbf{x}}) \backslash \mathcal{K}^{\otimes L}(P_{\mathbf{x}})$ is a symmetric, non-confusable joint distribution.

Proof. Since $\widehat{P}_{\mathbf{x}_1,\cdots,\mathbf{x}_L} \in \mathsf{Sym}^{\otimes L}_{|\mathcal{X}|}(P_{\mathbf{x}}) \backslash \mathsf{CP}^{\otimes L}_{|\mathcal{X}|}$, by duality (Theorem 96) between the CP tensor cone and coP tensor cone, there is a copositive tensor $Q \in \mathsf{coP}^{\otimes L}_{|\mathcal{X}|}$ such that $\|Q\|_{\mathrm{F}} = 1$ (by normalization) and

$$\left\langle P_{\mathbf{x}_1,\cdots,\mathbf{x}_L}, Q \right\rangle = -\eta \tag{75}$$

for some $\eta > 0$. Note that, by definition of λ, $\eta > \lambda$. We will bound

$$\sum_{(i_1,\cdots,i_L)\in[|\mathcal{C}'|]^L} \left\langle \tau_{\underline{x}_{i_1},\cdots,\underline{x}_{i_L}}, Q \right\rangle$$

from above and below and argue that if $|\mathcal{C}'|$ is larger than some constant[14], then we get a strictly negative upper bound and a non-negative lower bound. Such a contradiction implies that no positive rate is possible for $(L-1)$-list decoding if $\widehat{P}_{\mathbf{x}_1,\cdots,\mathbf{x}_L}$ is a non-CP symmetric distribution.

Upper bound.

Case when $i_1,\cdots,i_L \in [|\mathcal{C}'|]$ are not all distinct. For $i_1 \leqslant \cdots \leqslant i_L \in [|\mathcal{C}'|]$ not all distinct,

$$\left\langle \tau_{\underline{x}_{i_1},\cdots,\underline{x}_{i_L}}, Q \right\rangle \leqslant \left\| \tau_{\underline{x}_{i_1},\cdots,\underline{x}_{i_L}} \right\|_{\mathrm{F}} \|Q\|_{\mathrm{F}} \tag{76}$$

$$\leqslant \left\| \tau_{\underline{x}_{i_1},\cdots,\underline{x}_{i_L}} \right\|_{\mathrm{sav}} \|Q\|_{\mathrm{F}} \tag{77}$$

$$\leqslant 1. \tag{78}$$

Eqn. (76) is by Cauchy–Schwarz inequality. Eqn. (77) is because q-norm of a vector is non-increasing in q. Eqn. (78) is because a probability/type vector has one-norm 1 and Q is normalized to have F-norm 1.

Thus

$$\sum_{\substack{(i_1,\cdots,i_L)\in[|\mathcal{C}'|]^L \\ \text{not all distinct}}} \left\langle \tau_{\underline{x}_{i_1},\cdots,\underline{x}_{i_L}}, Q \right\rangle \leqslant |\mathcal{C}'|^L - \binom{|\mathcal{C}'|}{L} L!.$$

[14] Note that we will actually show that the size of the code is upper bounded by a *constant* (independent of blocklength n), not just that the rate of the code is vanishing.

Case when $i_1, \cdots, i_L \in [|\mathcal{C}'|]$ are all distinct. By Lemma 57, for any $\underline{x}_{i_1}, \cdots, \underline{x}_{i_L} \in \mathcal{C}'$ distinct,

$$\begin{aligned}\left\|\tau_{\underline{x}_{i_1},\cdots,\underline{x}_{i_L}} - \widehat{P}_{\mathbf{x}_1,\cdots,\mathbf{x}_L}\right\|_{\mathrm{mav}} \leqslant& \left\|\tau_{\underline{x}_{i_1},\cdots,\underline{x}_{i_L}} - \widehat{P}_{\mathbf{x}_1,\cdots,\mathbf{x}_L}\right\|_{\mathrm{sav}}\\ \leqslant& \zeta.\end{aligned}$$

For any $(i_1, \cdots, i_L) \in \binom{|\mathcal{C}'|}{L}$ distinct, let $\Delta_{i_1,\cdots,i_L} \coloneqq \tau_{\underline{x}_{i_1},\cdots,\underline{x}_{i_L}} - \widehat{P}_{\mathbf{x}_1,\cdots,\mathbf{x}_L}$. Immediately, $\|\Delta_{i_1,\cdots,i_L}\|_{\mathrm{mav}} \leqslant \zeta$.

Now,

$$\left\langle \tau_{\underline{x}_{i_1},\cdots,\underline{x}_{i_L}}, Q\right\rangle = \langle \Delta_{i_1,\cdots,i_L}, Q\rangle + \left\langle \widehat{P}_{\mathbf{x}_1,\cdots,\mathbf{x}_L}, Q\right\rangle.$$

Note that

$$\begin{aligned}|\langle \Delta_{i_1,\cdots,i_L}, Q\rangle| =& \left|\sum_{(x_1,\cdots,x_L)\in\mathcal{X}^L} \Delta_{i_1,\cdots,i_L}(x_1,\cdots,x_L)Q(x_1,\cdots,x_L)\right|\\ \leqslant& \sum_{(x_1,\cdots,x_L)\in\mathcal{X}^L} |\Delta_{i_1,\cdots,i_L}(x_1,\cdots,x_L)| \qquad (79)\\ \leqslant& |\mathcal{X}|^L \cdot \zeta, \qquad (80)\end{aligned}$$

where Eqn. (79) follows from triangle inequality and $\|Q\|_{\mathrm{mav}} \leqslant \|Q\|_{\mathrm{sav}} \leqslant \|Q\|_{\mathrm{F}} = 1$.

Hence

$$\begin{aligned}\left\langle \tau_{\underline{x}_{i_1},\cdots,\underline{x}_{i_L}}, Q\right\rangle \leqslant& -\eta + |\mathcal{X}|^L \zeta \qquad (81)\\ \leqslant& -\lambda + \frac{\lambda}{2} \qquad (82)\\ =& -\frac{\lambda}{2},\end{aligned}$$

where Eqn. (81) follows from Eqn. (75) and Eqn. (80), Eqn. (82) is by the definition of λ (Eqn. (72)) and the choice of ζ (Eqn. (73)).

Therefore,

$$\sum_{(i_1,\cdots,i_L)\in[|\mathcal{C}'|]^L \text{ distinct}} \left\langle \tau_{\underline{x}_{i_1},\cdots,\underline{x}_{i_L}}, Q\right\rangle \leqslant -\frac{\lambda}{2}\binom{|\mathcal{C}'|}{L} L!.$$

Overall,

$$\begin{aligned}\sum_{(i_1,\cdots,i_L)\in[|\mathcal{C}'|]^L} \left\langle \tau_{\underline{x}_{i_1},\cdots,\underline{x}_{i_L}}, Q\right\rangle \leqslant& |\mathcal{C}'|^L - \binom{|\mathcal{C}'|}{L} L! - \frac{\lambda}{2}\binom{|\mathcal{C}'|}{L} L!\\ <& 0 \qquad (83)\end{aligned}$$

if $|\mathcal{C}'|$ is sufficiently large. To see this, note that $p(|\mathcal{C}'|) \coloneqq |\mathcal{C}'|^L - \binom{|\mathcal{C}'|}{L} L!$ is a polynomial in $|\mathcal{C}'|$ of degree $L-1$, while $-\frac{\lambda}{2}\binom{|\mathcal{C}'|}{L} L!$ is a polynomial in $|\mathcal{C}'|$ of degree L. To give an explicit bound on $|\mathcal{C}'|$, note that the RHS of (83) equals

$$\begin{aligned}p\left(|\mathcal{C}'|\right) - \frac{\lambda}{2}|\mathcal{C}'|\left(|\mathcal{C}'|-1\right)\cdots\left(|\mathcal{C}'|-(L-1)\right) \leqslant& L\cdot(L-1)!\cdot|\mathcal{C}'|^{L-1} - \frac{\lambda}{2}\left(|\mathcal{C}'|-(L-1)\right)^L\\ =& L!\cdot|\mathcal{C}'|^{L-1} - \frac{\lambda}{2}\left(|\mathcal{C}'|-(L-1)\right)^L.\end{aligned}$$

In the above inequality, to upper bound $p\left(|\mathcal{C}'|\right)$, we replace each term of p with a monomial with the largest possible coefficient in absolute value and the largest possible degree. To make the RHS negative, we want

$$(L!)^{\frac{1}{L}}\left|\mathcal{C}'\right|^{1-\frac{1}{L}} < \left(\frac{\lambda}{2}\right)^{\frac{1}{L}}\left|\mathcal{C}'\right| - \left(\frac{\lambda}{2}\right)^{\frac{1}{L}}(L-1).$$

One can easily check that when $|\mathcal{C}'| > 2(L-1)$,

$$\frac{1}{2}\left(\frac{\lambda}{2}\right)^{\frac{1}{L}}\left|\mathcal{C}'\right| < \left(\frac{\lambda}{2}\right)^{\frac{1}{L}}\left|\mathcal{C}'\right| - \left(\frac{\lambda}{2}\right)^{\frac{1}{L}}(L-1).$$

Moreover, when $|\mathcal{C}'| > \frac{2^{L+1}L!}{\lambda}$,

$$(L!)\left|\mathcal{C}'\right|^{1-\frac{1}{L}} < \frac{1}{2}\left(\frac{\lambda}{2}\right)^{\frac{1}{L}}\left|\mathcal{C}'\right|$$

is satisfied, so is the original inequality (83).

Overall, we have that

$$\sum_{(i_1,\cdots,i_L)\in[|\mathcal{C}'|]^L}\left\langle \tau_{\underline{x}_{i_1},\cdots,\underline{x}_{i_L}}, Q\right\rangle < 0$$

as long as

$$\left|\mathcal{C}'\right| > \max\left\{2(L-1), \frac{2^{L+1}L!}{\lambda}\right\}. \tag{84}$$

Though the bound (84) is crude, it is a *constant* not depending on the blocklength n.

Lower bound.

$$\begin{aligned}
&\sum_{(i_1,\cdots,i_L)\in[|\mathcal{C}'|]^L}\left\langle \tau_{\underline{x}_{i_1},\cdots,\underline{x}_{i_L}}, Q\right\rangle\\
=&\sum_{(i_1,\cdots,i_L)\in[|\mathcal{C}'|]^L}\sum_{(x_1,\cdots,x_L)\in\mathcal{X}^L}\tau_{\underline{x}_{i_1},\cdots,\underline{x}_{i_L}}(x_1,\cdots,x_L)Q(x_1,\cdots,x_L)\\
=&\sum_{(x_1,\cdots,x_L)\in\mathcal{X}^L}\sum_{(i_1,\cdots,i_L)\in[|\mathcal{C}'|]^L}\frac{1}{n}\sum_{j=1}^{n}\mathbb{1}_{\left\{\underline{x}_{i_1}(j)=x_1,\cdots,\underline{x}_{i_L}(j)=x_L\right\}}Q(x_1,\cdots,x_L)\\
=&\frac{1}{n}\sum_{(x_1,\cdots,x_L)\in\mathcal{X}^L}\sum_{j=1}^{n}\sum_{(i_1,\cdots,i_L)\in[|\mathcal{C}'|]^L}\mathbb{1}_{\left\{\underline{x}_{i_1}(j)=x_1\right\}}\cdots\mathbb{1}_{\left\{\underline{x}_{i_L}(j)=x_L\right\}}Q(x_1,\cdots,x_L)\\
=&\frac{1}{n}\sum_{(x_1,\cdots,x_L)\in\mathcal{X}^L}\sum_{j=1}^{n}\left(\sum_{i\in[|\mathcal{C}'|]}\mathbb{1}_{\left\{\underline{x}_i(j)=x_1\right\}}\right)\cdots\left(\sum_{i\in[|\mathcal{C}'|]}\mathbb{1}_{\left\{\underline{x}_i(j)=x_L\right\}}\right)Q(x_1,\cdots,x_L)\\
=&\frac{|\mathcal{C}'|^L}{n}\sum_{(x_1,\cdots,x_L)\in\mathcal{X}^L}\sum_{j=1}^{n}P_{\mathbf{x}}^{(j)}(x_1)\cdots P_{\mathbf{x}}^{(j)}(x_L)Q(x_1,\cdots,x_L) \qquad (85)\\
=&\frac{|\mathcal{C}'|^L}{n}\sum_{j=1}^{n}\left\langle\left(P_{\mathbf{x}}^{(j)}\right)^{\otimes L}, Q\right\rangle\\
\geqslant&0. \qquad (86)
\end{aligned}$$

To see equality (85), let $P_{\mathbf{x}}^{(j)}$ be the empirical distribution of the j-th *column* of $\mathcal{C}'$ as a $|\mathcal{C}'| \times n$ matrix, i.e., for $x \in \mathcal{X}$,

$$P_{\mathbf{x}}^{(j)}(x) \coloneqq \frac{1}{|\mathcal{C}'|} \sum_{i=1}^{|\mathcal{C}'|} \mathbb{1}_{\{\underline{x}_i(j)=x\}}.$$

The last inequality (86) follows since $\left(P_{\mathbf{x}}^{(j)}\right)^{\otimes L}$ is a completely positive tensor.

The lower bound and the upper bound are contradicting each other, which completes the proof. ◀

13.3 Asymmetric case

In this section, we handle the asymmetric case of the converse.

▶ **Definition 66** (Asymmetry of tensors). *For a joint distribution $P_{\mathbf{x}_1,\cdots,\mathbf{x}_L} \in \Delta\left(\mathcal{X}^L\right)$, alternatively a tensor $P_{\mathbf{x}_1,\cdots,\mathbf{x}_L} \in \mathsf{Ten}_{|\mathcal{X}|}^{\otimes L}$, define its* asymmetry *as*

$$\operatorname{asymm}(P_{\mathbf{x}_1,\cdots,\mathbf{x}_L}) \coloneqq \max_{(x_1,\cdots,x_L)\in\mathcal{X}^L} \max_{\sigma\in S_L\setminus\{\mathrm{id}\}} \left|P_{\mathbf{x}_1,\cdots,\mathbf{x}_L}(x_1,\cdots,x_L) - P_{\mathbf{x}_1,\cdots,\mathbf{x}_L}(x_{\sigma(1)},\cdots,x_{\sigma(L)})\right|.$$

▶ Remark 67. If $\operatorname{asymm}(P_{\mathbf{x}_1,\cdots,\mathbf{x}_L}) = 0$, then $P_{\mathbf{x}_1,\cdots,\mathbf{x}_L}$ is symmetric in the sense of Definition 64.

We will show that

▶ **Lemma 68** (Converse, asymmetric case). *If $P_{\mathbf{x}_1,\cdots,\mathbf{x}_L} \in \mathsf{Ten}_{|\mathcal{X}|}^{\otimes L}(P_{\mathbf{x}})$ is asymmetric as a tensor in $\mathsf{Ten}_{|\mathcal{X}|}^{\otimes L}(P_{\mathbf{x}})$ and has asymmetry α, then for any $0 < \zeta < \alpha$, any $(\zeta, P_{\mathbf{x}_1,\cdots,\mathbf{x}_L})$-equicoupled (w.r.t. max-absolute-value distance)*[15] *code $\mathcal{C}'$ has size at most*

$$|\mathcal{C}'| \leqslant \exp\left(\frac{c}{\alpha/\binom{L}{2} - \zeta}\right) + L - 2$$

for some absolute constant $c > 0$.

Lemma 68 is shown by reducing the problem, in a nontrivial way, from general values of L to $L = 2$ in which case it is known [43] that such codes cannot be large.

▶ **Lemma 69** (Reduction from general L to $L = 2$). *If $P_{\mathbf{x}_1,\cdots,\mathbf{x}_L} \in \mathsf{Ten}_{|\mathcal{X}|}^{\otimes L}$ has asymmetry $\operatorname{asymm}(P_{\mathbf{x}_1,\cdots,\mathbf{x}_L}) = \alpha$, then among the following distributions*

$$P_{\mathbf{y}_1,\mathbf{z}_1},\ P_{\mathbf{y}_2,\mathbf{z}_2},\cdots,P_{\mathbf{y}_{L-1},\mathbf{z}_{L-1}},$$

there is at least one distribution $P_{\mathbf{y}_{i^*},\mathbf{z}_{i^*}}$ ($i^* \in [L-1]$) with asymmetry at least

$$\operatorname{asymm}\left(P_{\mathbf{y}_{i^*},\mathbf{z}_{i^*}}\right) = \frac{\alpha}{\binom{L}{2}}.$$

[15] Note that ζ-equicoupledness w.r.t. sum-absolute-value distance implies ζ-equicoupledness w.r.t. max-absolute-value distance. Hence this lemma directly applies to the subcode we obtained in the previous section.

Here, for $i \in [L-1]$, $\mathbf{y}_i$ and $\mathbf{z}_i$ $(1 \leqslant i \leqslant L-1)$ are tuples of random variables defined as

$$\begin{aligned} \mathbf{y}_i &:= (\mathbf{x}_1, \cdots, \mathbf{x}_{i-1}, \mathbf{x}_i, \quad\quad \mathbf{x}_{i+2}, \cdots, \mathbf{x}_L), \\ \mathbf{z}_i &:= (\mathbf{x}_1, \cdots, \mathbf{x}_{i-1}, \quad\quad \mathbf{x}_{i+1}, \mathbf{x}_{i+2}, \cdots, \mathbf{x}_L), \end{aligned}$$

respectively.

Proof. The proof is by contradiction. We will show that if all of $\{P_{\mathbf{y}_i,\mathbf{z}_i}\}_{1\leqslant i\leqslant L-1}$ have small asymmetry, then they do not not suffice to back propagate their asymmetry using transpositions to result in the asymmetry α of $P_{\mathbf{x}_1,\cdots,\mathbf{x}_L}$. To make this intuition clear, assume, towards a contradiction, that all of the distributions $\{P_{\mathbf{y}_i,\mathbf{z}_i}\}_{1\leqslant i\leqslant L-1}$ have asymmetry strictly less than $\alpha' = \frac{\alpha}{\binom{L}{2}}$,

$$\operatorname{asymm}(P_{\mathbf{y}_i,\mathbf{z}_i}) < \frac{\alpha}{\binom{L}{2}}, \ \forall i \in [L-1]. \tag{87}$$

Assume the asymmetry of $P_{\mathbf{x}_1,\cdots,\mathbf{x}_L}$ is witnessed by coordinates $(x_1,\cdots,x_L) \in \mathcal{X}^L$ and permutation $\pi \in S_L$, i.e.,

$$\begin{aligned} \alpha &= \left|P_{\mathbf{x}_1,\cdots,\mathbf{x}_L}(x_1,\cdots,x_L) - P_{\mathbf{x}_1,\cdots,\mathbf{x}_L}(x_{\pi(1)},\cdots,x_{\pi(L)})\right| \\ &= \left|P_{\mathbf{x}_1,\cdots,\mathbf{x}_L}(x_1,\cdots,x_L) - P_{\mathbf{x}_{\pi(1)},\cdots,\mathbf{x}_{\pi(L)}}(x_1,\cdots,x_L)\right|. \end{aligned} \tag{88}$$

Note that the set of transpositions $\{\sigma_1,\cdots,\sigma_{L-1}\}$ forms a generator set of S_L, where

$$\sigma_i := \begin{pmatrix} 1 & \cdots & i-1 & i & i+1 & i+2 & \cdots & L \\ 1 & \cdots & i-1 & i+1 & i & i+2 & \cdots & L \end{pmatrix}.$$

Any permutation $\sigma \in S_L$ can be written as a product of σ_i's, $\sigma = \sigma_{i_\ell}\cdots\sigma_{i_1}$ for some positive integer ℓ and a subset of transpositions, $i_j \in [L-1]$ for each $j \in [\ell]$. Such a representation, in particular the value of ℓ, is not necessarily unique. Let

$$\ell(\sigma) := \min\{\ell \in \mathbb{Z}_{\geqslant 0} \colon \sigma = \sigma_{i_\ell}\cdots\sigma_{i_1} \text{ transposition representation}\}$$

be the *transposition length* of σ, i.e., the length of the shortest representation using product of transpositions. Let

$$\ell^* := \max_{\sigma\in S_L} \ell(\sigma).$$

We claim that $\ell^* \leqslant \binom{L}{2}$. To see this, it suffices to bound $\ell(\sigma)$ for the worst case permutation

$$\sigma = \begin{pmatrix} 1 & 2 & \cdots & L \\ L & L-1 & \cdots & 1 \end{pmatrix}.$$

The claim follows by noting that σ can be written as

$$\sigma = \prod_{j=1}^{L-1} \prod_{i=j,j-1,\cdots,1} \sigma_i, \tag{89}$$

which contains $\binom{L}{2}$ transpositions.

▶ Remark 70. A potential confusion may arise from two conflicting conventions that

1. a product is usually written from left to right, i.e.,

$$\prod_{i=1}^{\ell} \sigma_i = \sigma_1\cdots\sigma_\ell;$$

2. a composition of permutations acts like functions on an element from right to left, i.e., for $\sigma, \pi \in S_L$ and $i \in [L]$,

$$(\sigma\pi)(i) = \sigma(\pi(i)).$$

With this kept in mind, the representation in Eqn. (89) should be understood as

$$\sigma = (\sigma_1)(\sigma_2\sigma_1)\cdots(\sigma_{L-2}\cdots\sigma_2\sigma_1)(\sigma_{L-1}\cdots\sigma_2\sigma_1).$$

The product in the $(L-1)$-st parenthesis (from left to right) moves L in the initial sequence $(L, L-1, \cdots, 1)$ to the L-th position; the product in the $(L-2)$-nd parenthesis moves $L-1$ to the $(L-1)$-st position; ...; the permutation σ_1 in the 1-st parenthesis moves 2 to the 2-st position, and automatically 1 is in the 1-st position. We get the target sequence $(1, 2, \cdots, L)$. We can write

$$\pi = \prod_{j=\ell,\ell-1,\cdots,1} \sigma_{i_j}, \tag{90}$$

for some $\ell \leqslant \ell^* \leqslant \binom{L}{2}$.

Our assumption Eqn. (87) implies that, for any $(x_1, \cdots, x_L) \in \mathcal{X}^L$ and any transposition σ_i,

$$\begin{aligned}
&\left|P_{\mathbf{x}_1,\cdots,\mathbf{x}_L}(x_1,\cdots,x_L) - P_{\mathbf{x}_{\sigma_i(1)},\cdots,\mathbf{x}_{\sigma_i(L)}}(x_1,\cdots,x_L)\right| \\
=&\left|P_{\mathbf{x}_1,\cdots,\mathbf{x}_{i-1},\mathbf{x}_i,\mathbf{x}_{i+1},\mathbf{x}_{i+2},\cdots,\mathbf{x}_L}(x_1,\cdots,x_L) - P_{\mathbf{x}_1,\cdots,\mathbf{x}_{i-1},\mathbf{x}_{i+1},\mathbf{x}_i,\mathbf{x}_{i+2},\cdots,\mathbf{x}_L}(x_1,\cdots,x_L)\right| \\
=&\left|P_{(\mathbf{x}_1,\cdots,\mathbf{x}_{i-1},\mathbf{x}_i,\mathbf{x}_{i+2},\cdots,\mathbf{x}_L),(\mathbf{x}_1,\cdots,\mathbf{x}_{i-1},\mathbf{x}_{i+1},\mathbf{x}_{i+2},\cdots,\mathbf{x}_L)}\begin{pmatrix}(x_1,\cdots,x_{i-1},x_i,x_{i+2},\cdots,x_L),\\(x_1,\cdots,x_{i-1},x_{i+1},x_{i+2},\cdots,x_L)\end{pmatrix}\right. \\
&\quad\left. - P_{(\mathbf{x}_1,\cdots,\mathbf{x}_{i-1},\mathbf{x}_{i+1},\mathbf{x}_{i+2},\cdots,\mathbf{x}_L),(\mathbf{x}_1,\cdots,\mathbf{x}_{i-1},\mathbf{x}_i,\mathbf{x}_{i+2},\cdots,\mathbf{x}_L)}\begin{pmatrix}(x_1,\cdots,x_{i-1},x_i,x_{i+2},\cdots,x_L),\\(x_1,\cdots,x_{i-1},x_{i+1},x_{i+2},\cdots,x_L)\end{pmatrix}\right| \\
=&\left|P_{\mathbf{y}_i,\mathbf{z}_i}(y,z) - P_{\mathbf{z}_i,\mathbf{y}_i}(y,z)\right| \\
=&\left|P_{\mathbf{y}_i,\mathbf{z}_i}(y,z) - P_{\mathbf{y}_i,\mathbf{z}_i}(z,y)\right| \\
<&\alpha',
\end{aligned} \tag{91}$$

where

$$\begin{aligned}
y &:= (x_1, \cdots, x_{i-1}, x_i, x_{i+2}, \cdots, x_L), \\
z &:= (x_1, \cdots, x_{i-1}, x_{i+1}, x_{i+2}, \cdots, x_L).
\end{aligned}$$

Now

$$\begin{aligned}
\alpha =& \left|P_{\mathbf{x}_1,\cdots,\mathbf{x}_L}(x_1,\cdots,x_L) - P_{\mathbf{x}_{\pi(1)},\cdots,\mathbf{x}_{\pi(L)}}(x_1,\cdots,x_L)\right| && (92)\\
\leqslant& \left|P_{\mathbf{x}_1,\cdots,\mathbf{x}_L}(x_1,\cdots,x_L) - P_{\mathbf{x}_{\sigma_{i_1}(1)},\cdots,\mathbf{x}_{\sigma_{i_1}(L)}}(x_1,\cdots,x_L)\right| \\
&+ \left|P_{\mathbf{x}_{\sigma_{i_1}(1)},\cdots,\mathbf{x}_{\sigma_{i_1}(L)}}(x_1,\cdots,x_L) - P_{\mathbf{x}_{\pi(1)},\cdots,\mathbf{x}_{\pi(L)}}(x_1,\cdots,x_L)\right| && (93)\\
<&\alpha' + \left|P_{\mathbf{x}_{\sigma_{i_1}(1)},\cdots,\mathbf{x}_{\sigma_{i_1}(L)}}(x_1,\cdots,x_L) - P_{\mathbf{x}_{\pi(1)},\cdots,\mathbf{x}_{\pi(L)}}(x_1,\cdots,x_L)\right| && (94)\\
\leqslant&\alpha' + \left|P_{\mathbf{x}_{\sigma_{i_1}(1)},\cdots,\mathbf{x}_{\sigma_{i_1}(L)}}(x_1,\cdots,x_L) - P_{\mathbf{x}_{\sigma_{i_2}\sigma_{i_1}(1)},\cdots,\mathbf{x}_{\sigma_{i_2}\sigma_{i_1}(L)}}(x_1,\cdots,x_L)\right| \\
&+ \left|P_{\mathbf{x}_{\sigma_{i_2}\sigma_{i_1}(1)},\cdots,\mathbf{x}_{\sigma_{i_2}\sigma_{i_1}(L)}}(x_1,\cdots,x_L) - P_{\mathbf{x}_{\pi(1)},\cdots,\mathbf{x}_{\pi(L)}}(x_1,\cdots,x_L)\right| && (95)\\
<&2\alpha' + \left|P_{\mathbf{x}_{\sigma_{i_2}\sigma_{i_1}(1)},\cdots,\mathbf{x}_{\sigma_{i_2}\sigma_{i_1}(L)}}(x_1,\cdots,x_L) - P_{\mathbf{x}_{\pi(1)},\cdots,\mathbf{x}_{\pi(L)}}(x_1,\cdots,x_L)\right| && (96)
\end{aligned}$$

$$\cdots$$

$$\leqslant(\ell-1)\alpha' + \left|P_{\mathbf{x}_{\sigma_{i_{\ell-1}}\cdots\sigma_{i_1}(1)},\cdots,\mathbf{x}_{\sigma_{i_{\ell-1}}\cdots\sigma_{i_1}(L)}}(x_1,\cdots,x_L) - P_{\mathbf{x}_{\pi(1)},\cdots,\mathbf{x}_{\pi(L)}}(x_1,\cdots,x_L)\right| \tag{97}$$

$$=(\ell-1)\alpha' + \Big|P_{\mathbf{x}_{\sigma_{i_{\ell-1}}\cdots\sigma_{i_1}(1)},\cdots,\mathbf{x}_{\sigma_{i_{\ell-1}}\cdots\sigma_{i_1}(L)}}(x_1,\cdots,x_L) - P_{\mathbf{x}_{\sigma_{i_\ell}\sigma_{i_{\ell-1}}\cdots\sigma_{i_1}(1)},\cdots,\mathbf{x}_{\sigma_{i_\ell}\sigma_{i_{\ell-1}}\cdots\sigma_{i_1}(L)}}(x_1,\cdots,x_L)\Big| \tag{98}$$

$$<\ell\alpha' \tag{99}$$

$$\leqslant\binom{L}{2}\alpha'$$

$$=\alpha. \tag{100}$$

1. Eqn. (92) follows from Eqn. (88).
2. Eqn. (93), (95), etc., are by triangle inequality.
3. Eqn. (94), (96), (99), etc. , are by Eqn. (91).
4. Eqn. (97) is by recursively applying the previous calculations.
5. Eqn. (98) is by the transposition representation of π (Eqn. (90)).
6. Eqn. (100) is by the choice of α'.

We reach a contradiction that α is strictly less than itself. This finishes the proof. ◀

Next, we show the key lemma 68 in this section. Note that, according to the statement, Lemma 68 is independent of the channel that the code $\mathcal{C}'$ is used for. Hence we will directly prove the random variable version of this lemma which is concerned with fundamental properties of joint distributions. If the joint distribution of a sequence of random variables has all of its size-L marginals being ζ-close to some *asymmetric* distribution, then such a sequence cannot be infinitely long. We will prove a finite upper bound on the length of the sequence by reducing this problem from the general $L > 2$ case to the $L = 2$ case. In the $L = 2$ case, prior work [43] shows that this is indeed the case.

▶ **Lemma 71** (Converse, asymmetric case, $L = 2$ [43]). *Assume $P_{\mathbf{x}_1,\mathbf{x}_2} \in \Delta(\mathcal{X}^2)$ has asymmetry* $\operatorname{asymm}(P_{\mathbf{x}_1,\mathbf{x}_2}) = \alpha$. *Let $\mathbf{w}_1,\cdots,\mathbf{w}_M$ be a sequence of M random variables supported on $\mathcal{X}$ such that for every $1 \leqslant j_1 < j_2 \leqslant M$,*

$$\left\|P_{\mathbf{w}_{j_1},\mathbf{w}_{j_2}} - P_{\mathbf{x}_1,\mathbf{x}_2}\right\|_{\text{mav}} \leqslant \zeta.$$

for some $0 < \zeta < \alpha$. Then

$$M \leqslant \exp\left(\frac{c}{\alpha-\zeta}\right)$$

for some universal constant $c > 0$.

We are now ready to prove the restated version of Lemma 68.

▶ **Lemma 72** (Converse, asymmetric case, general L). *If a joint distribution $P_{\mathbf{x}_1,\cdots,\mathbf{x}_L} \in \Delta\left(\mathcal{X}^L\right)$ has asymmetry* $\operatorname{asymm}(P_{\mathbf{x}_1,\cdots,\mathbf{x}_L}) = \alpha$, *and a sequence of M random variables $\mathbf{w}_1,\cdots,\mathbf{w}_M$ supported on $\mathcal{X}$ satisfies that for any $1 \leqslant j_1 < \cdots < j_L \leqslant M$,*

$$\left\|P_{\mathbf{w}_{j_1},\cdots,\mathbf{w}_{j_L}} - P_{\mathbf{x}_1,\cdots,\mathbf{x}_L}\right\|_{\text{mav}} \leqslant \zeta. \tag{101}$$

Then

$$M \leqslant \exp\left(\frac{c}{\alpha/\binom{L}{2} - \zeta}\right) + L - 2$$

for some universal constant $c > 0$*.*

Proof. Construct the following $L-1$ sequences $\left\{\mathbf{v}^{(i)}\right\}_{1\leqslant i\leqslant L-1}$ of random variables, each of which has length $M-L+2$,

$$\begin{array}{lclllll}
\mathbf{v}^{(1)} & = & (\mathbf{v}_1^{(1)}, & \mathbf{v}_2^{(1)}, & \cdots, & \mathbf{v}_{M-L+2}^{(1)}), \\
\mathbf{v}^{(2)} & = & (\mathbf{v}_2^{(2)}, & \mathbf{v}_2^{(2)}, & \cdots, & \mathbf{v}_{M-L+3}^{(2)}), \\
\cdots & & & & & \\
\mathbf{v}^{(L-1)} & = & (\mathbf{v}_{L-1}^{(L-1)}, & \mathbf{v}_2^{(1)}, & \cdots, & \mathbf{v}_M^{(L-1)}).
\end{array}$$

For $1 \leqslant i \leqslant L-1$ and $i \leqslant j \leqslant M-L+i+1$, $\mathbf{v}_j^{(i)}$ is defined as atuple

$$\mathbf{v}_j^{(i)} := \left(\mathbf{w}_1, \cdots, \mathbf{w}_{i-1}, \mathbf{w}_j, \mathbf{w}_{M-L+i+2}, \cdots, \mathbf{w}_M\right).$$

Then, for any

$$\begin{array}{lclllllll}
v_1 & := & (x_1, & \cdots, & x_{i-1}, & x_i, & & x_{i+2}, & \cdots, & x_L) \in \mathcal{X}^{L-1}, \\
v_2 & := & (x_1, & \cdots, & x_{i-1}, & & x_{i+1}, & x_{i+2}, & \cdots, & x_L) \in \mathcal{X}^{L-1},
\end{array}$$

and $i \leqslant j_1 < j_2 \leqslant M-L+i+1$, we have

$$\begin{aligned}
&\left|P_{\mathbf{v}_{j_1}^{(i)}, \mathbf{v}_{j_2}^{(i)}}(v_1, v_2) - P_{\mathbf{y}_i, \mathbf{z}_i}(v_1, v_2)\right| \\
=&\Bigg|P_{(\mathbf{w}_1, \cdots, \mathbf{w}_{i-1}, \mathbf{w}_{j_1}, \mathbf{w}_{M-L+i+2}, \cdots, \mathbf{w}_M), (\mathbf{w}_1, \cdots, \mathbf{w}_{i-1}, \mathbf{w}_{j_2}, \mathbf{w}_{M-L+i+2}, \cdots, \mathbf{w}_M)} \\
&\qquad\qquad\begin{pmatrix} (x_1, \cdots, x_{i-1}, x_i, x_{i+2}, \cdots, x_L), \\ (x_1, \cdots, x_{i-1}, x_{i+1}, x_{i+2}, \cdots, x_L) \end{pmatrix} \\
&\quad - P_{(\mathbf{x}_1, \cdots, \mathbf{x}_{i-1}, \mathbf{x}_i, \mathbf{x}_{i+2}, \cdots, \mathbf{x}_L), (\mathbf{x}_1, \cdots, \mathbf{x}_{i-1}, \mathbf{x}_{i+1}, \mathbf{x}_{i+2}, \cdots, \mathbf{x}_L)}\begin{pmatrix} (x_1, \cdots, x_{i-1}, x_i, x_{i+2}, \cdots, x_L), \\ (x_1, \cdots, x_{i-1}, x_{i+1}, x_{i+2}, \cdots, x_L) \end{pmatrix}\Bigg| \\
=&\Big|P_{\mathbf{w}_1, \cdots, \mathbf{w}_{i-1}, \mathbf{w}_{j_1}, \mathbf{w}_{j_2}, \mathbf{w}_{M-L+i+2}, \cdots, \mathbf{w}_M}(x_1, \cdots, x_{i-1}, x_i, x_{i+1}, x_{i+2}, \cdots, x_L) \\
&\qquad\qquad - P_{\mathbf{x}_1, \cdots, \mathbf{x}_L}(x_1, \cdots, x_L)\Big| \\
\leqslant&\zeta,
\end{aligned}$$

by the assumption Eqn. (101). Therefore, all sequences $\mathbf{v}^{(i)}$'s are $(\zeta, P_{\mathbf{y}_i, \mathbf{z}_i})$-equicoupled, $1 \leqslant i \leqslant L-1$.

Since $P_{\mathbf{x}_1, \cdots, \mathbf{x}_L}$ is α-asymmetric, by Lemma 69, at least one of the distributions $P_{\mathbf{y}_i, \mathbf{z}_i}$'s ($1 \leqslant i \leqslant L-1$) is at least α'-asymmetric ($\alpha' = \alpha/\binom{L}{2}$). Without loss of generality, assume $P_{\mathbf{y}_{i_0}, \mathbf{z}_{i_0}}$ is $\geqslant \alpha'$-asymmetric. Then the i_0-th sequence $\mathbf{v}^{(i_0)}$ is short by Lemma 71,

$$M - L + 2 \leqslant \exp\left(\frac{c}{\alpha' - \zeta}\right),$$

for some universal constant $c > 0$. Hence

$$M \leqslant \exp\left(\frac{c}{\alpha/\binom{L}{2} - \zeta}\right) + L - 2,$$

which finishes the proof. ◀

▶ Remark 73 (Asymmetric but projectively symmetric tensors). Lemma 69 does not follow from naïvely marginalizing an asymmetric distribution $P_{\mathbf{x}_1,\cdots,\mathbf{x}_L}$ and hoping that $P_{\mathbf{x}_i,\mathbf{x}_j}$ is asymmetric for some $1 \leqslant i < j \leqslant L$. Just like there exist asymmetric matrices (self-couplings) with the same column sum and row sum, we should not expect that the asymmetry of a tensor is preserved under projections.

We say that a tensor $P_{\mathbf{x}_1,\cdots,\mathbf{x}_L} \in \mathsf{Ten}_{|\mathcal{X}|}^{\otimes L}$ is *ℓ-projectively symmetric* ($1 \leqslant \ell < L$) if all of its order-ℓ projections are symmetric, i.e., for any $1 \leqslant i_1 < \cdots < i_\ell \leqslant L$,

$$P_{\mathbf{x}_{i_1},\cdots,\mathbf{x}_{i_\ell}} \coloneqq [P_{\mathbf{x}_1,\cdots,\mathbf{x}_L}]_{\mathbf{x}_{i_1},\cdots,\mathbf{x}_{i_\ell}} \in \mathsf{Ten}_{|\mathcal{X}|}^{\otimes \ell}$$

is symmetric.

One can easily verify the following facts.

▶ **Lemma 74.** *Let $P_{\mathbf{x}_1,\cdots,\mathbf{x}_L}$ be a tensor of dimension $|\mathcal{X}|$ and order L.*

1. *If $P_{\mathbf{x}_1,\cdots,\mathbf{x}_L}$ is ℓ-projectively symmetric ($1 \leqslant \ell < L$), then all of its order-ℓ' ($1 \leqslant \ell' < \ell$) marginals are the same.*
2. *If $P_{\mathbf{x}_1,\cdots,\mathbf{x}_L}$ is ℓ-projectively symmetric ($1 \leqslant \ell < L$), then it is also ℓ'-projectively symmetric for any $1 \leqslant \ell' < \ell$.*
3. *A symmetric tensor $P_{\mathbf{x}_1,\cdots,\mathbf{x}_L}$ is also ℓ-projectively symmetric for all $1 \leqslant \ell < L$. In particular, it is a self-coupling, i.e., $P_{\mathbf{x}_i}$ is the same for all $i \in [L]$.*

We provide an example showing that the asymmetry of a tensor cannot be recovered from all of its lower order projections. That is, there is an asymmetric tensor with every projection of one less order being symmetric.

We now construct a concrete example. In order for a dimension-2 order-3 tensor $T\colon [2]^3 \to \mathbb{R}$ to be symmetric, it has to satisfy the following system $\mathcal{E}_1$ of linear equations,

$$t_{112} = t_{121}, \quad t_{121} = t_{211}, \quad t_{212} = t_{122}, \quad t_{122} = t_{221}.$$

where $t_{ijk} \coloneqq T(i,j,k)$ for $i,j,k \in [2]$. On the other hand, for it to be projectively symmetric, it has to satisfy the following system $\mathcal{E}_2$ of linear equations,

$$\begin{aligned} t_{122} + t_{121} &= t_{212} + t_{211}, \\ t_{112} + t_{122} &= t_{211} + t_{221}, \\ t_{121} + t_{221} &= t_{112} + t_{212}. \end{aligned}$$

Additionally, for T to represent a joint distribution, all entries should be non-negative and sum up to one. Note that $\mathcal{E}_2$ is a *less determined* system than $\mathcal{E}_1$, which means that we should be able to find a solution to $\mathcal{E}_2$ which does not satisfy $\mathcal{E}_1$.

Indeed, consider the following explicit example of $T \in \mathsf{Ten}_2^{\otimes 3}$. (See Fig. 12.)

$$\begin{aligned} t_{111} &= \frac{1}{60}, \quad t_{121} = \frac{1}{4}, \quad t_{112} = \frac{1}{6}, \quad t_{122} = \frac{1}{20}, \\ t_{211} &= \frac{1}{60}, \quad t_{221} = \frac{1}{5}, \quad t_{212} = \frac{17}{60}, \quad t_{222} = \frac{1}{60}. \end{aligned}$$

It is asymmetric but projectively symmetric. Note that T is forced to have multiple witnesses of asymmetry due to its projective symmetry. Indeed,

$$\begin{aligned} t_{121} - t_{112} &= t_{212} - t_{221} = \frac{5}{60}, \\ t_{121} - t_{211} &= t_{212} - t_{122} = \frac{14}{60}, \\ t_{112} - t_{211} &= t_{221} - t_{122} = \frac{9}{60}. \end{aligned}$$

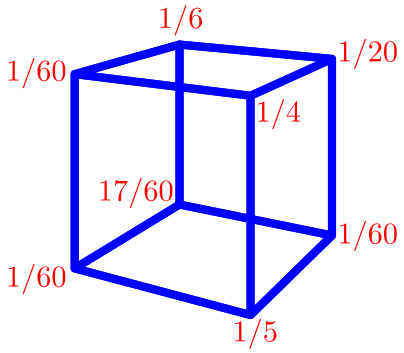


Figure 12 An asymmetric tensor $T \in \mathsf{Ten}_2^{\otimes 3}$ that is 2-projectively symmetric.

Therefore $\operatorname{asymm}(T) = \frac{14}{60} = \frac{7}{30}$, given by $t_{121} - t_{211}$ and $t_{212} - t_{122}$. All of its order-2 projections are given by

$$\begin{bmatrix} \frac{11}{60} & \frac{3}{10} \\ \frac{3}{10} & \frac{13}{60} \end{bmatrix}, \quad \begin{bmatrix} \frac{4}{15} & \frac{13}{60} \\ \frac{13}{60} & \frac{3}{10} \end{bmatrix}, \quad \begin{bmatrix} \frac{1}{30} & \frac{9}{20} \\ \frac{9}{20} & \frac{1}{15} \end{bmatrix}.$$

All of their margins are equal to $\begin{bmatrix} \frac{29}{60} \\ \frac{31}{60} \end{bmatrix}$.

In general, for any dimension-d order-L tensor, such examples can always be constructed due to the gap of degrees of freedom between the homogeneous linear systems $\mathcal{E}_1$ and $\mathcal{E}_2$.

14 Rethinking the converse

14.1 A cheap converse

If for a general $\mathcal{A} = (\mathcal{X}, \lambda_{\mathbf{x}}, \mathcal{S}, \lambda_{\mathbf{s}}, W_{\mathbf{y}|\mathbf{x},\mathbf{s}})$, for every $P_{\mathbf{x}} \in \lambda_{\mathbf{x}}$, the confusability set is a halfspace defined by a single linear constraint

$$\mathcal{K}^{\otimes L}(P_{\mathbf{x}}) \coloneqq \left\{ P_{\mathbf{x}_1,\cdots,\mathbf{x}_L} \in \mathcal{J}^{\otimes L}(P_{\mathbf{x}}) \colon \langle P_{\mathbf{x}_1,\cdots,\mathbf{x}_L}, C \rangle \leqslant b \right\},$$

for some tensor $C \in \mathsf{Ten}_{|\mathcal{X}|}^{\otimes L}$ and constant b, then the converse can be significantly simplified. In particular, we do not have to handle symmetric and asymmetric cases separately. We describe the proof idea below.

Proof. The proof essentially follow from the following observation. For any asymmetric $P_{\mathbf{x}_1,\cdots,\mathbf{x}_L}$, given any $P_{\mathbf{x}}$-constant composition $(\zeta, P_{\mathbf{x}_1,\cdots,\mathbf{x}_L})$-equicoupled code $\mathcal{C} = \{\underline{x}_i\}_{i=1}^M$ in $\mathcal{X}^n$ of size M, we can construct a code $\mathcal{C}' = \{\underline{x}'_i\}_{i=1}^M$ in $\mathcal{X}^{n \cdot M!}$ of the same size which is *symmetric*. Indeed, we can permute the rows of $\mathcal{C}$ using $\sigma \in S_M$ and juxtapose all possible ($M!$ of them in total) such row-permuted codes $\sigma(\mathcal{C})$. (See Fig. 13.) The resulting code $\mathcal{C}'$ is actually not only L-wise approximately equicoupled, but M-wise exactly equicoupled! For *any* $L \in [M]$ and any L-sized (not necessarily ordered) subset $\{i_1, \cdots, i_L\}$ of $[M]$, the joint type of $\underline{x}'_{i_1}, \cdots, \underline{x}'_{i_L}$ is *exactly* equal to

$$\tau_{\underline{x}'_{i_1},\cdots,\underline{x}'_{i_L}} = \frac{1}{\binom{M}{L}} \sum_{\{i_1,\cdots,i_L\} \in \binom{[M]}{L}} \frac{1}{L!} \sum_{\sigma \in S_L} \tau_{\underline{x}_{\sigma(i_1)},\cdots,\underline{x}_{\sigma(i_L)}},$$

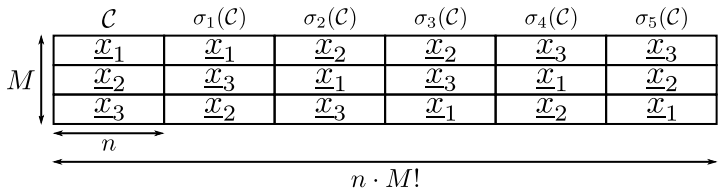


Figure 13 Construction of $\mathcal{C}'$ by permuting rows of $\mathcal{C} = \{\underline{x}_1, \underline{x}_2, \underline{x}_3\}$ using $\sigma \in S_3$ (where $S_3 = \{\mathrm{id}, \sigma_1, \cdots, \sigma_5\}$) and juxtaposing all $\sigma(\mathcal{C})$ (6 of them in total) together.

which is *symmetric* and independent of the choice of the list $(i_1, \cdots, i_L)$ (hence let us denote it by $\widehat{P}_{\mathbf{x}_1,\cdots,\mathbf{x}_L}$). In particular, letting $L = M$, we get that

$$\tau_{\underline{x}'_1,\cdots,\underline{x}'_M} = \frac{1}{M!} \sum_{\sigma \in S_M} \tau_{\underline{x}_{\sigma(1)},\cdots,\underline{x}_{\sigma(M)}}.$$

To see the above claims, note that if we juxtapose two pairs of codewords $(\underline{x}_1, \underline{x}_2)$ and $(\underline{x}'_1, \underline{x}'_2)$, we get a pair of longer codewords $(\widetilde{\underline{x}}_1, \widetilde{\underline{x}}_2) \coloneqq (\underline{x}_1 \circ \underline{x}'_1, \underline{x}_2 \circ \underline{x}'_2)$ (where $\circ$ denotes concatenation) with joint type

$$\tau_{\widetilde{\underline{x}}_1,\widetilde{\underline{x}}_2} = \frac{1}{2}(\tau_{\underline{x}_1,\underline{x}_2} + \tau_{\underline{x}'_1,\underline{x}'_2}).$$

This still holds if two pairs of codewords of different blocklengths are juxtaposed. Say, $(\underline{x}_1, \underline{x}_2)$ has blocklength n while $(\underline{x}'_1, \underline{x}'_2)$ has blocklength n'. Then

$$\tau_{\widetilde{\underline{x}}_1,\widetilde{\underline{x}}_2} = \frac{n}{n+n'}\tau_{\underline{x}_1,\underline{x}_2} + \frac{n'}{n+n'}\tau_{\underline{x}'_1,\underline{x}'_2}.$$

Back to the proof of the converse in such a spacial case, since the confusability set is defined by a single linear constraint, any convex combinations of non-confusable joint types is still outside the confusability set, in particular, $\widehat{P}_{\mathbf{x}_1,\cdots,\mathbf{x}_L}$. We hence reduce the problem to the symmetric case and the rest of the proof is handled by Theorem 65. ◀

14.2 Towards a unifying converse

We feel it unusual that we have to use drastically different techniques to prove the symmetric and the asymmetric parts of the converse. We suspect that it can be proved in a unifying way using the duality between CP and coP tensors which is the source of contradiction in our current proof of the symmetric case.

Note that the duality holds only in the space of *symmetric* tensors. To be specific, traditionally, CP and coP tensors are defined to be symmetric. And they are dual cones living in the ambient space $\mathsf{Sym}_n^{\otimes}$. If we extend the definitions of CP and coP tensors to the set of *all* (including asymmetric) tensors, then it is unclear whether duality still holds. Indeed, there are pairs of cones which are dual to each other in a certain ambient space but are no long dual in a larger ambient space. In a word, the ambient space that the dual cone is computed with respect to matters much.

We provide evidence showing that the symmetric and asymmetric parts of the converse can be potentially unified by the Plotkin-type bound since duality between CP and coP tensors–the core of the double counting argument–fortunately holds in larger generality.

Duality. We know that $\mathsf{CP}^{\otimes L}_{|\mathcal{X}|}$ and $\mathsf{coP}^{\otimes L}_{|\mathcal{X}|}$ are dual cones in the space $\mathsf{Sym}^{\otimes L}_{|\mathcal{X}|}$ of *symmetric* tensors. However, $\widehat{P}_{\mathbf{x}_1,\cdots,\mathbf{x}_L}$ (associated to the equicoupled subcode extracted using hypergraph Ramsey's theorem) is not guaranteed to be symmetric. We claim that duality still holds in the space $\mathsf{Ten}^{\otimes L}_{|\mathcal{X}|}$ of *all* tensors. Hence, copositive witness Q of a non-CP $\widehat{P}_{\mathbf{x}_1,\cdots,\mathbf{x}_L}$ exists even when $\widehat{P}_{\mathbf{x}_1,\cdots,\mathbf{x}_L}$ is asymmetric.

▷ Claim 75. $\mathsf{CP}^{\otimes L}_{|\mathcal{X}|}$ and $\mathsf{coP}^{\otimes L}_{|\mathcal{X}|}$ are dual cones in $\mathsf{Ten}^{\otimes L}_{|\mathcal{X}|}$.

Proof. By definition,

$$\left(\mathsf{CP}^{\otimes L}_{|\mathcal{X}|}\right)^* := \left\{B \in \mathsf{Ten}^{\otimes L}_{|\mathcal{X}|} \colon \forall A \in \mathsf{CP}^{\otimes L}_{|\mathcal{X}|},\, \langle A, B\rangle \geqslant 0\right\}.$$

Note that it is important that B is now taken from $\mathsf{Ten}^{\otimes L}_{|\mathcal{X}|}$ rather than $\mathsf{Sym}^{\otimes L}_{|\mathcal{X}|}$. Also recall that

$$\mathsf{coP}^{\otimes L}_{|\mathcal{X}|} := \left\{B \in \mathsf{Ten}^{\otimes L}_{|\mathcal{X}|} \colon \forall \underline{x} \in \mathbb{R}^{|\mathcal{X}|}_{\geqslant 0},\, \left\langle B, \underline{x}^{\otimes L}\right\rangle \geqslant 0\right\}.$$

Note that this definition *differs* from the standard one 95 and this cone is potentially larger.[16] The goal is to show $\left(\mathsf{CP}^{\otimes L}_{|\mathcal{X}|}\right)^* = \mathsf{coP}^{\otimes L}_{|\mathcal{X}|}$.

The direction $\mathsf{coP}_{|\mathcal{X}|} \subseteq \left(\mathsf{CP}^{\otimes L}_{|\mathcal{X}|}\right)^*$ is trivial, since the definitions of CP and coP tensors remain the same but the dual cone is computed w.r.t. a larger space. The new dual cone we are considering is no smaller than the old one. The inclusion that used to hold in the traditional setting should continue to hold now. Indeed, take any $B \in \mathsf{coP}^{\otimes L}_{|\mathcal{X}|}$, for any $A = \sum_i \underline{x}_i^{\otimes L} \in \mathsf{CP}^{\otimes L}_{|\mathcal{X}|}$, where $\underline{x}_i \in \mathbb{R}^{|\mathcal{X}|}_{\geqslant 0}$,

$$\langle A, B\rangle = \left\langle \sum_i \underline{x}_i^{\otimes L}, B \right\rangle = \sum_i \left\langle B, \underline{x}_i^{\otimes L}\right\rangle.$$

Since $B \in \mathsf{coP}^{\otimes L}_{|\mathcal{X}|}$, by definition, all $\left\langle B, \underline{x}_i^{\otimes L}\right\rangle$'s are non-negative, hence so is $\langle A, B\rangle$. Therefore $B \in \left(\mathsf{CP}^{\otimes L}_{|\mathcal{X}|}\right)^*$.

Now we show $\left(\mathsf{CP}^{\otimes L}_{|\mathcal{X}|}\right)^* \subseteq \mathsf{coP}^{\otimes L}_{|\mathcal{X}|}$. Take any $B \in \left(\mathsf{CP}^{\otimes L}_{|\mathcal{X}|}\right)^*$ and any $\underline{x} \in \mathbb{R}^{|\mathcal{X}|}_{\geqslant 0}$. Then $\left\langle B, \underline{x}^{\otimes L}\right\rangle \geqslant 0$, since $\underline{x}^{\otimes L} \in \mathsf{CP}^{\otimes L}_{|\mathcal{X}|}$ and $B \in \left(\mathsf{CP}^{\otimes L}_{|\mathcal{X}|}\right)^*$. This finishes the whole proof. ◁

▶ Remark 76. In general, duality does not necessarily hold in a larger ambient space. Namely, computing dual cone w.r.t. a larger space may result in a larger cone. For instance, $\mathsf{PSD}_{|\mathcal{X}|}$ cone is known to be self dual in $\mathsf{Sym}_{|\mathcal{X}|}$, i.e., $\mathsf{PSD}^*_{|\mathcal{X}|} = \mathsf{PSD}_{|\mathcal{X}|}$. However, in $\mathsf{Mat}_{|\mathcal{X}|}$, $\mathsf{PSD}^*_{|\mathcal{X}|}$ is strictly containing $\mathsf{PSD}_{|\mathcal{X}|}$. To see this, note that any skew symmetric matrix B is in $\mathsf{PSD}^*_{|\mathcal{X}|}$ since for any PSD (hence symmetric) matrix A, $\langle A, B\rangle = 0 \geqslant 0$; while B is not necessarily PSD.

Define, for $\sigma \in S_L$, $\sigma\left(P_{\mathbf{x}_1,\cdots,\mathbf{x}_L}\right) := P_{\mathbf{x}_{\sigma(1)},\cdots,\mathbf{x}_{\sigma(L)}}$. Though duality holds for all symmetric and asymmetric tensors, we do not have a full proof of the converse using duality, since we have trouble bounding the term

$$\left\langle \sigma\left(P_{\mathbf{x}_1,\cdots,\mathbf{x}_L}\right), Q\right\rangle = \left\langle P_{\mathbf{x}_1,\cdots,\mathbf{x}_L}, \sigma(Q)\right\rangle$$

which does not necessarily equal $\langle P_{\mathbf{x}_1,\cdots,\mathbf{x}_L}, Q\rangle$ for *asymmetric* Q.

We next show that such asymmetric witness Q does exist and is sometimes necessary in the sense that, some asymmetric (hence non-CP) tensors have *no* symmetric witness. This means that the dual cone of coP w.r.t. $\mathsf{Ten}^{\otimes L}_{|\mathcal{X}|}$ (instead of $\mathsf{Sym}^{\otimes L}_{|\mathcal{X}|}$) is strictly larger.

[16] Indeed, we will see shortly that it is strictly larger.

Asymmetric distributions without symmetric coP witness. Let $L = 2$. We construct an asymmetric self-coupling $P_{\mathbf{x}_1,\mathbf{x}_2} \in \Delta\left([3]^2\right)$ without symmetric coP witness Q such that $\langle P_{\mathbf{x}_1,\mathbf{x}_2}, Q\rangle < 0$. Indeed, let

$$P_{\mathbf{x}_1,\mathbf{x}_2} = \begin{bmatrix} \frac{4}{9} & \frac{7}{48} & \frac{11}{144} \\ \frac{3}{16} & \frac{1}{16} & 0 \\ \frac{5}{144} & \frac{1}{24} & \frac{1}{144} \end{bmatrix}.$$

Note that

$$P_{\mathbf{x}_1} = P_{\mathbf{x}_2} = \begin{bmatrix} \frac{2}{3} \\ \frac{1}{4} \\ \frac{1}{12} \end{bmatrix} =: P_{\mathbf{x}}.$$

Then

$$\frac{P_{\mathbf{x}_1,\mathbf{x}_2} + P_{\mathbf{x}_1,\mathbf{x}_2}^\top}{2} = \begin{bmatrix} \frac{4}{9} & \frac{1}{6} & \frac{1}{18} \\ \frac{1}{6} & \frac{1}{16} & \frac{1}{48} \\ \frac{1}{18} & \frac{1}{48} & \frac{1}{144} \end{bmatrix} = \begin{bmatrix} \frac{2}{3} \\ \frac{1}{4} \\ \frac{1}{12} \end{bmatrix} \begin{bmatrix} \frac{2}{3} & \frac{1}{4} & \frac{1}{12} \end{bmatrix} = P_{\mathbf{x}} P_{\mathbf{x}}^\top.$$

If there was a symmetric coP Q such that $\langle P_{\mathbf{x}_1,\mathbf{x}_2}, Q\rangle < 0$, then

$$\begin{aligned} \left\langle P_{\mathbf{x}} P_{\mathbf{x}}^\top, Q\right\rangle &= \frac{1}{2}\left(\langle P_{\mathbf{x}_1,\mathbf{x}_2}, Q\rangle + \left\langle P_{\mathbf{x}_1,\mathbf{x}_2}^\top, Q\right\rangle\right) \\ &= \frac{1}{2}\left(\langle P_{\mathbf{x}_1,\mathbf{x}_2}, Q\rangle + \left\langle P_{\mathbf{x}_1,\mathbf{x}_2}, Q^\top\right\rangle\right) \\ &= \langle P_{\mathbf{x}_1,\mathbf{x}_2}, Q\rangle < 0. \end{aligned}$$

However, $P_{\mathbf{x}} P_{\mathbf{x}}^\top$ is CP, so $\left\langle P_{\mathbf{x}} P_{\mathbf{x}}^\top, Q\right\rangle \geqslant 0$, which is a contradiction.

15 Sanity checks

Consider the bit-flip model.

In this section, we are going to verify the correctness of our characterization of the generalized Plotkin point using the bit-flip model as a running example. For $L = 3, 4$,[17] we will numerically recover Blinovsky's [9] characterization of the Plotkin point P_{L-1} for $(p, L-1)$-list decoding. In particular, $P_2 = 1/4$ and $P_3 = 5/16$.

15.1 $L = 3$

We first consider $(L-1)$-list decoding for $L - 1 = 2$, i.e., $L = 3$. It is known that the Plotkin point at $L - 1 = 2$ is $P_2 = 1/4$.

Fix any input distribution $P_{\mathbf{x}} := \mathrm{Bern}(w) = \begin{bmatrix} 1-w \\ w \end{bmatrix}$ for $0 < w < 1$. We first compute $\mathcal{J}^{\otimes 3}(P_{\mathbf{x}})$, $\mathcal{K}^{\otimes 3}(P_{\mathbf{x}})$. Let $p_{i,j,k,\ell} := P_{\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3,\mathbf{y}}(i,j,k,\ell)$ where $i,j,k,\ell \in \{0,1\}$.

$$\mathcal{J}^{\otimes 3}(P_{\mathbf{x}}) = \left\{P_{\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3} \in \Delta\left(\{0,1\}^3\right) : P_{\mathbf{x}_i} = P_{\mathbf{x}},\ i = 1,2,3\right\}$$

[17] For $L = 2$, i.e., the unique decoding case, the work [43] already recovers the classic Plotkin bound $P_1 = 1/4$.

$$= \left\{ P_{\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3} : \begin{array}{rl} p_{i,j,k} \geqslant & 0,\ i,j,k \in \{0,1\} \\ \sum_{i,j,k} p_{i,j,k} = & 1 \\ \sum_{i,j} p_{i,j,1} = & w \\ \sum_{i,k} p_{i,1,k} = & w \\ \sum_{j,k} p_{1,j,k} = & w \end{array} \right\}.$$

$$\begin{aligned} &\mathcal{K}^{\otimes 3}(P_{\mathbf{x}}) \\ &= \left\{ P_{\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3} = [P_{\mathbf{x}_2,\mathbf{x}_2,\mathbf{x}_3,\mathbf{y}}]_{\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3} \in \mathcal{J}^{\otimes 3}(P_{\mathbf{x}}) : \begin{array}{rl} P_{\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3,\mathbf{y}} \in & \Delta\left(\{0,1\}^4\right) \\ P_{\mathbf{x}_i,\mathbf{y}}(0,1) + P_{\mathbf{x}_i,\mathbf{y}}(1,0) \leqslant & p,\ i = 1,2,3 \end{array} \right\} \\ &= \left\{ [P_{\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3,\mathbf{y}}]_{\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3} \in \mathcal{J}^{\otimes 3}(P_{\mathbf{x}}) : \begin{array}{rl} p_{i,j,k,\ell} \geqslant & 0,\ i,j,k,\ell \in \{0,1\} \\ \sum_{i,j,k,\ell} p_{i,j,k,\ell} = & 1 \\ \sum_{j,k} p_{0,j,k,1} + p_{1,j,k,0} \leqslant & p \\ \sum_{i,k} p_{i,0,k,1} + p_{i,1,k,0} \leqslant & p \\ \sum_{i,j} p_{i,j,0,1} + p_{i,j,1,0} \leqslant & p \end{array} \right\}. \end{aligned}$$

$\widehat{\mathcal{J}}^{\otimes(L+1)}(P_{\mathbf{x}})$ and $\widehat{\mathcal{K}}^{\otimes(L+1)}(P_{\mathbf{x}})$ are extended formulations of $\mathcal{J}^{\otimes L}(P_{\mathbf{x}})$ and $\mathcal{K}^{\otimes L}(P_{\mathbf{x}})$, respectively.

$$\widehat{\mathcal{J}}^{\otimes 4}(P_{\mathbf{x}}) = \left\{ P_{\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3,\mathbf{y}} : \begin{array}{rl} p_{i,j,k,\ell} \geqslant & 0,\ i,j,k,\ell \in \{0,1\} \\ \sum_{i,j,k,\ell \in \{0,1\}} p_{i,j,k,\ell} = & 1 \\ \sum_{i,j} p_{i,j,1} = & w \\ \sum_{i,k} p_{i,1,k} = & w \\ \sum_{j,k} p_{1,j,k} = & w \end{array} \right\}.$$

$$\widehat{\mathcal{K}}^{\otimes 4}(P_{\mathbf{x}}) = \left\{ P_{\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3,\mathbf{y}} \in \widehat{\mathcal{J}}^{\otimes 4}(P_{\mathbf{x}}) : \begin{array}{rl} \sum_{j,k \in \{0,1\}} p_{0,j,k,1} + p_{1,j,k,0} \leqslant & p \\ \sum_{i,k \in \{0,1\}} p_{i,0,k,1} + p_{i,1,k,0} \leqslant & p \\ \sum_{i,j \in \{0,1\}} p_{i,j,0,1} + p_{i,j,1,0} \leqslant & p \end{array} \right\}.$$

To verify the value of Plotkin point P_{L-1} at $L = 3$, it suffices to verify that, if $w = 1/2$, then $P_{\mathbf{x}}^{\otimes 3} \notin \mathcal{K}^{\otimes 3}(P_{\mathbf{x}})$ iff $p < 1/4$, since we know that the optimizing input distribution when codewords are weight unconstrained is uniform. To this end, define a hyperplane

$$\mathcal{H}\left(P_{\mathbf{x}}^{\otimes 3}\right) := \left\{ P_{\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3,\mathbf{y}} \in \widehat{\mathcal{J}}^{\otimes 4}(P_{\mathbf{x}}) : [P_{\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3,\mathbf{y}}]_{\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3} = P_{\mathbf{x}}^{\otimes 3} \right\}.$$

Note that $P_{\mathbf{x}}^{\otimes 3} \notin \mathcal{K}^{\otimes 3}(P_{\mathbf{x}})$ is equivalent to $\mathcal{H}\left(P_{\mathbf{x}}^{\otimes 3}\right) \cap \widehat{\mathcal{K}}^{\otimes 4}(P_{\mathbf{x}}) = \varnothing$. Since $\mathcal{H}\left(P_{\mathbf{x}}^{\otimes 3}\right)$ depends on w and $\widehat{\mathcal{K}}^{\otimes 4}(P_{\mathbf{x}})$ depends on w, p, we write them as $\mathcal{H}(w)$ and $\widehat{\mathcal{K}}^{\otimes 4}(w,p)$, respectively, for simplicity.

We claim that the Plotkin point P_{L-1} is precisely the optimal value of the following LP, i.e., the smallest p^* such that the hyperplane $\mathcal{H}(1/2)$ has no intersection with the corresponding high-dimensional polytope $\widehat{\mathcal{K}}^{\otimes 4}(1/2, p^*)$.

$$\begin{array}{rl} \min & p \\ \text{subject to} & \mathcal{H}(1/2) \cap \widehat{\mathcal{K}}^{\otimes 4}(1/2,p) \neq \varnothing. \end{array}$$

Equivalently, collecting all constraints together, we want to find the minimal p so that the polytope (the feasible region of the LP) defined by the following constraints is nonempty.

$$P_{\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3,\mathbf{y}} \in \widehat{\mathcal{J}}^{\otimes 4}(P_{\mathbf{x}})$$

$$
\begin{aligned}
[P_{\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3,\mathbf{y}}]_{\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3} =& P_{\mathbf{x}}^{\otimes 3}\\
\sum_{j,k\in\{0,1\}} p_{0,j,k,1} + p_{1,j,k,0} \leqslant& p\\
\sum_{i,k\in\{0,1\}} p_{i,0,k,1} + p_{i,1,k,0} \leqslant& p\\
\sum_{i,j\in\{0,1\}} p_{i,j,0,1} + p_{i,j,1,0} \leqslant& p.
\end{aligned}
$$

Expanding everything out and noting that the first constraint regarding constant composition $P_{\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3,\mathbf{y}\in\hat{\mathcal{J}}^{\otimes 4}(P_{\mathbf{x}})}$ is redundant since it is the same as the constraint $[P_{\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3,\mathbf{y}}]_{\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3} = P_{\mathbf{x}}^{\otimes 3} \in \mathcal{J}^{\otimes 3}(P_{\mathbf{x}})$, we simplify the defining (in)equalities of the polytope as follows,

$$
\begin{aligned}
p_{i,j,k,\ell} \geqslant& 0,\ i,j,k,\ell \in \{0,1\}\\
\sum_{i,j,k,\ell\in\{0,1\}} p_{i,j,k,\ell} =& 1\\
p_{i,j,k,0} + p_{i,j,k,1} =& 1/8,\ i,j,k \in \{0,1\}\\
\sum_{j,k\in\{0,1\}} p_{0,j,k,1} + p_{1,j,k,0} \leqslant& p\\
\sum_{i,k\in\{0,1\}} p_{i,0,k,1} + p_{i,1,k,0} \leqslant& p\\
\sum_{i,j\in\{0,1\}} p_{i,j,0,1} + p_{i,j,1,0} \leqslant& p,
\end{aligned}
$$

since $P_{\mathbf{x}}^{\otimes 3}(i,j,k) = P_{\mathbf{x}}(i)P_{\mathbf{x}}(j)P_{\mathbf{x}}(k) = 1/8$ for all $i,j,k \in \{0,1\}$.

Let

$$\underline{p} \coloneqq \begin{bmatrix} p_{0,0,0,0} & \cdots & p_{1,1,1,1} \end{bmatrix}^\top.$$

The LP can be written in a compact form as

$$
\begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1\\
1 & 1 & & & & & & & & & & & & & & \\
 & & 1 & 1 & & & & & & & & & & & & \\
 & & & & 1 & 1 & & & & & & & & & & \\
 & & & & & & 1 & 1 & & & & & & & & \\
 & & & & & & & & 1 & 1 & & & & & & \\
 & & & & & & & & & & 1 & 1 & & & & \\
 & & & & & & & & & & & & 1 & 1 & & \\
 & & & & & & & & & & & & & & 1 & 1
\end{bmatrix}
\underline{p} =
\begin{bmatrix} 1\\1/8\\1/8\\1/8\\1/8\\1/8\\1/8\\1/8\\1/8 \end{bmatrix},
$$

$$
\begin{bmatrix}
1 & & 1 & & 1 & & 1 & 1 & & 1 & & 1 & & 1 & \\
1 & & 1 & 1 & & 1 & & & 1 & & 1 & 1 & & 1 & \\
1 & 1 & & & 1 & 1 & & & 1 & 1 & & & 1 & 1 &
\end{bmatrix}
\underline{p} \leqslant
\begin{bmatrix} p\\p\\p \end{bmatrix}
$$

$$\underline{p} \geqslant \underline{0}.$$

Observe that as p increases, the linear system becomes monotonically easier to be satisfied. Checked by `Mathematica`, the above LP is feasible if $p > 1/4$ (and hence the distribution $\begin{bmatrix}1/2\\1/2\end{bmatrix}^{\otimes 3}$ is confusable) and is infeasible if $p < 1/4$ (and hence $\begin{bmatrix}1/2\\1/2\end{bmatrix}^{\otimes 3}$ is not confusable). Therefore, the $(p, L-1)$-list decoding capacity hits 0 precisely at $p = 1/4$.

15.2 $L = 4$

For $L = 4$, one can obtain a similar LP whose infeasibility is equivalent to $\mathcal{H}\left(\begin{bmatrix}1/2\\1/2\end{bmatrix}^{\otimes 4}\right)$ and $\widehat{\mathcal{K}}^{\otimes 5}\left(\begin{bmatrix}1/2\\1/2\end{bmatrix}^{\otimes 4}, p\right)$ being disjoint.

$$\begin{bmatrix}
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\
1&1&&&&&&&&&&&&&&&&&&&&&&&&&&&&\\
&&1&1&&&&&&&&&&&&&&&&&&&&&&&&&&\\
&&&&1&1&&&&&&&&&&&&&&&&&&&&&&&&\\
&&&&&&1&1&&&&&&&&&&&&&&&&&&&&&&\\
&&&&&&&&1&1&&&&&&&&&&&&&&&&&&&&\\
&&&&&&&&&&1&1&&&&&&&&&&&&&&&&&&\\
&&&&&&&&&&&&1&1&&&&&&&&&&&&&&&&\\
&&&&&&&&&&&&&&1&1&&&&&&&&&&&&&&\\
&&&&&&&&&&&&&&&&1&1&&&&&&&&&&&&\\
&&&&&&&&&&&&&&&&&&1&1&&&&&&&&&&\\
&&&&&&&&&&&&&&&&&&&&1&1&&&&&&&&\\
&&&&&&&&&&&&&&&&&&&&&&1&1&&&&&&\\
&&&&&&&&&&&&&&&&&&&&&&&&1&1&&&&\\
&&&&&&&&&&&&&&&&&&&&&&&&&&1&1&&\\
&&&&&&&&&&&&&&&&&&&&&&&&&&&&1&1
\end{bmatrix}\underline{p} = \begin{bmatrix}1\\1/16\\1/16\\1/16\\1/16\\1/16\\1/16\\1/16\\1/16\\1/16\\1/16\\1/16\\1/16\\1/16\\1/16\\1/16\\1/16\end{bmatrix},$$

$$\begin{bmatrix}
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\
1&1&1&1&1&1&1&1&&&&&&&&\\
1&1&1&1&&&&&1&1&1&1&&&&\\
1&1&&&1&1&&&1&1&&&1&1&&\\
&&1&1&&&1&1&&&1&1&&&1&1
\end{bmatrix}\underline{p} \leqslant \begin{bmatrix}p\\p\\p\\p\end{bmatrix},$$

$$\underline{p} \geqslant \underline{0},$$

where

$$\underline{p} = \begin{bmatrix}p_{0,0,0,0,0} & \cdots & p_{1,1,1,1,1}\end{bmatrix}^{\top}.$$

One can numerically check that the above LP is feasible if $p > 5/16$ and infeasible otherwise.

In general, to check whether

$$\mathcal{H}\left(\begin{bmatrix}1/2\\1/2\end{bmatrix}^{\otimes L}\right) \cap \widehat{\mathcal{K}}^{\otimes(L+1)}\left(\begin{bmatrix}1/2\\1/2\end{bmatrix}^{\otimes L}, p\right)$$

is empty, it boils down to checking the infeasibility of a linear program with 2^{L+1} variables and $2^{L+1} + 1 + 2^L + L$ constraints, 2^{L+1} of them for non-negativity of probability mass, 1 of them for probability mass summing up to one, 2^L of them for ensuring that $P_{\mathbf{x}_1,\cdots,\mathbf{x}_L} \in \mathcal{J}^{\otimes L}(P_{\mathbf{x}})$ is a $(P_{\mathbf{x}}, L)$-self-coupling, L of them for the non-confusability guarantee: $P_{\mathbf{x}_1,\cdots,\mathbf{x}_L} \notin \mathcal{K}^{\otimes L}(P_{\mathbf{x}})$. The size of the program (or the number of defining constraints of the corresponding polytope) grows exponentially in L. However, since we are concerned with absolute constant L in this paper, for any given L, the feasibility can be certified in constant time. Observe that, since the LP in the bit-flip setting is so structured, one can write it down explicitly by hand for any given L.

16 Blinovsky [9] revisited

In this section, we *fully* recover Blinovsky's [9] results on characterization of the Plotkin points P_{L-1} for $(p, L-1)$-list decoding under the bit-flip model.

Let ϕ be the standard bijection between $\{0,1\}$ and $\{-1,1\}$,

$$\begin{array}{rccc}
\phi\colon & \mathbb{F}_2 & \to & \{-1,1\}\\
& 0 & \mapsto & 1\\
& 1 & \mapsto & -1.
\end{array}$$

We identify the type $\tau_{\underline{x}} \in \mathcal{P}^{(n)}(\mathbb{F}_2)$ of a binary length-n vector $\underline{x} \in \mathbb{F}_2^n$ using a $\{-1,1\}$-valued random variable $\mathbf{x}$ defined as

$$\Pr[\mathbf{x}=-1] = \frac{wt_{\mathrm{H}}(\underline{x})}{n}, \quad \Pr[\mathbf{x}=1] = 1 - \frac{wt_{\mathrm{H}}(\underline{x})}{n}.$$

Indeed the distribution $P_{\mathbf{x}} \in \mathcal{P}^{(n)}(\{-1,1\})$ of $\mathbf{x}$ is the type of the image $\phi(\underline{x})$ of $\underline{x}$ under ϕ.

$$P_{\mathbf{x}}(\phi(0)) = \tau_{\underline{x}}(0), \quad P_{\mathbf{x}}(\phi(1)) = \tau_{\mathbf{x}}(1).$$

For a collection of vectors $\underline{x}_1, \cdots, \underline{x}_k \in \mathbb{F}_2^n$, their joint type is now represented by a sequence of random variables $\mathbf{x}_1, \cdots, \mathbf{x}_k$ with joint distribution $P_{\mathbf{x}_1,\cdots,\mathbf{x}_k}$, for any $x_1, \cdots, x_k \in \{-1,1\}$,

$$\begin{aligned} P_{\mathbf{x}_1,\cdots,\mathbf{x}_k}(x_1,\cdots,x_k) =& \Pr[\mathbf{x}_1 = x_1, \cdots, \mathbf{x}_k = x_k] \\ =& \tau_{\underline{x}_1,\cdots,\underline{x}_k}(\phi^{-1}(x_1), \cdots, \phi^{-1}(x_k)). \end{aligned}$$

It is easy to check that, for $\underline{x}_1, \underline{x}_2 \in \mathbb{F}_2^n$,

$$\frac{d_{\mathrm{H}}(\underline{x}_1, \underline{x}_2)}{n} = \frac{1}{2}\left(1 - \mathop{\mathbb{E}}_{(\mathbf{x}_1,\mathbf{x}_2)\sim P_{\mathbf{x}_1,\mathbf{x}_2}}[\mathbf{x}_1\mathbf{x}_2]\right). \tag{102}$$

Indeed

$$\begin{aligned} \text{RHS} =& \frac{1}{2}\left(1 - \tau_{\underline{x}_1,\underline{x}_2}(0,1)\cdot(-1) - \tau_{\underline{x}_1,\underline{x}_2}(1,0)\cdot(-1) - \tau_{\underline{x}_1,\underline{x}_2}(0,0)\cdot 1 - \tau_{\underline{x}_1,\underline{x}_2}(1,1)\cdot 1\right) \\ =& \frac{1}{2}\left(1 + \frac{d_{\mathrm{H}}(\underline{x}_1,\underline{x}_2)}{n} - \left(1 - \frac{d_{\mathrm{H}}(\underline{x}_1,\underline{x}_2)}{n}\right)\right) \\ =& \text{LHS}. \end{aligned}$$

Let

$$r := \mathop{\mathbb{E}}_{(\mathbf{x}_1,\cdots,\mathbf{x}_L)\sim\{-1,1\}^L}\left[|\mathbf{x}_1 + \cdots + \mathbf{x}_L|\right], \tag{103}$$

be the expected translation distance of a 1-dimensional unbiased random walk after L steps. Each $\mathbf{x}_i$ $(1 \leqslant i \leqslant L)$ is independent and uniformly distributed on $\{-1,1\}$.

▶ **Theorem 77.** *The Plotkin point P_{L-1} for $(p, L-1)$-list decoding is given by*

$$P_{L-1} = \frac{1 - r/L}{2}.$$

▶ Remark 78. Note that the formula in Theorem 77 agrees with the one by Blinovsky. To see this, we first compute r. For odd $L = 2k+1$, where $k \in \mathbb{Z}_{>0}$ is some strictly positive integer, it is easy to see that

$$\begin{aligned} r =& \mathbb{E}\left[|\mathbf{x}_1 + \cdots + \mathbf{x}_L|\right] \\ =& \sum_{i=0}^{k} \frac{2\binom{L}{i}}{2^L}(L-2i). \end{aligned}$$

Recall that, by binomial theorem (Fact (18)),

$$2^L = \sum_{i=0}^{L}\binom{L}{i} = \sum_{i=0}^{k} 2\binom{L}{i}.$$

Now we simplify the formula in Theorem 77.

$$\begin{aligned}
P_{L-1} =& \frac{1}{2} - \frac{r}{2L} \\
=& \sum_{i=0}^{k} \frac{\binom{L}{i}}{2^L} - \sum_{i=0}^{k} \left(1 - \frac{2i}{L}\right) \frac{\binom{L}{i}}{2^L} \\
=& \sum_{i=0}^{k} \frac{2i}{L} \frac{\binom{L}{i}}{2^L} \\
=& \sum_{i=1}^{k} \frac{i}{L} \frac{\frac{L}{i}\binom{L-1}{i-1}}{2^{L-1}} && (104) \\
=& \frac{1}{2^{L-1}} \sum_{i=0}^{k-1} \binom{L-1}{i} \\
=& \frac{1}{2^{L-1}} \frac{1}{2} \left(2^{L-1} - \binom{L-1}{k}\right) && (105) \\
=& \frac{1}{2} - 2^{-L} \binom{2k}{k},
\end{aligned}$$

where Eqn. (104) is by Fact (16); Eqn. (105) follows from binomial theorem (Fact (18)) again,

$$2^{L-1} = \binom{2k}{k} + 2 \sum_{i=0}^{k-1} \binom{2k}{i}.$$

▶ **Lemma 79** (Lower bound). *The Plotkin point P_{L-1} for $(p, L-1)$-list decoding is lower bounded by*

$$P_{L-1} \geqslant \frac{1 - r/L}{2}.$$

That is, if $p < P_{L-1}$, then the $(p, L-1)$-list decoding capacity is positive, i.e., there is an infinite sequence of $(p, L-1)$-list decodable codes of positive rate.

Proof. We will show that if $p = \frac{1 - \frac{r+\eta}{L}}{2} < \frac{1-r/L}{2}$ for any $\eta > 0$, then the product distribution $\mathrm{Bern}^{\otimes L}(1/2)$ lies outside the corresponding confusability set $\mathcal{K}^{\otimes L}(\mathrm{Bern}(1/2))$. Using the framework developed in this paper, a random code of a suitable positive rate in which each codeword is sampled independently and uniformly from $\mathcal{T}_{\underline{\mathbf{x}}}(\mathrm{Bern}(1/2))$ is $(p, L-1)$-list decodable w.h.p.

The proof is by contradiction. If $P_{\mathbf{x}_1,\cdots,\mathbf{x}_L} := \mathrm{Bern}^{\otimes L}(1/2)$ is confusable, then, by the definition 37 of confusability of tuples, an L-tuple of distinct codewords $\underline{x}_1, \cdots, \underline{x}_L$ of joint type $\tau_{\underline{x}_1,\cdots,\underline{x}_L} = P_{\mathbf{x}_1,\cdots,\mathbf{x}_L}$ can be covered by a ball of radius np centered around some $\underline{y} \in \mathbb{F}_2^n$. Equivalently, by the definition 38 of confusability of distributions, there is a refinement $P_{\mathbf{x}_1,\cdots,\mathbf{x},\mathbf{y}} \in \Delta\left(\{-1,1\}^{L+1}\right)$ such that $[P_{\mathbf{x}_1,\cdots,\mathbf{x}_L,\mathbf{y}}]_{\mathbf{x}_1,\cdots,\mathbf{x}_L} = P_{\mathbf{x}_1,\cdots,\mathbf{x}_L}$, and for every $i \in [L]$,

$$P_{\mathbf{x}_i,\mathbf{y}}(0,1) + P_{\mathbf{x}_i,\mathbf{y}}(1,0) \leqslant p.$$

This means that for every $i \in [L]$,

$$\mathbb{E}[\mathbf{x}_i \mathbf{y}] \geqslant \frac{r+\eta}{L},$$

by the relation (Eqn. (102)) between Hamming distance between vectors and correlation of their random variable representations. Hence

$$\mathbb{E}\left[(\mathbf{x}_1 + \cdots + \mathbf{x}_L)\,\mathbf{y}\right] \geqslant r + \eta. \tag{106}$$

The $\{-1,1\}$-valued random variable $\mathbf{y}$ that has the largest correlation with $\mathbf{x}_1 + \cdots + \mathbf{x}_L$ is $\mathbf{y} = \mathsf{MAJ}\left(\mathbf{x}_1, \cdots, \mathbf{x}_L\right)$, where

$$\begin{array}{rccc} \mathsf{MAJ}: & \{-1,1\}^L & \to & \{-1,1\} \\ & (x_1, \cdots, x_L) & \mapsto & \operatorname{sgn}\left(x_1 + \cdots + x_L\right). \end{array}$$

is the majority function. To see this, just expand the above expectation,

$$\begin{aligned} \mathbb{E}\left[(\mathbf{x}_1 + \cdots + \mathbf{x}_L)\,\mathbf{y}\right] &= \sum_{x_1, \cdots, x_L, y \in \{-1,1\}} P_{\mathbf{x}_1, \cdots, \mathbf{x}_L, \mathbf{y}}(x_1, \cdots, x_L, y)(x_1 + \cdots + x_L)y \\ &= \sum_{x_1, \cdots, x_L \in \{-1,1\}} P_{\mathbf{x}_1, \cdots, \mathbf{x}_L}(x_1, \cdots, x_L) \\ &\qquad \sum_{y \in \{-1,1\}} P_{\mathbf{y}|\mathbf{x}_1, \cdots, \mathbf{x}_L}(y|x_1, \cdots, x_L)(x_1 + \cdots + x_L)y. \end{aligned}$$

Note that, each summand

$$P_{\mathbf{y}|\mathbf{x}_1, \cdots, \mathbf{x}_L}(1|x_1, \cdots, x_L)(x_1 + \cdots + x_L) - P_{\mathbf{y}|\mathbf{x}_1, \cdots, \mathbf{x}_L}(-1|x_1, \cdots, x_L)(x_1 + \cdots + x_L)$$

is maximized when the conditional probability mass of $\mathbf{y}$ is concentrated on the singleton $\operatorname{sgn}(x_1 + \cdots + x_L)$,

$$P_{\mathbf{y}|\mathbf{x}_1, \cdots, \mathbf{x}_L}(\operatorname{sgn}(x_1 + \cdots + x_L)|x_1, \cdots, x_L) = 1, P_{\mathbf{y}|\mathbf{x}_1, \cdots, \mathbf{x}_L}(-\operatorname{sgn}(x_1 + \cdots + x_L)|x_1, \cdots, x_L) = 0.$$

In this case, each summand attains its maxima

$$\operatorname{sgn}(x_1 + \cdots + x_L)(x_1 + \cdots + x_L) = |x_1 + \cdots + x_L|.$$

Overall, the corresponding maximal correlation is precisely

$$\begin{aligned} &\mathbb{E}(\mathbf{x}_1 + \cdots + \mathbf{x}_L)\mathsf{MAJ}(\mathbf{x}_1, \cdots, \mathbf{x}_L) \\ &= \sum_{x_1, \cdots, x_L \in \{-1,1\}} P_{\mathbf{x}_1, \cdots, \mathbf{x}_L}(x_1, \cdots, x_L)\,|x_1 + \cdots + x_L| \\ &= \mathop{\mathbb{E}}_{(\mathbf{x}_1, \cdots, \mathbf{x}_L) \sim P_{\mathbf{x}_1, \cdots, \mathbf{x}_L}} \left[|\mathbf{x}_1 + \cdots + \mathbf{x}_L|\right]. \end{aligned} \tag{107}$$

Using the above observation, we get

$$\begin{aligned} r &= \mathop{\mathbb{E}}_{(\mathbf{x}_1, \cdots, \mathbf{x}_L) \sim \{-1,1\}^L} \left[|\mathbf{x}_1 + \cdots + \mathbf{x}_L|\right] && (108) \\ &= \mathbb{E}\left[(\mathbf{x}_1 + \cdots + \mathbf{x}_L)\,\mathsf{MAJ}\left(\mathbf{x}_1, \cdots, \mathbf{x}_L\right)\right] && (109) \\ &\geqslant r + \eta, && (110) \end{aligned}$$

Eqn. (108) is by the definition of r (Eqn. (103)). Eqn. (109) follows from Eqn. (107). Eqn. (110) is by Eqn. (110). We hence reach a contradiction which finishes the proof. ◀

▶ **Lemma 80** (Upper bound). *The Plotkin point P_{L-1} for $(p, L-1)$-list decoding is upper bounded by*

$$P_{L-1} \leqslant \frac{1 - r/L}{2}.$$

That is, if $p > P_{L-1}$, then no positive rate is possible, i.e, there is no infinite sequence of $(p, L-1)$-list decodable codes of positive rate.

Proof. Our goal is to show that if $p > P_{L-1}$, then $C_{L-1} = 0$. Suppose $p = \frac{1-\frac{r-\eta}{L}}{2}$ for a constant $\eta > 0$.

We are going to show that any infinite sequence of codes $\mathcal{C}_n$ each of positive rate is not $(p, L-1)$-list decodable. First, by the previous argument in last section, we can extract a sequence of subcodes $\mathcal{C}'_n \subseteq \mathcal{C}_n$ of positive rate satisfying that, for every tuple of *distinct* codewords $\underline{x}_1, \cdots, \underline{x}_L \in \mathcal{C}'$ and $x_1, \cdots, x_L \in \mathbb{F}_2$,

$$\left|\tau_{\underline{x}_1,\cdots,\underline{x}_L}(x_1,\cdots,x_L) - \widehat{P}_{\mathbf{x}_1,\cdots,\mathbf{x}_L}(x_1,\cdots,x_L)\right| \leqslant \zeta$$

for some *symmetric* distribution $\widehat{P}_{\mathbf{x}_1,\cdots,\mathbf{x}_L} \in \Delta\left(\mathcal{X}^L\right)$ and some positive constant $\zeta > 0$. In favour of the proceeding calculations, it suffices to take

$$\zeta = \frac{L}{(L-1)r2^{L+2}}\eta. \tag{111}$$

To show non-list decodability of $\mathcal{C}'$ (and hence $\mathcal{C}$), we will argue that there is a list $(\underline{x}_{i_1}, \cdots, \underline{x}_{i_L}) \in \binom{\mathcal{C}'}{L}$ that can be covered by a ball of radius np centered around the point $\mathsf{MAJ}\left(\underline{x}_{i_1}, \cdots, \underline{x}_{i_L}\right)$. The proof is by contradiction. Suppose this is not the case, i.e., no list can be covered by the ball centered at its majority. Define, for $(i_1, \cdots, i_L) \in \left[2^{nR}\right]^L$,

$$Q_{i_1,\cdots,i_L} = (\mathbf{x}_{i_1} + \cdots + \mathbf{x}_{i_L}) \cdot \mathsf{MAJ}\left(\mathbf{x}_{i_1}, \cdots, \mathbf{x}_{i_L}\right) - r.$$

We will provide a *strictly negative* upper bound and a *non-negative* lower bound on

$$Q \coloneqq \mathop{\mathbb{E}}_{(\mathbf{i}_1,\cdots,\mathbf{i}_L)\sim[2^{nR}]^L} \mathop{\mathbb{E}}_{(\mathbf{x}_{\mathbf{i}_1},\cdots,\mathbf{x}_{\mathbf{i}_L})\sim P_{\mathbf{x}_{\mathbf{i}_1},\cdots,\mathbf{x}_{\mathbf{i}_L}}} [Q_{\mathbf{i}_1,\cdots,\mathbf{i}_L}],$$

which is a contradiction and finishes the proof.

Upper bound on Q. By the assumption of list decodability, for every L-tuple of distinct codewords $\underline{x}_1, \cdots, \underline{x}_L \in \mathcal{C}'$, there is a codeword $\underline{x}_i$ ($i \in [L]$) among them such that

$$d_{\mathrm{H}}\left(\underline{x}_i, \mathsf{MAJ}\left(\underline{x}_1, \cdots, \underline{x}_L\right)\right) \geqslant np.$$

Equivalently,

$$\mathbb{E}\left[\mathbf{x}_i \mathsf{MAJ}\left(\mathbf{x}_1, \cdots, \mathbf{x}_L\right)\right] \leqslant \frac{r-\eta}{L}.$$

Since $\widehat{P}_{\mathbf{x}_1,\cdots,\mathbf{x}_L}$ is symmetric and $\mathcal{C}'$ is $\left(\zeta, \widehat{P}_{\mathbf{x}_1,\cdots,\mathbf{x}_L}\right)$-equicoupled, we expect that the term $\mathbb{E}\left[\mathbf{x}_j \mathsf{MAJ}\left(\mathbf{x}_1, \cdots, \mathbf{x}_L\right)\right] \lesssim \frac{r-\eta}{L}$ for all $j \in [L]$, potentially with some slack depending on ζ. Indeed, for any $j \in [L] \backslash \{i\}$ (without loss of generality, assume $j > i$),

$$\begin{aligned}
&\left|\mathbb{E}\left[\mathbf{x}_i \mathsf{MAJ}\left(\mathbf{x}_1, \cdots, \mathbf{x}_L\right)\right] - \mathbb{E}\left[\mathbf{x}_j \mathsf{MAJ}\left(\mathbf{x}_1, \cdots, \mathbf{x}_L\right)\right]\right| \\
=&\left| \sum_{x_1,\cdots,x_L \in \{-1,1\}} \tau_{\underline{x}_1,\cdots,\underline{x}_L}(\phi^{-1}(x_1), \cdots, \phi^{-1}(x_L)) x_i \mathsf{MAJ}\left(x_1, \cdots, x_L\right) \right. \\
&\left. - \sum_{x_1,\cdots,x_L \in \{-1,1\}} \tau_{\underline{x}_1,\cdots,\underline{x}_L}(\phi^{-1}(x_1), \cdots, \phi^{-1}(x_L)) x_j \mathsf{MAJ}\left(x_1, \cdots, x_L\right) \right| \\
=&\left| \sum_{x_1,\cdots,x_L \in \{-1,1\}} \tau_{\underline{x}_1,\cdots,\underline{x}_L}(\phi^{-1}(x_1), \cdots, \phi^{-1}(x_L)) x_i \mathsf{MAJ}\left(x_1, \cdots, x_L\right) \right.
\end{aligned} \tag{112}$$

$$
\begin{aligned}
&- \sum_{x_{\sigma(1)},\cdots,x_{\sigma(L)}\in\{-1,1\}} \tau_{\underline{x}_1,\cdots,\underline{x}_L}(\phi^{-1}(x_{\sigma(1)}),\cdots,\phi^{-1}(x_{\sigma(L)}))x_{\sigma(j)}\mathsf{MAJ}\left(x_{\sigma(1)},\cdots,x_{\sigma(L)}\right)\Bigg| \qquad (113)\\
=&\Bigg|\sum_{x_1,\cdots,x_L\in\{-1,1\}} \tau_{\underline{x}_1,\cdots,\underline{x}_L}(\phi^{-1}(x_1),\cdots,\phi^{-1}(x_L))x_i\mathsf{MAJ}\left(x_1,\cdots,x_L\right)\\
&- \sum_{x_1,\cdots,x_L\in\{-1,1\}} \tau_{\underline{x}_1,\cdots,\underline{x}_L}(\phi^{-1}(x_{\sigma(1)}),\cdots,\phi^{-1}(x_{\sigma(L)}))x_i\mathsf{MAJ}\left(x_1,\cdots,x_L\right)\Bigg|\\
=&\left|\sum_{x_1,\cdots,x_L\in\{-1,1\}}\left[\begin{array}{c}\left(\begin{array}{c}\tau_{\underline{x}_1,\cdots,\underline{x}_L}(\phi^{-1}(x_1),\cdots,\phi^{-1}(x_L))\\ -\widehat{P}_{\mathbf{x}_1,\cdots,\mathbf{x}_L}(\phi^{-1}(x_1),\cdots,\phi^{-1}(x_L))\end{array}\right)\\ +\left(\begin{array}{c}\widehat{P}_{\mathbf{x}_1,\cdots,\mathbf{x}_L}(\phi^{-1}(x_{\sigma(1)}),\cdots,\phi^{-1}(x_{\sigma(L)}))\\ -\tau_{\underline{x}_1,\cdots,\underline{x}_L}(\phi^{-1}(x_{\sigma(1)}),\cdots,\phi^{-1}(x_\sigma(L)))\end{array}\right)\end{array}\right]x_i\mathsf{MAJ}\left(x_1,\cdots,x_L\right)\right| \qquad (114)\\
\leqslant&\left(\begin{array}{c}\left|\begin{array}{c}\tau_{\underline{x}_1,\cdots,\underline{x}_L}(\phi^{-1}(x_1),\cdots,\phi^{-1}(x_L))\\ -\widehat{P}_{\mathbf{x}_1,\cdots,\mathbf{x}_L}(\phi^{-1}(x_1),\cdots,\phi^{-1}(x_L))\end{array}\right|\\ +\left|\begin{array}{c}\widehat{P}_{\mathbf{x}_1,\cdots,\mathbf{x}_L}(\phi^{-1}(x_{\sigma(1)}),\cdots,\phi^{-1}(x_{\sigma(L)}))\\ -\tau_{\underline{x}_1,\cdots,\underline{x}_L}(\phi^{-1}(x_{\sigma(1)}),\cdots,\phi^{-1}(x_\sigma(L)))\end{array}\right|\end{array}\right)\left|\sum_{x_1,\cdots,x_L\in\{-1,1\}} x_i\mathsf{MAJ}\left(x_1,\cdots,x_L\right)\right| \qquad (115)\\
\leqslant&2\zeta\cdot\frac{2^L}{L}\mathop{\mathbb{E}}_{(\mathbf{x}_1,\cdots,\mathbf{x}_L)\sim\{-1,1\}^L}\left[(\mathbf{x}_1+\cdots+\mathbf{x}_L)\,\mathsf{MAJ}\left(\mathbf{x}_1,\cdots,\mathbf{x}_L\right)\right] \qquad (116)\\
=&\frac{2^{L+1}}{L}\zeta\mathop{\mathbb{E}}_{(\mathbf{x}_1,\cdots,\mathbf{x}_L)\sim\{-1,1\}^L}\left[|\mathbf{x}_1+\cdots+\mathbf{x}_L|\right]\\
=&\frac{2^{L+1}r}{L}\zeta. \qquad (117)
\end{aligned}
$$

In the above chain of equalities and inequalities, we used the following facts.

1. In Eqn. (113), $\sigma\in S_L$ denotes the transposition which swaps the i-th and j-th element,
$$\sigma=\begin{pmatrix}1&\cdots&i-1&i&i+1&\cdots&j-1&j&j+1&\cdots&L\\1&\cdots&i-1&j&i+1&\cdots&j-1&i&j+1&\cdots&L\end{pmatrix}.$$
2. Eqn. (114) is due to symmetry of $\widehat{P}_{\mathbf{x}_1,\cdots,\mathbf{x}_L}$.
3. Inequality (115) is by triangle inequality of absolute value.
4. Eqn. (116) follows since
$$\left|\sum_{x_1,\cdots,x_L\in\{-1,1\}} x_i\mathsf{MAJ}\left(x_1,\cdots,x_L\right)\right|=2^L\left|\frac{1}{L}\sum_{i=1}^{L}\sum_{x_1,\cdots,x_L\in\{-1,1\}}\frac{1}{2^L}x_i\mathsf{MAJ}\left(x_1,\cdots,x_L\right)\right|,$$
and the expectation is over $\mathbf{x}_i$'s which are independent and uniformly distributed on $\{-1,1\}$.

Now, for any $j\in[L]\backslash\{i\}$,

$$
\begin{aligned}
\mathbb{E}\left[\mathbf{x}_j\mathsf{MAJ}\left(\mathbf{x}_1,\cdots,\mathbf{x}_L\right)\right]=&\mathbb{E}\left[\mathbf{x}_i\mathsf{MAJ}\left(\mathbf{x}_1,\cdots,\mathbf{x}_L\right)\right]+\left(\mathbb{E}\left[\mathbf{x}_j\mathsf{MAJ}\left(\mathbf{x}_1,\cdots,\mathbf{x}_L\right)\right]\right.\\
&\left.-\mathbb{E}\left[\mathbf{x}_i\mathsf{MAJ}\left(\mathbf{x}_1,\cdots,\mathbf{x}_L\right)\right]\right)\\
\leqslant&\frac{r-\eta}{L}+\frac{2^{L+1}r}{L}\zeta.
\end{aligned}
$$

Thus we have

$$\mathbb{E}\left[(\mathbf{x}_1+\cdots+\mathbf{x}_L)\,\mathsf{MAJ}\left(\mathbf{x}_1,\cdots,\mathbf{x}_L\right)\right]\leqslant r-\eta+\frac{2^{L+1}r(L-1)}{L}\zeta.$$

That is,

$$\begin{aligned}\mathbb{E}\left[Q_{1,\cdots,L}\right] =& \mathbb{E}\left[\left(\mathbf{x}_1+\cdots+\mathbf{x}_L\right)\mathsf{MAJ}\left(\mathbf{x}_1,\cdots,\mathbf{x}_L\right)-r\right]\\ \leqslant& -\eta+\frac{2^{L+1}r(L-1)}{L}\zeta\\ =& -\frac{\eta}{2},\end{aligned} \tag{118}$$

where the last Eqn. (118) follows by the choice of ζ (Eqn. (111)). Since the above calculations work for any list $\underline{x}_1,\cdots,\underline{x}_L\in\mathcal{C}'$ of *distinct* codewords, we have that for $(i_1,\cdots,i_L)\in\binom{[M']}{L}$, the same bound holds,

$$\mathbb{E}\left[Q_{i_1,\cdots,i_L}\right]\leqslant-\frac{\eta}{2}.$$

For lists $(i_1,\cdots,i_L)\in[M']^L$ that are not all distinct, we use the trivial bound,

$$\begin{aligned}\mathbb{E}\left[Q_{i_1,\cdots,i_L}\right] =& \mathbb{E}\left[\left|\mathbf{x}_{i_1}+\cdots+\mathbf{x}_{i_L}\right|-r\right]\\ \leqslant& L-r.\end{aligned}$$

Overall we have

$$\begin{aligned}Q=& \mathop{\mathbb{E}}_{(\mathbf{i}_1,\cdots,\mathbf{i}_L)\sim[2^{nR}]^L}\mathbb{E}\left[Q_{\mathbf{i}_1,\cdots,\mathbf{i}_L}\right]\\ =&\frac{1}{2^{nRL}}\left(\sum_{i_1,\cdots,i_L\in[2^{nR}]\text{ distinct}}Q_{i_1,\cdots,i_L}+\sum_{i_1,\cdots,i_L\in[2^{nR}]\text{ not distinct}}Q_{i_1,\cdots,i_L}\right)\\ \leqslant&\frac{1}{2^{nRL}}\Bigg[2^{nR}\left(2^{nR}-1\right)\cdots\left(2^{nR}-L+1\right)\left(-\frac{\eta}{2}\right)\\ &+\left(2^{nRL}-2^{nR}\left(2^{nR}-1\right)\cdots\left(2^{nR}-L+1\right)\right)(L-r)\Bigg]\\ <&0.\end{aligned} \tag{119}$$

The last inequality (119) holds if

$$\left|\mathcal{C}'\right|>\max\left\{2(L-1),\frac{2^{L+1}L!(L+r)}{\eta}\right\},$$

by similar calculations to Sec. 13.2.

Lower bound on Q. Following the calculations in the proof of generalized Plotkin bound for list decoding, we have

$$\begin{aligned}&Q+r\\ =&\mathop{\mathbb{E}}_{(\mathbf{i}_1,\cdots,\mathbf{i}_L)\sim[2^{nR}]^L}\mathbb{E}\left[\left|\mathbf{x}_{\mathbf{i}_1}+\cdots+\mathbf{x}_{\mathbf{i}_L}\right|\right]\\ =&\frac{1}{2^{nRL}}\sum_{i_1,\cdots,i_L\in[2^{nR}]}\sum_{x_1,\cdots,x_L\in\{-1,1\}}\tau_{\underline{x}_{i_1},\cdots,\underline{x}_{i_L}}\left(\phi^{-1}(x_1),\cdots,\phi^{-1}(x_L)\right)\left|x_1+\cdots+x_L\right|\\ =&\frac{1}{2^{nRL}}\sum_{\substack{i_1,\cdots,i_L\in\left[2^{nR}\right]\\ x_1,\cdots,x_L\in\{-1,1\}}}\frac{1}{n}\sum_{j=1}^{n}\mathbb{1}_{\left\{\underline{x}_{i_1}(j)=\phi^{-1}(x_1)\right\}}\cdots\mathbb{1}_{\left\{\underline{x}_{i_L}(j)=\phi^{-1}(x_L)\right\}}\left|x_1+\cdots+x_L\right|\end{aligned} \tag{120}$$

$$= \frac{1}{n}\sum_{j=1}^{n}\sum_{x_1,\cdots,x_L\in\{-1,1\}}\prod_{\ell=1}^{L}\left(\frac{1}{2^{nR}}\sum_{i\in[2^{nR}]}\mathbb{1}_{\{\underline{x}_i(j)=\phi^{-1}(x_\ell)\}}\right)|x_1+\cdots+x_L| \tag{121}$$

$$= \frac{1}{n}\sum_{j=1}^{n}\sum_{x_1,\cdots,x_L\in\{-1,1\}}\prod_{\ell=1}^{L}P_{\mathbf{x}}^{(j)}(\phi^{-1}(x_\ell))\,|x_1+\cdots+x_L| \tag{122}$$

$$= \mathop{\mathbb{E}}_{\mathbf{j}\sim[n]}\left[\mathop{\mathbb{E}}_{\left(\mathbf{x}_1^{(\mathbf{j})},\cdots,\mathbf{x}_L^{(\mathbf{j})}\right)\sim\left(P_{\mathbf{x}}^{(\mathbf{j})}\right)^{\otimes L}}\left[\left|\mathbf{x}_1^{(\mathbf{j})}+\cdots+\mathbf{x}_L^{(\mathbf{j})}\right|\right]\right]. \tag{123}$$

In the above calculations, we used the following definitions and facts.

1. Eqn. (120) follows from the definition of joint types.
2. Eqn. (121) is obtained by rearranging terms.
3. In Eqn. (122), as before, we let, for $j\in[n]$, $x\in\mathbb{F}_2$,

$$P_{\mathbf{x}}^{(j)}(x)=\frac{1}{2^{nR}}\sum_{i\in[2^{nR}]}\mathbb{1}_{\{\underline{x}_i(j)=x\}}$$

denote the empirical distribution of the j-th *column* of $\mathcal{C}'$ when viewed as an $M'\times n$ matrix.

In expression (123), the j-th summand can be viewed as the translation distance of a non-lazy one-dimensional random walk after L steps. The walker moves left ($x=1$) with probability $P_{\mathbf{x}}^{(j)}(1)$ and moves right ($x=0$) with probability $P_{\mathbf{x}}^{(j)}(0)$. It is not hard to check that the expected translation distance is minimized when the walker is unbiased, i.e., when $P_{\mathbf{x}}^{(j)}(1)=P_{\mathbf{x}}^{(j)}(0)=1/2$. This is formally justified in Appendix C. Hence, for every $j\in[n]$,

$$\mathop{\mathbb{E}}_{\left(\mathbf{x}_1^{(j)},\cdots,\mathbf{x}_L^{(j)}\right)\sim\left(P_{\mathbf{x}}^{(j)}\right)^{\otimes L}}\left[\left|\mathbf{x}_1^{(j)}+\cdots+\mathbf{x}_L^{(j)}\right|\right]-r\geqslant 0.$$

Since the above bound is valid for every $j\in[n]$, it is still valid averaged over $\mathbf{j}\sim[n]$. Hence we have $Q\geqslant 0$. ◀

17 GV rate vs. cloud rate

In this section, we are concerned with the question of unique decoding (special case where $L-1=1$) under the bit-flip model.

In [43], bounds on achievable rates of codes for general adversarial channels are provided. A Gilbert–Varshamov-type expression was obtained using a purely random code construction, and a rate lower bound (we call *cloud rate*) that generalizes the GV-type expression was given by a cloud code construction. We evaluate both bounds under the bit-flip model. We show that the Gilbert–Varshamov-type bound for general adversarial channels indeed coincide with the classic GV bound in this particular setting. We also provide a convex program for evaluating the cloud rate.

We use the probability vector $\begin{bmatrix}P_{\mathbf{x}}(1) & \cdots & P_{\mathbf{x}}(|\mathcal{X}|)\end{bmatrix}^\top$ to denote a distribution $P_{\mathbf{x}}\in\Delta(\mathcal{X})$. Take any input distribution

$$P_{\mathbf{x}}=\mathrm{Bern}(w)=\begin{bmatrix}1-w\\ w\end{bmatrix},$$

from $\Delta(\{0,1\})$, we first explicitly compute the basic objects we are concerned with in this paper.

$$
\begin{aligned}
\Delta :=& \Delta(\{0,1\}) \\
=& \left\{ P_{\mathbf{x}_1,\mathbf{x}_2} \in \mathbb{R}^{2\times 2} \colon \begin{array}{rl} P_{\mathbf{x}_1,\mathbf{x}_2}(x_1,x_2) & \geqslant 0,\ \forall x_1,x_2 \\ \sum_{x_1,x_2} P_{\mathbf{x}_1,\mathbf{x}_2}(x_1,x_2) & = 1 \end{array} \right\} \\
=& \left\{ \begin{bmatrix} a & c \\ d & b \end{bmatrix} \in \mathbb{R}^{2\times 2} \colon \begin{array}{rl} a,b,c,d & \geqslant 0 \\ a+b+c+d & = 1 \end{array} \right\} \\
=& \left\{ \begin{bmatrix} a & c \\ 1-a-b-c & b \end{bmatrix} \in \mathbb{R}^{2\times 2} \colon \begin{array}{rl} a,b,c & \geqslant 0 \\ a+b+c & \leqslant 1 \end{array} \right\}.
\end{aligned}
$$

$$
\begin{aligned}
\mathcal{J}(w) :=& \mathcal{J}\left(\begin{bmatrix} 1-w \\ w \end{bmatrix} \right) \\
=& \{ P_{\mathbf{x}_1,\mathbf{x}_2} \in \Delta \colon P_{\mathbf{x}_1} = P_{\mathbf{x}_2} = P_{\mathbf{x}} \} \\
=& \left\{ \begin{bmatrix} a & c \\ d & b \end{bmatrix} \in \mathbb{R}^{2\times 2} \colon \begin{array}{l} a,b,c,d \geqslant 0 \\ a+b+c+d = 1 \\ d+b = w \\ c+b = w \end{array} \right\} \\
=& \left\{ \begin{bmatrix} 1-w-d & d \\ d & w-d \end{bmatrix} \in \mathbb{R}^{2\times 2} \colon 0 \leqslant d \leqslant \min\{w, 1-w\} \right\}.
\end{aligned}
$$

$$
\begin{aligned}
\mathcal{K}(w,p) :=& \mathcal{K}\left(\begin{bmatrix} 1-w \\ w \end{bmatrix} \right) \\
=& \{ P_{\mathbf{x}_1,\mathbf{x}_2} \in \mathcal{J}(w) : P_{\mathbf{x}_1,\mathbf{x}_2}(0,1) + P_{\mathbf{x}_1,\mathbf{x}_2}(1,0) \leqslant 2p \} \\
=& \left\{ \begin{bmatrix} 1-w-d & d \\ d & w-d \end{bmatrix} \in \mathbb{R}^{2\times 2} \colon 0 \leqslant d \leqslant \min\{w, 1-w, p\} \right\}.
\end{aligned}
$$

Since $\mathsf{CP}_2 = \mathsf{DNN}_2$, we have

$$
\begin{aligned}
&\mathsf{CP}_2(w) \\
=& \mathsf{CP}_2 \cap \mathcal{J}(w) \\
=& \left\{ \begin{bmatrix} w-d & d \\ d & 1-w-d \end{bmatrix} : 0 \leqslant d \leqslant \min\{w, 1-w\},\ (w-d)(1-w-d) - d^2 \geqslant 0 \right\} \\
=& \left\{ \begin{bmatrix} w-d & d \\ d & 1-w-d \end{bmatrix} : 0 \leqslant d \leqslant w - w^2 \right\}.
\end{aligned}
$$

Note that to ensure $\mathsf{CP}_2(w) \backslash \mathcal{K}(w,p) \neq \varnothing$, we need

$$
0 < p < 1/4,\ w \in \left(\tfrac{1-\sqrt{1-4p}}{2}, \tfrac{1+\sqrt{1-4p}}{2} \right).
$$

In other words, $0 < w < 1$ and $0 < p < w - w^2$. In this case,

$$
\mathcal{K}(w,p) = \left\{ \begin{bmatrix} 1-w-d & d \\ d & w-d \end{bmatrix} \in \mathbb{R}^{2\times 2} \colon 0 \leqslant d \leqslant p \right\}.
$$

Actually, if the above conditions hold, then when $1/3 \leqslant w < 1$, the boundary of $\mathcal{K}(w,p)$ is p and the boundary of $\mathsf{CP}_2(w)$ is $w - w^2$. Note that the right boundary $\begin{bmatrix} (1-w)^2 & w-w^2 \\ w-w^2 & w^2 \end{bmatrix} = \begin{bmatrix} 1-w \\ w \end{bmatrix}^{\otimes 2}$ of $\mathsf{CP}_2(w)$ is the *only* distribution in $\mathsf{CP}_2(w)$ of CP-rank-1.

GV rate. We first state the GV-type expression given by in [43].

▶ **Lemma 81** (Gilbert–Varshamov rate). *For a general adversarial channel given by* $\mathcal{A} = \{\mathcal{X}, \lambda_{\mathbf{x}}, \mathcal{S}, \lambda_{\mathbf{s}}, \mathcal{Y}, W_{\mathbf{y}|\mathbf{x},\mathbf{s}}\}$*, its unique decoding capacity is at least*

$$R_{\mathrm{GV}} = \max_{P_{\mathbf{x}} \in \lambda_{\mathbf{x}}} \min_{P_{\mathbf{x}_1,\mathbf{x}_2} \in \mathcal{K}(P_{\mathbf{x}})} I(\mathbf{x};\mathbf{x}'),$$

where the mutual information is calculated using $P_{\mathbf{x}_1,\mathbf{x}_2}$*.*

We now evaluate the above expression under the bit-flip model.

$$\begin{aligned}
R_{\mathrm{GV}} &= \max_{P_{\mathbf{x}} \in \lambda_{\mathbf{x}}} \min_{P_{\mathbf{x}_1,\mathbf{x}_2} \in \mathcal{K}(P_{\mathbf{x}})} I(\mathbf{x};\mathbf{x}') \\
&= \max_{\begin{bmatrix} 1-w \\ w \end{bmatrix} \in \Delta} \min_{\begin{bmatrix} 1-w-d & d \\ d & w-d \end{bmatrix} \in \mathcal{K}(w,p)} D\left(\begin{bmatrix} 1-w-d & d \\ d & w-d \end{bmatrix} \middle\| \begin{bmatrix} 1-w \\ w \end{bmatrix}^{\otimes 2} \right) \\
&= \max_{0<w<1} \min_{0 \leqslant d \leqslant p} (w-d) \log \frac{w-d}{w^2} + 2d \log \frac{d}{w(1-w)} + (1-w-d) \log \frac{1-w-d}{(1-w)^2} \\
&= \max_{0<w<1} (w-p) \log \frac{w-p}{w^2} + 2p \log \frac{p}{w(1-w)} + (1-w-p) \log \frac{1-w-p}{(1-w)^2} \\
&= (1/2-p) \log \frac{1/2-p}{(1/2)^2} + 2p \log \frac{p}{(1/2)(1-1/2)} + (1-1/2-p) \log \frac{1-1/2-p}{(1-1/2)^2} \\
&= 1 - H(2p).
\end{aligned}$$

This matches the classic GV bound given a greedy volume packing argument.

Cloud rate. We now state the cloud rate expression given by [43].

▶ **Lemma 82** (Cloud rate).

For a general adversarial channel $\mathcal{A} = \{\mathcal{X}, \lambda_{\mathbf{x}}, \mathcal{S}, \lambda_{\mathbf{s}}, \mathcal{Y}, W_{\mathbf{y}|\mathbf{x},\mathbf{s}}\}$, its unique decoding capacity is at least

$$R_{\mathrm{cloud}} = \max_{P_{\mathbf{x}} \in \lambda_{\mathbf{x}}} \max_{P_{\mathbf{x}_1,\mathbf{x}_2} \in \mathsf{CP}_2(P_{\mathbf{x}}) \setminus \mathcal{K}(P_{\mathbf{x}})} \max_{\substack{P_{\mathbf{u}}, P_{\mathbf{x}|\mathbf{u}}: \\ \left[P_{\mathbf{u}} P_{\mathbf{x}|\mathbf{u}}^{\otimes 2}\right]_{\mathbf{x}_1,\mathbf{x}_2} = P_{\mathbf{x}_1,\mathbf{x}_2}}} \min_{P_{\mathbf{u},\mathbf{x}_1,\mathbf{x}_1} \in \mathcal{K}_{\mathrm{cloud}}(P_{\mathbf{u},\mathbf{x}})} D\left(P_{\mathbf{u},\mathbf{x}_1,\mathbf{x}_2} \middle\| P_{\mathbf{u}} P_{\mathbf{x}|\mathbf{u}}^{\otimes 2}\right),$$

where

$$\mathcal{K}_{\mathrm{cloud}}(P_{\mathbf{u},\mathbf{x}}) := \left\{ [P_{\mathbf{u},\mathbf{x}_1,\mathbf{x}_2,\mathbf{s}_1,\mathbf{s}_2,\mathbf{y}}]_{\mathbf{u},\mathbf{x}_1,\mathbf{x}_2} \in \Delta\left(\mathcal{U} \times \mathcal{X}^2\right) : \begin{array}{rl} P_{\mathbf{u},\mathbf{x}_1,\mathbf{x}_2,\mathbf{s}_1,\mathbf{s}_2,\mathbf{y}} \in & \Delta\left(\mathcal{U} \times \mathcal{X}^2 \times \mathcal{S}^2 \times \mathcal{Y}\right) \\ P_{\mathbf{s}_1}, P_{\mathbf{s}_2} \in & \lambda_{\mathbf{s}} \\ P_{\mathbf{u},\mathbf{x}_1,\mathbf{s}_1,\mathbf{y}} = & P_{\mathbf{u},\mathbf{x}} P_{\mathbf{s}_1|\mathbf{u},\mathbf{x}_1} W_{\mathbf{y}|\mathbf{x}_1,\mathbf{s}_1} \\ P_{\mathbf{u},\mathbf{x}_2,\mathbf{s}_2,\mathbf{y}} = & P_{\mathbf{u},\mathbf{x}} P_{\mathbf{s}_2|\mathbf{u},\mathbf{x}_2} W_{\mathbf{y}|\mathbf{x}_2,\mathbf{s}_2} \end{array} \right\}.$$

▶ Remark 83. The reason that [43] has to define a different confusability set $\mathcal{K}_{\mathrm{cloud}}$ when cloud code is using is that as a part of the code design, the distributions $P_{\mathbf{u}}, P_{\mathbf{u}|\mathbf{x}}$ are revealed to every party, including the adversary, hence he may be able to inject noise patterns that are potentially more malicious compared with the case where he does not have such knowledge. We refer the readers to the proof in [43].

In the bit-flip setting, it is easy to verify that

$$\begin{aligned}\mathcal{K}_{\text{cloud}}(P_{\mathbf{u},\mathbf{x}}) &= \left\{ P_{\mathbf{u},\mathbf{x}_1,\mathbf{x}_2} \in \Delta\left(\mathcal{U}\times\mathcal{X}^2\right) : \begin{array}{rl} P_{\mathbf{u},\mathbf{x}_1} = P_{\mathbf{u},\mathbf{x}_2} = & P_{\mathbf{u},\mathbf{x}} \\ P_{\mathbf{x}_1,\mathbf{x}_2}(0,1) + P_{\mathbf{x}_1,\mathbf{x}_2}(1,0) \leqslant & 2p \end{array} \right\} \\ &= \left\{ p \in \mathbb{R}^{|\mathcal{U}|\times 2\times 2} : \begin{array}{rl} p_{u,x_1,x_2} \geqslant & 0,\ \forall u,x_1,x_2 \\ \sum_{u,x_1,x_2} p_{u,x_1,x_2} = & 1 \\ \sum_{x_2} p_{u,x_1,x_2} = & p_{u,x_1},\ \forall u,x_1 \\ \sum_{x_1} p_{u,x_1,x_2} = & p_{u,x_2},\ \forall u,x_2 \\ \sum_u p_{u,0,1} + p_{u,1,0} \leqslant & 2p \end{array} \right\}.\end{aligned}$$

We use the notation $p_{u,x_1,x_2} \coloneqq P_{\mathbf{u},\mathbf{x}_1,\mathbf{x}_2}(u,x_1,x_2)$ and $p_{u,x} \coloneqq P_{\mathbf{u},\mathbf{x}}(u,x)$ for all $u \in \mathcal{U}, x_1, x_2 \in \{0,1\}$. The third maximization is over all extensions which correspond to CP decompositions of $P_{\mathbf{x}_1,\mathbf{x}_2}$. Note that for a CP matrix, its CP decomposition is not necessarily unique, even if we require the decomposition to meet the CP-rank [23]. A CP decomposition of a CP distribution can contain an arbitrarily large number of terms. Here we focus on decompositions which *meet* the CP-rank of $P_{\mathbf{x}_1,\mathbf{x}_2}$. That is, $|\mathcal{U}| = \text{CP-rank}(P_{\mathbf{x}_1,\mathbf{x}_2})$.

Note that the objective function KL-divergence also equals

$$D\left(P_{\mathbf{u},\mathbf{x}_1,\mathbf{x}_2} \middle\| P_{\mathbf{u}} P_{\mathbf{x}|\mathbf{u}}^{\otimes 2}\right) = I\left(\mathbf{x}_1;\mathbf{x}_2|\mathbf{u}\right),$$

where the mutual information is w.r.t. $P_{\mathbf{u},\mathbf{x}_1,\mathbf{x}_2}$.

Note that even if we could show $R_{\text{cloud}} \leqslant R_{\text{GV}}$, this does *not* mean cloud codes will never attain a rate larger than the GV bound. It only means that the cloud rate expression we have cannot take values larger than the GV bound. This is because our bounds are only achievable, but we do not have matching upper bounds. Indeed, this is an extremely difficult question even under simple models.

Actually *all* CP decompositions meeting the CP-rank of a CP distribution can be computed. For a CP-rank-2 distribution $\begin{bmatrix} 1-w-b & b \\ b & w-b \end{bmatrix} \in \mathsf{CP}_2(w) \backslash \mathcal{K}(w,p)$ where $b \neq w - w^2$, we write its CP decomposition as

$$\begin{aligned}\begin{bmatrix} 1-w-b & b \\ b & w-b \end{bmatrix} =& \alpha \begin{bmatrix} 1-u \\ u \end{bmatrix}^{\otimes 2} + \beta \begin{bmatrix} 1-v \\ v \end{bmatrix}^{\otimes 2} \\ =& \begin{bmatrix} \alpha(1-u)^2 + \beta(1-v)^2 & \alpha u(1-u) + \beta v(1-v) \\ \alpha u(1-u) + \beta v(1-v) & \alpha u^2 + \beta v^2 \end{bmatrix}.\end{aligned}$$

Solving the equation in terms of b and u, we have

$$\begin{aligned}\alpha \coloneqq& \alpha(w,b,u) = \frac{w-b-w^2}{u^2+w-2uw-b}, \\ \beta \coloneqq& \beta(w,b,u) = 1-\alpha = \frac{(u-w)^2}{u^2+w-2uw-b}, \\ v \coloneqq& v(w,b,u) = \frac{b-w+uw}{w-u},\end{aligned}$$

where $u \in \left[0, \frac{b}{1-w}\right] \cup \left[\frac{w-b}{w}, 1\right]$.

Any such decomposition gives rise to a joint distribution $P_{\mathbf{u}} P_{\mathbf{x}|\mathbf{u}}^{\otimes 2}$ which is a $2\times 2\times 2$ tensor.

$$P_{\mathbf{u}=0} P_{\mathbf{x}|\mathbf{u}=0}^{\otimes 2} = \begin{bmatrix} \alpha(1-u)^2 & \alpha u(1-u) \\ \alpha u(1-u) & \alpha(1-u)^2 \end{bmatrix}, \quad P_{\mathbf{u}=1} P_{\mathbf{x}|\mathbf{u}=1}^{\otimes 2} = \begin{bmatrix} \beta v^2 & \beta v(1-v) \\ \beta v(1-v) & \beta(1-v)^2 \end{bmatrix}.$$

It also induces a distribution $P_{\mathbf{u},\mathbf{x}}$.

$$P_{\mathbf{u},\mathbf{x}} = \begin{bmatrix} \alpha(1-u) & \alpha u \\ \beta(1-v) & \beta v \end{bmatrix}.$$

Now for any CP decomposition $P_{\mathbf{u},\mathbf{x}_1,\mathbf{x}_2}$ of a CP distribution $P_{\mathbf{x}_1,\mathbf{x}_2} = \begin{bmatrix} w-b & b \\ b & 1-w-b \end{bmatrix}$, the inner minimization can be written as minimizing a convex function over a polytope.

$$\begin{array}{rl} \min_p & D(p\|P_{\mathbf{u}}P_{\mathbf{x}|\mathbf{u}}^{\otimes 2}) \\ \text{subject to} & p \in \mathcal{K}_{\text{cloud}}(P_{\mathbf{u},\mathbf{x}}) \end{array}.$$

It can be expanded in the following explicit form.

$$\begin{array}{rl} \min_p & p_{0,0,0}\log\frac{p_{0,0,0}}{\alpha(1-u)^2} + p_{0,0,1}\log\frac{p_{0,0,1}}{\alpha u(1-u)} + p_{0,1,0}\log\frac{p_{0,1,0}}{\alpha u(1-u)} + p_{0,1,1}\log\frac{p_{0,1,1}}{\alpha u^2} \\ & + p_{1,0,0}\log\frac{p_{1,0,0}}{\beta(1-v)^2} + p_{1,0,1}\log\frac{p_{1,0,1}}{\beta v(1-v)} + p_{1,1,0}\log\frac{p_{1,0,0}}{\beta v(1-v)} + p_{1,1,1}\log\frac{p_{1,1,1}}{\beta v^2} \\ \text{subject to} & \left.\begin{array}{l} p_{i,j,k} \geqslant 0,\ \forall i,j,k \\ \sum_{i,j,k} p_{i,j,k} = 1 \end{array}\right\} p \in \Delta\left(\{0,1\}^3\right) \\ & \left.\begin{array}{l} p_{0,0,0} + p_{0,0,1} = \alpha(1-u) \\ p_{0,1,0} + p_{0,1,1} = \alpha u \\ p_{1,0,0} + p_{1,0,1} = \beta(1-v) \\ p_{1,1,0} + p_{1,1,1} = \beta v \end{array}\right\} [P_{\mathbf{u},\mathbf{x}_1,\mathbf{x}_2}]_{\mathbf{u},\mathbf{x}_1} = P_{\mathbf{u},\mathbf{x}} \\ & \left.\begin{array}{l} p_{0,0,0} + p_{0,1,0} = \alpha(1-u) \\ p_{0,0,1} + p_{0,1,1} = \alpha u \\ p_{1,0,0} + p_{1,1,0} = \beta(1-v) \\ p_{1,0,1} + p_{1,1,1} = \beta v \end{array}\right\} [P_{\mathbf{u},\mathbf{x}_1,\mathbf{x}_2}]_{\mathbf{u},\mathbf{x}_2} = P_{\mathbf{u},\mathbf{x}} \\ & p_{0,0,1} + p_{0,1,0} + p_{1,0,1} + p_{1,1,0} \leqslant 2p. \end{array}$$

Note that it is implied by the given constraints that $p_{u,x_1,x_2} = p_{u,x_2,x_1}$. Also, the p.m.f. constraint $\sum_{u,x_1,x_2} p_{u,x_1,x_2} = 1$ is actually redundant. Hence the problem can be simplified as follows.

$$\begin{array}{rl} \min_{\underline{p}} & p_{0,0,0}\log\frac{p_{0,0,0}}{\alpha(1-u)^2} + 2p_{0,0,1}\log\frac{p_{0,0,1}}{\alpha u(1-u)} + p_{0,1,1}\log\frac{p_{0,1,1}}{\alpha u^2} \\ & + p_{1,0,0}\log\frac{p_{1,0,0}}{\beta(1-v)^2} + 2p_{1,0,1}\log\frac{p_{1,0,1}}{\beta v(1-v)} + p_{1,1,1}\log\frac{p_{1,1,1}}{\beta v^2} \\ \text{subject to} & -p_{i,j,k} \leqslant 0,\ \forall i,j,k \\ & p_{0,0,0} + p_{0,0,1} = \alpha u \\ & p_{0,0,1} + p_{0,1,1} = \alpha(1-u) \\ & p_{1,0,0} + p_{1,0,1} = \beta v \\ & p_{1,0,1} + p_{1,1,1} = \beta(1-v) \\ & p_{0,0,1} + p_{1,0,1} \leqslant p. \end{array}$$

Let $D^*(w,b,u)$ denote the optimal value of the above minimization. The final cloud rate is given by

$$\max_{0<w<1} \max_{p<b\leqslant w-w^2} \max_{u\in\left[0,\frac{b}{1-w}\right]\cup\left[\frac{w-b}{w},1\right]} D^*(w,b,u),$$

where the first maximization corresponds to finding the optimal input distribution $\begin{bmatrix} 1-w \\ w \end{bmatrix}$, second maximization corresponds to finding the optimal CP distribution $\begin{bmatrix} 1-w-b & b \\ b & w-b \end{bmatrix}$ outside $\mathcal{K}(w)$, and the third optimization corresponds to finding the optimal CP-decomposition $\alpha\begin{bmatrix} 1-u \\ u \end{bmatrix}^{\otimes 2} + \beta\begin{bmatrix} 1-v \\ v \end{bmatrix}^{\otimes 2}$ of the optimal CP distribution.

18 Concluding remarks and open problems

In this paper, we study the list decoding problem on general adversarial channels for both large and small list sizes. Given any channel, for large (yet constant) list sizes, we prove the list decoding theorem which identifies the fundamental limit of list decoding. For small (yet arbitrary universal constant) list sizes, we characterize when positive rate list decodable codes are possible.

Many open questions are left after this work is done. We list some of them for future study.

1. In this paper, we made no attempt towards understanding channels with arbitrary transition distributions $W_{\mathbf{y}|\mathbf{x},\mathbf{s}}$ (instead of only those corresponding to deterministic bivariate functions). Pushing our results to such a general setting remains an intriguing open question.
2. Other adversarial channels under further assumptions, e.g., online (causal) channels, channels with feedback, channels with bounded memory, etc., are less understood. There are results regarding each of these topics under very restricted models, e.g., bit-flips [13, 6], deletions [12], etc..
3. We do not have any nontrivial *upper* bound on $(L-1)$-list decoding capacity for general adversarial channels. Existing upper bounds for error correction codes seem tricky to generalize. A reasonable starting point might be to extend the classic Elias–Bassalygo bound [4] whose proof has a similar spirit as the Plotkin bound.
4. Given any adversarial channel, when we are "below the Plotkin point" (i.e., there are non-confusable CP distributions), can we construct *explicit* codes of positive rate? We know that random codes are list decodable w.h.p..

References

1 Rudolf Ahlswede. Channels with arbitrarily varying channel probability functions in the presence of noiseless feedback. *Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete*, 25(3):239–252, 1973.

2 Noga Alon, Boris Bukh, and Yury Polyanskiy. List-decodable zero-rate codes. *IEEE Transactions on Information Theory*, 65(3):1657–1667, 2018.

3 Alexei Ashikhmin, Alexander Barg, and Simon Litsyn. A new upper bound on codes decodable into size-2 lists. In *Numbers, Information and Complexity*, pages 239–244. Springer, 2000.

4 L. A. Bassalygo. New upper boundes for error-correcting codes. *Problems of Information Transmisson*, 1:32–35, 1965.

5 Avraham Ben-Aroya, Dean Doron, and Amnon Ta-Shma. Near-Optimal Erasure List-Decodable Codes, 2018. URL: `https://eccc.weizmann.ac.il/report/2018/065/`.

6 Elwyn R Berlekamp. *Block coding with noiseless feedback*. PhD thesis, Massachusetts Institute of Technology, 1964.

7 Abhishek Bhowmick and Shachar Lovett. List decoding Reed-Muller codes over small fields. *arXiv preprint*, 2014. `arXiv:1407.3433`.

8 David Blackwell, Leo Breiman, and A. J. Thomasian. The Capacity of a Class of Channels. *Ann. of Mathematical Statistics*, 30(4):1229–1241, 1959.

9 Vladimir M Blinovsky. Bounds for codes in the case of list decoding of finite volume. *Problems of Information Transmission*, 22:7–19, 1986.

10 Vladimir M Blinovsky. Code bounds for multiple packings over a nonbinary finite alphabet. *Problems of Information Transmission*, 41:23–32, 2005.

11 Vladimir M Blinovsky. On the convexity of one coding-theory function. *Problems of Information Transmission*, 44:34–39, 2008.

12 Boris Bukh, Venkatesan Guruswami, and Johan Håstad. An improved bound on the fraction of correctable deletions. *IEEE Transactions on Information Theory*, 63(1):93–103, 2016.

13 Z. Chen, S. Jaggi, and M.Langberg. A Characterization of the Capacity of Online (causal) Binary Channels. In *Proc. ACM Symp. on Discrete Algorithms (SODA)*, Portland, U.S.A, June 2015.

14 Sean Clark. How to show $\sum_{i=1}^{\lceil L/2 \rceil} \frac{\binom{2i-2}{i-1}}{i} 2^{-2i} = 1/2 - 2^{-L-1}\binom{L}{(L-1)/2}$? Mathematics Stack Exchange, February 2019. URL: `https://math.stackexchange.com/q/3101258` (version: 2019-02-05).

15 G. D. Cohen, S. N. Litsyn, and G. Zemor. Upper bounds on generalized distances. *IEEE transactions on Information Theory*, 40:2090–2092, November 1994.

16 Imre Csiszár and János Körner. *Information theory: coding theorems for discrete memoryless systems*. Cambridge University Press, 2011.

17 Imre Csiszár and Prakash Narayan. The Capacity of the Arbitrarily Varying Channel Revisited : Positivity, Constraints. *IEEE transactions on Information Theory*, 34:181–193, 1988.

18 Philippe Delsarte. An algebraic approach to the association schemes of coding theory. Technical report, Philips Research Laboratories, 1973.

19 Ilias Diakonikolas, Daniel M Kane, and Alistair Stewart. List-decodable robust mean estimation and learning mixtures of spherical gaussians. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1047–1060. ACM, 2018.

20 Dean Doron, Dana Moshkovitz, Justin Oh, and David Zuckerman. Nearly Optimal Pseudorandomness From Hardness. *ECCC preprint TR19-099*, 2019.

21 Peter Elias. List decoding for noisy channels. Technical report, Research Laboratory of Electronics, Massachusetts Institute of Technology, 1957.

22 Edgar N Gilbert. A comparison of signalling alphabets. *The Bell system technical journal*, 31(3):504–522, 1952.

23 Patrick Groetzner and Mirjam Dür. A factorization method for completely positive matrices. *preprint*, 2018.

24 V. Guruswami. *List Decoding of Error Correcting Codes (Lecture Notes in Computer Science),*. Springer-Verlag, NY, 2004.

25 Venkatesan Guruswami. List decoding in average-case complexity and pseudorandomness. In *2006 IEEE Information Theory Workshop-ITW'06 Punta del Este*, pages 32–36. IEEE, 2006.

26 Venkatesan Guruswami and Srivatsan Narayanan. Combinatorial limitations of average-radius list decoding. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 591–606. Springer, 2013.

27 Christopher J Hillar and Lek-Heng Lim. Most tensor problems are NP-hard. *Journal of the ACM (JACM)*, 60(6):45, 2013.

28 Sushrut Karmalkar, Pravesh Kothari, and Adam Klivans. List-Decodable Linear Regression. *arXiv preprint*, 2019. `arXiv:1905.05679`.

29 A. Lapidoth and P. Narayan. Reliable Communication under Channel Uncertainty. *IEEE transactions on Information Theory*, 44:2148–2177, 1998.

30 Jessie MacWilliams. A theorem on the distribution of weights in a systematic code. *Bell System Technical Journal*, 42(1):79–94, 1963.

31 R. J. McEliece, E. R. Rodemich, H. Jr. Rumsey, and L. R. Welch. New upper bounds on the rate of a code via the Delsarte-MacWilliams inequalities. *IEEE transactions on Information Theory*, 23, 1977.

32 Michael Navon and Alex Samorodnitsky. Linear programming bounds for codes via a covering argument. *Discrete & Computational Geometry*, 41(2):199, 2009.

33 Morris Plotkin. Binary codes with specified minimum distance. *IRE Transactions on Information Theory*, 6(4):445–450, 1960.

34 Yury Polyanskiy. Upper bound on list-decoding radius of binary codes. *IEEE Transactions on Information Theory*, 62(3):1119–1128, 2016.

35 Prasad Raghavendra and Morris Yau. List Decodable Learning via Sum of Squares. *arXiv preprint*, 2019. arXiv:1905.04660.

36 Atri Rudra and Mary Wootters. Every list-decodable code for high noise has abundant near-optimal rate puncturings. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, 2014.

37 Atri Rudra and Mary Wootters. It'll probably work out: improved list-decoding through random operations. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science*, 2015.

38 Atri Rudra and Mary Wootters. Average-radius list-recoverability of random linear codes. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, 2018.

39 Anand Sarwate. *Robust and adaptive communication under uncertain interference*. PhD thesis, University of California, Berkeley, 2008.

40 Michael Sipser and Daniel A Spielman. Expander codes. *IEEE transactions on Information Theory*, 42(6):1710–1722, 1996.

41 Amnon Ta-Shma. Explicit, almost optimal, epsilon-balanced codes. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, 2017.

42 RR Varshamov. Estimate of the number of signals in error correcting codes. *Docklady Akad. Nauk, SSSR*, 117:739–741, 1957.

43 Xishi (Nicholas) Wang, Amitalok J. Budkuley, Andrej Bogdanov, and Sidharth Jaggi. When are large codes possible for AVCs? In *IEEE International Symposium on Information Theory (ISIT), Paris*, pages 632–636. IEEE, 2019.

44 L. R. Welch, R. J. McEliece, and H. Jr. Rumsey. A low-rate improvement on the Elias bound. *IEEE transactions on Information Theory*, 23, 1974.

45 Mary Wootters. On the list decodability of random linear codes with large error rates. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, 2013.

46 John M Wozencraft. List decoding. *Quarterly Progress Report*, 48:90–95, 1958.

47 Victor Vasilievich Zyablov and Mark Semenovich Pinsker. List concatenated decoding. *Problemy Peredachi Informatsii*, 17(4):29–33, 1981.

A CP tensors and coP tensors

A.1 Tensor products

▶ **Definition 84** (Tensor product). *For two tensors* $A \in \mathsf{Ten}_n^{\otimes m}, B \in \mathsf{Ten}_n^{\otimes \ell}$, *Their* tensor product *is defined as*

$$A \otimes B := [A\,(i_1, \cdots, i_m)\,B\,(j_1, \cdots, j_\ell)] \in \mathsf{Ten}_n^{\otimes (m+\ell)}.$$

▶ **Definition 85** (Frobenius inner product, Frobenius norm). *For two tensors* $A, B \in \mathsf{Ten}_n^{\otimes m}$, *Their* inner product *is defined as*

$$\langle A, B\rangle := \sum_{i_1,\cdots,i_m \in [n]} A(i_1, \cdots, i_m) B(i_1, \cdots, i_m).$$

The Frobenius norm *is defined as* $\|A\|_F := \sqrt{\langle A, A\rangle}$.

▶ **Definition 86** (Hadamard product). *For two tensors* $A, B \in \mathsf{Ten}_n^{\otimes m}$, *Their* Hadamard product *is defined as*

$$A \circ B := [A(i_1, \cdots, i_m) B(i_1, \cdots, i_m)] \in \mathsf{Ten}_n^{\otimes m}.$$

A.2 Tensor decomposition

▶ **Definition 87** (Canonical decomposition). *For a tensor* $A \in \mathsf{Ten}_n^{\otimes m}$*, its* canonical decomposition *has form*

$$A = \sum_{j=1}^{r} \alpha_j \bigotimes_{i=1}^{m} \underline{x}_{j,i},$$

where each $\underline{x}_{j,i} \in \mathbb{S}_2^{n-1}$*. The smallest* r *for* A *to admit such a decomposition is called the* rank *of* A*. If* A *is symmetric, then*

$$A = \sum_{j=1}^{r} \alpha_j \underline{x}_j^{\otimes m}$$

is an analog of the eigendecomposition of symmetric matrices. The smallest r *is called the* symmetric rank *of* A*.*

▶ **Conjecture 88.** *For* $A \in \mathsf{Sym}_n^{\otimes m}$*,* $\mathrm{rank}(A) = \text{sym-rank}(A)$*.*

▶ Remark 89. It is known to be true if $\mathrm{rank}(A) \leqslant m$.

▶ **Definition 90** (Tucker decomposition). *For a tensor* $A \in \mathsf{Ten}_n^{\otimes m}$*, the* Tucker decomposition *has form*

$$A = \sum_{j_1=1}^{r_1} \cdots \sum_{j_m=1}^{r_m} \alpha_{j_1,\cdots,j_m} \bigotimes_{i=1}^{m} \underline{x}_{j_i,j}.$$

It is an analogy of the singular value decomposition of matrices.

A tensor $A \in \mathsf{Ten}_n^{\otimes m}$ has $n(m-1)^{n-1}$ eigenvalues. A may have non-real eigenvalues even if A is symmetric. If an eigenvector is real, then the corresponding eigenvalue is also real. Such eigenvalues are called *H-eigenvalues*. They always exist for even-order tensors.

A.3 Special tensors

▶ **Definition 91** (NN tensors). *A tensor is said to be* non-negative *if each of its entry is non-negative. The set of order-m dimension-n non-negative tensors is denoted by* $\mathsf{NN}_n^{\otimes m}$

▶ **Definition 92** (PSD tensors, PD Tensors). *For even* m*,* $A \in \mathsf{Ten}_n^{\otimes m}$ *is* positive semidefinite (PSD) *if* $\langle A, \underline{x}^{\otimes m} \rangle \geqslant 0$ *for any* $\underline{x} \in \mathbb{R}^n$*.* A *is* positive definite (PD) *if the above inequality is strict for all* $\underline{x} \neq 0$*.*

The sets of PSD *and* PD *tensors is denoted by* $\mathsf{PSD}_n^{\otimes m}$ *and* $\mathsf{PD}_n^{\otimes m}$*, respectively.*

▶ **Definition 93** (CP tensors, CP tensor rank). *A tensor* $P \in \mathsf{Ten}_n^{\otimes m}$ *is said to be* completely positive *if for some* $r \geqslant 1$*, there are component-wise non-negative vectors* $\underline{p}_1, \cdots, \underline{p}_r \in \mathbb{R}^n_{\geqslant 0}$ *such that*

$$P = \sum_{j=1}^{r} \underline{p}_j^{\otimes m}.$$

The set of CP *tensors is denoted by* $\mathsf{CP}_n^{\otimes m}$*. The least* r *such that* P *has a completely positive decomposition is called the* CP-rank *of* P*. If* $\mathrm{span}\{P_1, \cdots, P_r\} = \mathbb{R}^n$ *then* P *is said to be* strongly CP*.*

▶ **Fact 94.** *Verifying if a symmetric non-negative tensor is* CP *is* NP*-hard.*

▶ **Definition 95** (coP tensors). *$A \in \mathsf{Sym}_n^{\otimes m}$ is* copositive *if $\langle A, \underline{x}^{\otimes}\rangle \geqslant 0$ for all $\underline{x} \in \mathbb{R}^n_{\geqslant 0}$. The set of copositive tensors is denoted by $\mathsf{coP}_n^{\otimes m}$.*

▶ **Theorem 96** (Duality). *$\mathsf{CP}_n^{\otimes m}$ and $\mathsf{coP}_n^{\otimes m}$ are closed convex pointed cones with nonempty interior in $\mathsf{Sym}_n^{\otimes m}$. For $m \geqslant 2$, $n \geqslant 1$, they are dual to each other.*

▶ **Definition 97** (DNN tensors). *For even m, $A \in \mathsf{Sym}_n^{\otimes m}$ is* doubly non-negative (DNN) *if A is entry-wise non-negative and $\langle A, \underline{x}^{\otimes m}\rangle$ is a sum-of-square as a polynomial in the components of $\underline{x}$.*

▶ **Fact 98.** *The double non-negativity of a tensor can be verified in polynomial time using SDP.*

▶ **Fact 99.** *The following inclusion relations between different sets of special tensors hold.*
1. $\mathsf{PSD}_n^{\otimes m} \subseteq \mathsf{coP}_n^{\otimes m}$.
2. $\mathsf{CP}_n^{\otimes m} \subseteq \mathsf{DNN}_n^{\otimes m} \subseteq \mathsf{NN}_n^{\otimes m} \subseteq \mathsf{coP}_n^{\otimes m} \subseteq \mathsf{Sym}_n^{\otimes m}$.

B Hypergraph Ramsey numbers

Let $R_k^{(r)}(s_1, \cdots, s_k)$ denote the smallest size of an r-uniform hypergraph such that for any k-colouring, there must be a monochromatic clique of size s_i for some $i \in [k]$.

Define tower function $t_1(x) = x$ and $t_{i+1}(x) = 2^{t_i(x)}$.

▶ **Lemma 100** (Properties of hypergraph Ramsey numbers). 1. *For any $i \in [k]$, and $s_j \geqslant r$ $(j \neq i)$,*

$$R_k^{(r)}(s_1, \cdots, s_{i-1}, r, s_{i+1}, \cdots, s_k) = R_{k-1}^{(r)}(s_1, \cdots, s_{i-1}, s_{i+1}, \cdots, s_k).$$

2. *For any $\sigma \in S_k$,*

$$R_k^{(r)}(s_1, \cdots, s_k) = R_k^{(r)}(s_{\sigma(1)}, \cdots, s_{\sigma(k)}).$$

▶ **Lemma 101** (Finiteness of hypergraph Ramsey numbers). *For any positive integers r, k, s_1, $\cdots$, s_k, the hypergraph Ramsey number $R_k^{(r)}(s_1, \cdots, s_k)$ is finite. In particular, it satisfies the following recursive inequalities.*

$$R_k^{(r)}(s_1, \cdots, s_k) \leqslant 1 + R_k^{(r-1)}\Big(R_k^{(r)}(s_1 - 1, s_2, \cdots, s_k), R_k^{(r)}(s_1, s_2 - 1, \cdots, s_k), \cdots, \cdots, R_k^{(r)}(s_1, s_2, \cdots, s_k - 1)\Big),$$

$$R_k^{(r)}(s_1, \cdots, s_k) \leqslant 1 + \sum_{i=1}^{k} R_k^{(r-1)}\Big(R_k^{(r)}(s_1, \cdots, s_{i-1}, s_i - 1, s_{i+1}, \cdots, s_k), \cdots, \cdots, R_k^{(r)}(s_1, \cdots, s_{i-1}, s_i - 1, s_{i+1}, \cdots, s_k)\Big),$$

$$R_k^{(r)}(s_1, \cdots, s_k) \leqslant R_{k-1}^{(r)}\left(s_1, \cdots, s_{k-2}, R_2^{(r)}(s_{k-1}, s_k)\right),$$

▶ **Lemma 102** (Bounds on hypergraph Ramsey numbers)**.**

1. *For any* s, t*,*

$$R_2^{(r)}(s,t) \leqslant 2^{\binom{R_2^{(r-1)}(s-1,t-1)}{r-1}}.$$

2. *For* $r \geqslant 3$*, there are constants* $c, c' > 0$ *such that*

$$t_{r-1}(c \cdot s^2) \leqslant R_2^{(r)}(s,s) \leqslant t_r(c' \cdot s).$$

3. *For* $s > k \geqslant 2$*, there are constants* $c, c' > 0$ *such that*

$$t_r(c \cdot k) < R_k^{(r)}(s, \cdots, s) < t_r(c' \cdot k \log k).$$

C Expected translation distance of a one-dimensional random walk

▶ **Lemma 103.** *Consider a random walk* $\mathbf{x}_1, \cdots, \mathbf{x}_L$ *of length* L*. Each* $\mathbf{x}_i$ $(1 \leqslant i \leqslant L)$ *is an independent and identically distributed* $\{-1, 1\}$*-valued random variable satisfying*

$$\Pr[\mathbf{x}_i = 1] = p, \quad \Pr[\mathbf{x}_i = -1] = 1 - p.$$

Without loss of generality, assume $p \geqslant 1/2$*. Then, we have that the expected translation distance* $\mathbb{E}[|\mathbf{x}_1 + \cdots + \mathbf{x}_L|]$ *of this random walk after* L *steps is minimized when* $p = 1/2$*.*

Proof. Create another walk $\mathbf{x}'_1, \cdots, \mathbf{x}'_L$ with $p = 1/2$ that is coupled with $\mathbf{x}_1, \cdots, \mathbf{x}_L$ in the following way.

$$\Pr\left[\mathbf{x}_i = 1 | \mathbf{x}'_i = 1\right] = 1, \quad \Pr\left[\mathbf{x}_i = 1 | \mathbf{x}'_i = -1\right] = 2p - 1.$$

It is easy to see that the distribution of $\mathbf{x}_1, \cdots, \mathbf{x}_L$ is preserved under this coupling.

$$\begin{aligned}\Pr[\mathbf{x}_i = 1] =& \Pr\left[\mathbf{x}'_i = 1\right] \Pr\left[\mathbf{x}_i = 1 | \mathbf{x}'_i = 1\right] + \Pr\left[\mathbf{x}'_i = -1\right] \Pr\left[\mathbf{x}_i = 1 | \mathbf{x}'_i = -1\right] \\ =& \frac{1}{2} \cdot 1 + \frac{1}{2} \cdot (2p - 1) \\ =& p.\end{aligned}$$

Now,

$$\begin{aligned}&\mathbb{E}[|\mathbf{x}_1 + \cdots + \mathbf{x}_L|] - \mathbb{E}\left[|\mathbf{x}'_1 + \cdots + \mathbf{x}'_L|\right] \\ =& \sum_{d \in \{-L, -L+2, \cdots, L-2, L\}} \\ & \qquad \sum_{\substack{x_1, \cdots, x_L \in \{-1,1\} \\ \sum_{i=1}^L x_i = d}} \Pr\left[\mathbf{x}'_1 = x_1, \cdots, \mathbf{x}'_L = x_L\right] \mathbb{E}\left[\left|\sum_{i=1}^L \mathbf{x}_i\right| - |d| \,\middle|\, \mathbf{x}'_1 = x_1, \cdots, \mathbf{x}'_L = x_L\right].\end{aligned}$$

For each translation distance $d \in \{-L, -L+2, \cdots, L-2, L\}$ and trajectory $x_1, \cdots, x_L \in \{-1, 1\}$ such that $\sum_{i=1}^L x_i = d$, let $\ell := \{i \in [L] \colon x_i = -1\}$. Note $2(d + \ell) = L$. We have

$$\begin{aligned}\mathbb{E}\left[\left|\sum_{i=1}^L \mathbf{x}_i\right| - |d| \,\middle|\, \mathbf{x}'_1 = x_1, \cdots, \mathbf{x}'_L = x_L\right] =& ((2p-1) \cdot 1 + (1 - (2p-1)) \cdot (-1))\ell - (-\ell) \\ =& 2(2p-1)\ell,\end{aligned}$$

which is non-negative and attains its minima 0 when $p = 1/2$. This finishes the proof. ◀

D Blinovsky [9] vs. Alon–Bukh–Polyanskiy [2]

In this section we show that, though differing ostensibly, the formulas of the Plotkin points for $(p, L-1)$-list decoding given by Blinovsky and Alon–Bukh–Polyanskiy actually agree with each other. The proof is essentially due to the user `Sean Clark` on `Mathematics Stack Exchange` [14].

For $L = 2k$ or $2k+1$ for some positive integer $k \in \mathbb{Z}_{>0}$, Blinovsky's formula is

$$P_{L-1} = \sum_{i=1}^{k} \frac{\binom{2(i-1)}{i-1}}{i} 2^{-2i};$$

while Alon–Bukh–Polyanskiy wrote it as

$$P_{L-1} = \frac{1}{2} - 2^{-2k-1}\binom{2k}{k}.$$

We are going to show that

▶ **Lemma 104.** *For any $k \geqslant 1$,*

$$\sum_{i=1}^{k} \frac{\binom{2(i-1)}{i-1}}{i} 2^{-2i} = \frac{1}{2} - 2^{-2k-1}\binom{2k}{k}.$$

Proof. To see the above two expressions are always evaluated to the same value, we first massage the above equation. Multiplying 2^{2k+2} on both sides, shifting the summation index and rearranging terms, we have

$$\sum_{i=0}^{k-1} \frac{\binom{2i}{i}}{i+1} 2^{2(k-i)} = 2^{2k+1} - 2\binom{2k}{k}.$$

Adding $\frac{\binom{2k}{k}}{k+1}$ on both sides, we get

$$\begin{aligned}
\sum_{i=0}^{k} \frac{\binom{2i}{i}}{i+1} 2^{2(k-i)} =& 2^{2k+1} - \left(2 - \frac{1}{k+1}\right)\binom{2k}{k} \\
=& 2^{2k+1} - \frac{2k+1}{k+1}\binom{2k}{k} \\
=& 2^{2k+1} - \binom{2k+1}{k+1} && (124) \\
=& 2^{2k+1} - \binom{2k+1}{k}, && (125)
\end{aligned}$$

where Eqn. (124) is by Fact (16) and Eqn. (125) is by Fact (15).

To show

$$\sum_{i=0}^{k} \frac{\binom{2i}{i}}{i+1} 2^{2(k-i)} = 2^{2k+1} - \binom{2k+1}{k}, \tag{126}$$

we conduct induction on k.

1. When $k = 0$, LHS $= 1 =$ RHS.
2. Assume (126) holds for certain $k \geqslant 1$. We want to show it also holds for $k+1$.

$$\begin{aligned}
\sum_{i=0}^{k+1} \frac{\binom{2i}{i}}{i+1} 2^{2(k+1-i)} =& 2^2 \sum_{i=0}^{k} \frac{\binom{2i}{i}}{i+1} 2^{2(k-i)} + \frac{\binom{2(k+1)}{k+1}}{k+2} \\
=& 2^2 \left(2^{2k+1} - \binom{2k+1}{k} \right) + \frac{\binom{2k+2}{k+1}}{k+2} && (127) \\
=& 2^{2(k+1)+1} - 2 \left(\binom{2k+1}{k} + \binom{2k+1}{k+1} \right) + \frac{\binom{2k+2}{k+1}}{k+2} && (128) \\
=& 2^{2(k+1)+1} - \left(2 - \frac{1}{k+2} \right) \binom{2k+2}{k+1} && (129) \\
=& 2^{2(k+1)+1} - \frac{2k+3}{k+2} \binom{2k+2}{k+1} \\
=& 2^{2(k+1)+1} - \binom{2(k+1)+1}{(k+1)+1}. && (130)
\end{aligned}$$

Eqn. (127), (128), (129) and (130) follow from induction hypothesis, Fact (15), Fact (17) and Fact (16), respectively. Hence Eqn. (126) holds for $k+1$ as well. ◀

Online Computation with Untrusted Advice

Spyros Angelopoulos
Sorbonne Université, CNRS, Laboratoire d'informatique de Paris 6, LIP6, Paris, France
spyros.angelopoulos@lip6.fr

Christoph Dürr
Sorbonne Université, CNRS, Laboratoire d'informatique de Paris 6, LIP6, Paris, France
christoph.durr@lip6.fr

Shendan Jin
Sorbonne Université, CNRS, Laboratoire d'informatique de Paris 6, LIP6, Paris, France
shendan.jin@lip6.fr

Shahin Kamali
Department of Computer Science, University of Manitoba, Winnipeg, Canada
shahin.kamali@umanitoba.ca

Marc Renault
Computer Sciences Department, University of Wisconsin – Madison, Madison, WI, USA
mrenault@cs.wisc.edu

Abstract

The advice model of online computation captures the setting in which the online algorithm is given some partial information concerning the request sequence. This paradigm allows to establish tradeoffs between the amount of this additional information and the performance of the online algorithm. However, unlike real life in which advice is a recommendation that we can choose to follow or to ignore based on trustworthiness, in the current advice model, the online algorithm treats it as infallible. This means that if the advice is corrupt or, worse, if it comes from a malicious source, the algorithm may perform poorly. In this work, we study online computation in a setting in which the advice is provided by an untrusted source. Our objective is to quantify the impact of untrusted advice so as to design and analyze online algorithms that are robust and perform well even when the advice is generated in a malicious, adversarial manner. To this end, we focus on well- studied online problems such as ski rental, online bidding, bin packing, and list update. For ski-rental and online bidding, we show how to obtain algorithms that are Pareto-optimal with respect to the competitive ratios achieved; this improves upon the framework of Purohit et al. [NeurIPS 2018] in which Pareto-optimality is not necessarily guaranteed. For bin packing and list update, we give online algorithms with worst-case tradeoffs in their competitiveness, depending on whether the advice is trusted or not; this is motivated by work of Lykouris and Vassilvitskii [ICML 2018] on the paging problem, but in which the competitiveness depends on the reliability of the advice. Furthermore, we demonstrate how to prove lower bounds, within this model, on the tradeoff between the number of advice bits and the competitiveness of any online algorithm. Last, we study the effect of randomization: here we show that for ski-rental there is a randomized algorithm that Pareto-dominates any deterministic algorithm with advice of any size. We also show that a single random bit is not always inferior to a single advice bit, as it happens in the standard model.

2012 ACM Subject Classification Theory of computation → Online algorithms

Keywords and phrases Online computation, competitive analysis, advice complexity, robust algorithms, untrusted advice

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.52

Related Version A full version of the paper is available at `https://arxiv.org/abs/1905.05655`.

Funding Research supported by the CNRS-PEPS Project Advice.

1 Introduction

Suppose that you have an investment account with a significant amount in it, and that your financial institution advises you periodically on investments. One day, your banker informs you that company X will soon receive a big boost, and advises to use the entire account to buy stocks. If you were to completely trust the banker's advice, there are naturally two possibilities: either the advice will prove correct (which would be great) or it will prove wrong (which would be catastrophic). A prudent customer would take this advice with a grain of salt, and would not be willing to risk everything. In general, our understanding of advice is that it entails *knowledge that is not foolproof*.

In this work we focus on the online computation with advice. Our motivation stems from observing that, unlike the real world, the advice under the known models is often closer to "fiat" than "recommendation". Our objective is to propose a model which allows the possibility of incorrect advice, with the objective of obtaining more realistic and robust online algorithms.

1.1 Online computation and advice complexity

In the standard model of online computation that goes back to the seminal work of Sleator and Tarjan [26], an online algorithm receives as input a sequence of *requests*. For each request in this sequence, the algorithm must make an irrevocable decision concerning the item, without any knowledge of future requests. The performance of an online algorithm is usually evaluated by means of the competitive ratio, which is the worst-case ratio of the cost incurred by the algorithm (assuming a minimization problem) to the cost of an ideal solution that knows the entire sequence in advance.

In practice, however, online algorithms are often provided with some (limited) knowledge of the input, such as lookahead on some of the upcoming requests, or knowledge of the input size. While competitive analysis is still applicable, especially from the point of view of the analysis of a known, given algorithm, a new model was required to formally quantify the power and limitations of offline information. The term *advice complexity* was first coined by Dobrev et al. [12], and subsequent formal models were presented by Böckenhauer et al. [6] and Emek et al. [13], with this goal in mind. More precisely, in the advice setting, the online algorithm receives some bits that encode information concerning the sequence of input items. As expected, this additional information can boost the performance of the algorithm, which is often reflected in better competitive ratios.

Under the current models, the advice bits can encode any information about the input sequence; indeed, defining the "right" information to be conveyed to the algorithm plays an important role in obtaining better online algorithms. Clearly, the performance of the online algorithm can only improve with larger number of advice bits. The objective is thus to identify the exact trade-offs between the size of the advice and the performance of the algorithm. This is meant to provide a smooth transition between the purely online world (nothing is known about the input) and the purely "offline" world (everything is known about the input). In the last decade, a substantial number of online optimization problems have been studied in the advice model; we refer the reader to the survey of Boyar et al. [7] for an in-depth discussion of developments in this field.

As argued in detail in [7], there are compelling reasons to study the advice complexity of online computation. Lower bounds establish strict limitations on the power of any online algorithm; there are strong connections between randomized online algorithms and online algorithms with advice (see, e.g., [16]); online algorithms with advice can be of practical

interest in settings in which it is feasible to run multiple algorithms and output the best solution (see [17] about obtaining improved data compression algorithms by means of list update algorithms with advice); and the first complexity classes for online computation have been based on advice complexity [8].

Notwithstanding such interesting attributes, the known advice model has certain drawbacks. The advice is always assumed to be some error-free information that may be used to encode some property often explicitly connected to the optimal solution. In many settings, one can argue that such information cannot be readily available, which implies that the resulting algorithms are often impractical.

1.2 Online computation with untrusted advice

In this work, we address what is a significant drawback in the online advice model. Namely, all previous works assume that advice is, in all circumstances, completely trustworthy, and precisely as defined by the algorithm. Since the advice is infallible, no reasonable online algorithm with advice would choose to ignore the advice.

It should be fairly clear that such assumptions are very unrealistic or undesirable. Advice bits, as all information, are prone to transmission errors. In addition, the known advice models often require that the information encodes some information about the input, which, realistically, cannot be known exactly (e.g., some bits of the optimal, offline solution). Last, and perhaps more significantly, a malicious entity that takes control of the advice oracle can have a catastrophic impact. For a very simple example, consider the well-known ski rental problem: this is a simple, yet fundamental resource allocation, in which we have to decide ahead of time whether to rent or buy equipment without knowing the time horizon in advance. In the traditional advice model, one bit suffices to be optimal: 0 for renting throughout the horizon, 1 for buying right away. However, if this bit is wrong, then the online algorithm has unbounded competitive ratio, i.e., can perform extremely badly. In contrast, an online algorithm that does not use advice at all has competitive ratio at most 2, i.e., its output can be at most twice as costly as the optimal one.

The above observations were recently made in the context of online algorithms with machine-learned predictions. Lykouris and Vassilvitskii [21] and Purohit et al. [23] show how to use predictors to design and analyze algorithms with two properties: (i) if the predictor is good, then the online algorithm should perform close to the best offline algorithm (what is called *consistency*); and (ii) if the predictor is bad, then the online algorithm should gracefully degrade, i.e., its performance should be close to that of the online algorithm without predictions (what is called *robustness*).

Motivated by these definitions from machine learning, in this work we analyze online algorithms based on their performance in both settings of trusted and untrusted advice. In particular, we will characterize the performance of an online algorithm A by a pair of competitive ratios, denoted by (r_A, w_A), respectively. Here, r_A is the competitive ratio achieved assuming that the advice encodes precisely what it is meant to capture; we call this ratio the competitive ratio with *trusted* (thus, always correct) advice. In contrast, w_A is the competitive ratio of A when the advice is untrusted (thus, potentially wrong). More precisely, in accordance with the worst-case nature of competitive analysis, we allow the incorrect advice to be chosen *adversarially*. Namely, assuming a deterministic online algorithm A, the incorrect advice string is generated by a malicious, adversarial entity.

To formalize the above concept, assume the standard advice model, in which a deterministic online algorithm A processes a sequence of requests $\sigma = (\sigma[i])_{i\in[1,n]}$ using an advice tape. At each time t, A serves request $\sigma[t]$, and its output is a function of $\sigma[1..\ldots t-1]$ and

$\phi \in \{0,1\}^*$. Let $A(\sigma,\phi)$ denote the cost incurred by A on input σ, using an advice string ϕ. Denote by r_A, w_A as

$$r_A = \sup_{\sigma} \inf_{\phi} \frac{A(\sigma,\phi)}{\textsc{Opt}(\sigma)}, \quad \text{and} \quad w_A = \sup_{\sigma} \sup_{\phi} \frac{A(\sigma,\phi)}{\textsc{Opt}(\sigma)}, \tag{1}$$

where $\textsc{Opt}(\sigma)$ denotes the optimal offline cost for σ. Then we say that algorithm A is (r,w)-competitive for every $r \geq r_A$ and $w \geq w_A$. In addition, we say that A has advice complexity $s(n)$ if for every request sequence σ of length n, the algorithm A depends only on the first $s(n)$ bits of the advice string ϕ. To illustrate this definition, the opportunistic 1-bit advice algorithm for ski rental that was described above is $(1,\infty)$-competitive, whereas the standard competitively optimal algorithm without advice is $(2,2)$-competitive. In general, every online algorithm A without advice or ignoring its advice is trivially (w,w)-competitive, where w is the competitive ratio of A.

Hence, we can associate every algorithm A to a point in the 2-dimensional space with coordinates (r_A, w_A). These points are in general incomparable, e.g., it is difficult to argue that a $(2,10)$-competitive algorithm is better than a $(4,8)$-competitive algorithm. However, one can appeal to the notion of *dominance*, by saying that algorithm A dominates algorithm B if $r_A \leq r_B$ and $w_A \leq w_B$. More precisely, we are interested in finding the Pareto frontier in this representation of all online algorithms. For the ski rental example, the two above mentioned algorithms belong to the Pareto set.

A natural goal is to describe this Pareto frontier, which in general, may be comprised of several algorithms with vastly different statements. Ideally, however, one would like to characterize it by a single *family* $\mathcal{A}$ of algorithms, with similar statements (e.g., algorithms in $\mathcal{A}$ are obtained by appropriately selecting a parameter). We say that $\mathcal{A}$ is *Pareto-optimal* if it consists of pairwise incomparable algorithms, and for every algorithm B, there exists $A \in \mathcal{A}$ such that A dominates B. Regardless of optimality, given $\mathcal{A}$, we will describe its competitiveness by means of a function $f : \mathbb{R}_{\geq 1} \to \mathbb{R}_{\geq 1}$ such that for every ratio r there is an $(r, f(r))$-competitive algorithm in $\mathcal{A}$. This function will in general depend on parameters of the problem, such as, for example, the buying cost B in the ski rental problem.

1.3 Contribution

We study various online problems in the setting of untrusted advice. We also demonstrate that it is possible to establish both upper and lower bounds on the tradeoff between the size of the advice and the competitiveness in this new advice model. We begin in Section 2 with a simple, yet illustrative online problem as a case study, namely the *ski-rental* problem. Here, we give a Pareto-optimal algorithm with only one bit of advice. We also show that this algorithm is Pareto-optimal even in the space of all (deterministic) algorithms with advice of *any* size.

In Section 3 we study the *online bidding* problem, in which the objective is to guess an unknown, hidden value, using a sequence of bids. This problem was introduced in [11] as a vehicle for formalizing efficient doubling, and has applications in several important online and offline optimization problems. As with ski rental, this is another problem for which a trivial online algorithm is $(1,\infty)$-competitive. We first show how to find a Pareto-optimal strategy, when the advice encodes the hidden value, and thus can have unbounded size. Moreover, we study the competitiveness of the problem with only k bits of advice, for some fixed k, and show both upper and lower bounds on the achieved competitive ratios. The results illustrate that is is possible to obtain non-trivial lower bounds on the competitive ratios, in terms of the advice size. In particular, the lower bound implies that, unlike the ski rental problem, Pareto-optimality is not possible with a bounded number of advice bits.

In Sections 4 and 5, we study the *bin packing* and *list update* problems; these problems are central in the analysis of online problems and competitiveness, and have numerous applications in practice. For these problems, an efficient advice scheme should address the issues of "what constitutes good advice" as well as "how the advice should be used by the algorithm". We observe that the existing algorithms with advice perform poorly in the case the advice is untrusted. To address this, we give algorithms that can be "tuned" based on how much we are willing to trust the advice. This enables us to show guarantees in the form $(r, f(r))$-competitiveness, where r is strictly better than the competitive ratio of all deterministic online algorithms and $f(r)$ smoothly decreases as r grows, while still being close to the worst-case competitive ratio. To illustrate this, consider the bin packing problem. Our $(r, f(r))$-competitive algorithm has $f(r) = \max\{33 - 18r, 7/4\}$ for any $r \geq 1.5$. If $r = 1.5$, our algorithm is $(1.5, 6)$-competitive, and matches the performance of a known algorithm [10]. However, with a slight increase of r, one can improve competitiveness in the event the advice is untrusted. For instance, choosing $r = 1.55$, we obtain $f(r) = 5.1$. In other words, the algorithm designer can hedge against untrusted advice, by a small sacrifice in the trusted performance. Thus we can interpret r as the "risk" for trusting the advice: the smaller the r, the bigger the risk. Likewise, for the list update problem, our $(r, f(r))$-competitive algorithm has $f(r) = 2 + \frac{10-3r}{9r-5}$ for $r \in [5/3, 2]$. If the algorithm takes maximum risk, i.e., if r is smallest, the algorithm is equivalent to an existing $(5/3, 5/2)$-competitive algorithm [9]. Again, by increasing r, we better safeguard against the event of untrusted advice.

All the above results pertain to deterministic online algorithms. In Section 6, we study the power of randomization in online computation with untrusted advice. First, we show that the randomized algorithm of Purohit et al. [23] for the ski rental problem Pareto-dominates any deterministic algorithm, even when the latter is allowed unbounded advice. Furthermore, we show an interesting difference between the standard advice model and the model we introduce: in the former, an advice bit can be at least as powerful as a random bit, since an advice bit can effectively simulate any efficient choice of a random bit. In contrast, we show that in our model, there are situations in which a randomized algorithm with L advice bits and one random bit is Pareto-incomparable to the Pareto-optimal deterministic algorithm with $L + 1$ advice bits. This confirms the intuition that a random bit is considered trusted, and thus not obviously inferior to an advice bit.

While our work addresses issues similar to [21] and [23], in that trusted advice is related to consistency whereas untrusted advice is related to robustness, it differs in two significant aspects: First, our ideal objective is to identify an optimal family of algorithms, and we show that in some cases (ski rental, online bidding), this is indeed possible; when this is not easy or possible, we can still provide approximations. Note that finding a Pareto-optimal family of algorithms presupposes that the exact competitiveness of the online problem with no advice is known. For problems such as bin packing, the exact optimal competitive ratios are not known. Hence, a certain degree of approximation is unavoidable in such cases. In contrast, [21, 23] focus on "smooth" tradeoffs between the trusted and untrusted competitive ratios, but do not address the issues related to optimality and approximability of these tradeoffs.

Second, our model considers the size of advice and its impact on the algorithm's performance, which is the main focus of the advice complexity field. For all problems we study, we parameterize advice by its size, i.e., we allow advice of a certain size k. Specifically, the advice need not necessarily encode the optimal solution or the request sequence itself. This opens up more possibilities to the algorithm designer in regards to the choice of an appropriate advice oracle, which may have further practical applications in machine learning.

2 A warm-up: the ski rental problem

2.1 Background

The ski rental problem is a canonical example in online rent-or-buy problems. Here, the request sequence can be seen as vacation days, and on each day the vacationer (that is, the algorithm) must decide whether to continue renting skis, or buy them. Without loss of generality we assume that renting costs a unit per day, and buying costs $B \in \mathbb{N}^+$. The number of skiing days, which we denote by D, is unknown to the algorithm, and we observe that the optimal offline cost is $\min\{D, B\}$. Generalizations of ski rental have been applied in many settings, such as dynamic TCP acknowledgment [19], the parking permit problem [22], and snoopy caching [18].

Consider the single-bit advice setting. Suppose that the advice encodes whether to buy on day 1, or always rent. An algorithm that blindly follows the advice is optimal if the advice is trusted, but, if the advice is untrusted, the competitive ratio is as high as D/B, if $D > B$. Hence, this algorithm is $(1, \infty)$-competitive, for $D \to \infty$.

2.2 Ski rental with untrusted advice

We define the family of algorithms A_k, with parameter $0 < k \leq B$ as follows. There is a single bit of advice, which is the indicator of the event $D < B$. If the advice bit is 1, then A_k rents until until day $B - 1$ and buys on day B. Otherwise, the algorithms buys on day k.

▶ **Proposition 1.** *Algorithm A_k is $(1 + \frac{k-1}{B}, 1 + \frac{B-1}{k})$-competitive.*

Our algorithm A_k is slightly different from the one proposed in [23], which buys on day $\lceil B/k \rceil$ if the advice is 1 and is shown to be $(1+k/B, 1+B/k)$-competitive. More importantly, we show that A_k is Pareto-optimal in the space of all deterministic online algorithms with advice of *any size*. This implies that more than a single bit of advice will not improve the tradeoff between the trusted and untrusted competitive ratios.

▶ **Theorem 2.** *For any deterministic $(1 + \frac{k-1}{B}, w)$-competitive algorithm A, with $1 \leq k \leq B$, with advice of any size, it holds that $w \geq 1 + \frac{B-1}{k}$.*

Proof. Let A be an algorithm with trusted competitive ratio at most $1 + \frac{k-1}{B}$. First, note that if the advice is untrusted, the competitive ratio cannot be better than the competitive ratio of a purely online algorithm. For ski-rental, it is known that no online algorithm can achieve a competitive ratio better than $1 + (B-1)/B$ [18]. So, in the case $k = B$, the claim trivially holds. In the remainder of the proof, we assume $k < B$.

We use σ_D to denote the instance of the problem in which the number of skiing days is D, and use $A_t(\sigma_D)$ to denote the cost of A for σ_D in case of trusted advice.

Consider a situation in which the input is σ_{B+k} and the advice for A is trusted. Let j be the day the algorithm will buy under this advice. Since the advice is trusted and thus $\textsc{Opt}(\sigma_{B+k}) = B$, it must be that

$$A_t(\sigma_{B+k}) \leq \left(1 + \frac{k-1}{B}\right) \textsc{Opt}(\sigma_{B+k}),$$

which implies $j < B + k$. In other words, A indeed buys on day j. We conclude that $A_t(\sigma_{B+k}) = j - 1 + B$ which further implies $j \leq k$.

Let x be the trusted advice A receives on input σ_{B+k} and suppose A receives the same advice x on input σ_j. Note that x can be trusted or untrusted for σ_j. The important point is that A serves σ_j in the same way it serves σ_{B+k}, that is, it rents for $j - 1$ days and buys

on day j. The cost of A for σ_j is then $j-1+B$, while $\textsc{Opt}(\sigma_j)=j$. The ratio between the cost of the algorithm and Opt is therefore $1+\frac{B-1}{j}$, which is at least $1+\frac{B-1}{k}$ since $j\leq k$. Note that $1+\frac{B-1}{k}>1+\frac{k-1}{B}$ (since we assumed $k<B$) and therefore the advice in this situation has to be untrusted, by the assumption on the trusted competitive ratio of A. We conclude that the untrusted competitive ratio must be at least $1+(B-1)/k$. ◀

3 Online bidding

3.1 Background

In the *online bidding* problem, a player wants to guess a hidden, unknown real value $u\geq 1$. To this end, the player submits a sequence $X=(x_i)$ of increasing *bids*, until one of them is at least u. The strategy of the player is defined by this sequence of bids, and the cost of guessing the hidden value u is equal to $\sum_{i=1}^{j}x_i$, where j is such that $x_{j-1}<u\leq x_j$. Hence the following natural definition of the competitive ratio of the bidder's strategy.

$$w_X=\sup_u \frac{\sum_{i=1}^{j}x_i}{u},\ \text{where } j \text{ is such that } x_{j-1}<u\leq x_j.$$

The problem was introduced in [11] as a canonical problem for formalizing doubling-based strategies in online and offline optimization problems, such as searching for a target on the line, minimum latency, and hierarchical clustering. It is worth noting that online bidding is identical to the problem of minimizing the *acceleration ratio of interruptible algorithms* [25]; the latter and its generalizations are problems with many practical applications in AI (see, for instance [20]).

Without advice, the best competitive ratio is 4, and can be achieved using the doubling strategy $x_i=2^i$. If the advice encodes[1] the value u, and assuming trusted advice, bidding $x_1=u$ is a trivial optimal strategy. The above observations imply that there are simple strategies that are $(4,4)$-competitive and $(1,\infty)$-competitive, respectively.

3.2 Online bidding with untrusted advice

Suppose that $w\geq 4$ is a fixed, given parameter. We will show a Pareto-optimal bidding strategy X^*_u, assuming that the advice encodes u, which is $(\frac{w-\sqrt{w^2-4w}}{2},w)$-competitive (Theorem 5).

We begin with some definitions. Since the index of the bid which reveals the value will be important in the analysis, we define the class $S_{m,u}$, with $m\in\mathbb{N}^+$ as the set of bidding strategies with advice u which are w-competitive, and which, if the advice is trusted, succeed in finding the value with precisely the m-th bid. We say that a strategy $X\in S_{m,u}$ that is (r,w)-competitive dominates $S_{m,u}$ if for every $X'\in S_{m,u}$, such that X' is (r',w)-competitive, $r\leq r'$ holds.

The high-level idea is to identify, for any given m, a dominant strategy in $S_{m,u}$. Let $X^*_{m,u}$ denote such a strategy, and denote by $(r^*_{m,u},w)$ its competitiveness. Then $X^*_{m,u}$ and $r^*_{m,u}$ are the solutions to an infinite linear program which we denote by $P_{m,u}$, and which is shown below. For convenience, for any strategy X, we will always define x_0 to be equal to 1.

[1] We assume that the advice provides the exact value u to the algorithm. For practical considerations, it suffices to assume an oracle that provides an $(1+\epsilon)$-approximation of the hidden value, for sufficiently small $\epsilon>0$. This will only affect the competitive ratios by the same negligible factor.

$$
\begin{aligned}
\min \quad & r_{m,u} && (P_{m,u})\\
\text{s.t. } & x_i < x_{i+1}, \quad i \in \mathbb{N}^+\\
& x_{m-1} < u \le x_m\\
& \sum_{j=1}^{m} x_j \le r_{m,u} \cdot u\\
& \sum_{j=1}^{i} x_j \le w \cdot x_{i-1}, \quad i \in \mathbb{N}^+\\
& x_i \ge 0, \quad i \in \mathbb{N}^+.
\end{aligned}
\qquad
\begin{aligned}
\min \quad & \frac{1}{u} \cdot \sum_{i=1}^{m} x_i && (L_{m,u})\\
\text{s.t. } & x_i < x_{i+1}, \quad i \in \mathbb{N}^+\\
& x_m = u\\
& \sum_{j=1}^{i} x_i \le w \cdot x_{i-1}, \quad i \in \mathbb{N}^+ && (C_i)\\
& x_i \ge 0, \quad i \in \mathbb{N}^+.
\end{aligned}
$$

Note that in $P_{m,u}$ the constraints $\sum_{j=1}^{i} x_j \le w \cdot x_{i-1}$ guarantee that the untrusted competitive ratio of X is at most w, whereas the constraints $\sum_{j=1}^{m} x_j \le r_{m,u} \cdot u$ and $x_{m-1} < u \le x_m$ guarantee that if the advice is trusted, then X succeeds in finding u precisely with its m-th bid, and in this case the competitive ratio is $r_{m,u}$.

We also observe that an optimal solution $X^*_{m,u} = (x^*_i)_{i\ge 1}$ for $P_{m,u}$ must be such that $x_m = u$, otherwise one could define a strategy $X'_{m,u}$ in which $x'_i = x^*_i/\alpha$, for all $i \ge 1$, with $\alpha = u/x^*_m$, which is still feasible for $P_{m,u}$, is such that $x'_m = u$, and has better objective value than $X^*_{m,u}$, a contradiction. Furthermore, in an optimal solution, the constraint $\sum_{i=1}^{m} x_i \le r_{m,u} \cdot u$ must hold with equality. Therefore, $X^*_{m,u}$ and $r^*_{m,u}$ are also solutions to the linear program $L_{m,u}$.

Next, define $r^*_u = \inf_m r^*_{m,u}$, and $r^* = \sup_u r^*_u$. Informally, r^*_u, r^* are the optimal competitive ratios, assuming trusted advice. More precisely, the dominant strategy in the space of all w-competitive strategies is (r^*_u, w)-competitive, and r^* is an upper bound on r^*_u, assuming the worst-case choice of u.

We first argue how to compute $r^*_{m,u}$ and the corresponding strategy $X^*_{m,u}$, provided that $L_{m,u}$ is feasible. This is accomplished in Lemma 3. The main idea behind the technical proof is to show that in an optimal solution of $L_{m,u}$, all constraints C_i hold with equality. This allows us to describe the bids of the optimal strategy by means of a linear recurrence relation which we can solve so as to obtain an expression for the bids of $X^*_{m,u}$.

Define the sequences a_i and b_i as follows:

$$
a_i = \frac{a_{i-1}}{w - 1 - b_{i-1}}, \text{ with } a_0 = 1, \text{ and } \quad b_i = \frac{1 + b_{i-1}}{w - 1 - b_{i-1}}, \text{ with } b_0 = 0, \tag{2}
$$

Moreover, for $w > 4$, let $\rho_1 = \frac{w - \sqrt{w^2-4w}}{2}$ and $\rho_2 = \frac{w + \sqrt{w^2-4w}}{2}$ denote the two roots of $x^2 - wx + w$, the characteristic polynomial of the above linear recurrence.

▶ **Lemma 3.** *For every m define $X_{m,u}$ as follows:*

- *If $w > 4$, then $x_{m,u,i} = \alpha \cdot \rho_1^{i-1} + \beta \cdot \rho_2^{i-1}$, where $\alpha = \frac{a_{m-1}\rho_2^{m-1} - 1}{\rho_2^{m-1} - \rho_1^{m-1}} \cdot u$, and $\beta = \frac{a_{m-1}\rho_1^{m-1} - 1}{\rho_1^{m-1} - \rho_2^{m-1}} \cdot u$,*
- *If $w = 4$, then $x_{m,u,i} = (\alpha + \beta \cdot i) \cdot 2^i$, where $\alpha = \frac{2^{m-1} \cdot m \cdot a_{m-1} - 1}{2^m (m-1)} \cdot u$, and $\beta = \frac{1 - 2^{m-1} \cdot a_{m-1}}{2^m (m-1)} \cdot u$.*

Then, $X_{m,u}$ is an optimal feasible solution if and only if $a_{m-1} \cdot u \le w$.

We can now give the statement of the optimal strategy X^*_u. First, we can argue that the optimal objective value of $L_{m,u}$ is monotone increasing in m, thus it suffices to find the objective value of the smallest m^* for which $L_{m^*,u}$ is feasible; This can be accomplished with a binary search in the interval $[1, \lceil \log u \rceil]$, since we know that the doubling strategy in which

the i-th bid equals 2^i is w-competitive for all $w \geq 4$; hence $m^* \leq \lceil \log u \rceil$. Then X_u^* is derived as in the statement of Lemma 3. The time complexity of the algorithm is $O(\log \log u)$, since we can describe each a_i, b_i, and hence a_{m-1} in closed form, avoiding the recurrence which would add a $O(\log u)$ factor. The technical details can be found in the long version of this paper [2].

Last, the following lemma allows us to express r^* as a function of the values of the sequence b, which we can further exploit so as to obtain the exact value of r_u^*.

▶ **Lemma 4.** *It holds that* $r^* = 1 + \sum_{i=1}^{\infty} \prod_{j=1}^{i-1} b_j$. *Furthermore,* $r^* = \frac{w - \sqrt{w^2 - 4w}}{2}$.

Combining Lemmas 3 and 4 we obtain following result:

▶ **Theorem 5.** *Strategy* X_u^* *is Pareto-optimal and is* $(\frac{w - \sqrt{w^2 - 4w}}{2}, w)$*-competitive.*

Strategy X_u^* requires u as advice, which can be unbounded. A natural question is what competitiveness can one achieve with k advice bits, for some fixed k. We address this question both from the point of view of upper and lower bounds. Concerning upper bounds, we show the following:

▶ **Theorem 6.** *For every* $w \geq 4$*, there exists a bidding strategy with* k *bits of advice which is* (r, w)*-competitive, where*

$$r = \begin{cases} \frac{\left(w + \sqrt{w^2 - 4w}\right)^{1 + 1/K}}{2^{1/K}\left(w + \sqrt{w^2 - 4w} - 2\right)} & \text{if } w \leq (1+K)^2/K \\ \frac{(1+K)^{1+1/K}}{K} & \text{if } w \geq (1+K)^2/K. \end{cases}$$

and where $K = 2^k$.

In particular, for $w = 4$, the strategy of Theorem 6 is $(2^{1 + \frac{1}{2^k}}, 4)$-competitive, whereas X_u^* is $(2, 4)$-competitive. The following theorem gives a lower bound on the competitiveness of any bidding strategy with k bits. The result shows that one needs unbounded number of bids to achieve $(2, 4)$-competitiveness.

▶ **Theorem 7.** *For any bidding strategy with* k *advice bits that is* $(r, 4)$*-competitive it holds that* $r \geq 2 + \frac{1}{3 \cdot 2^k}$.

Proof sketch. We present only an outline. With k bits of advice, the online algorithm can differentiate only between $K = 2^k$ online bidding sequences, denoted by $X_1, \ldots, X_K$, each of which must have (untrusted) competitive ratio at least 4. Suppose, by way of contradiction, that the algorithm has trusted competitive ratio less than $2 + \frac{1}{3 \cdot 2^k}$. We reach a contradiction, by applying a game between the algorithm and the adversary, which proceeds in rounds. The adversary fixes a sufficiently large index $i \geq i_0$, for some i_0. In the first round, u is chosen by the adversary so as to be infinitesimally larger than $x_{K,i-1}$, namely the $(i-1)$-th bid of X_K. For the algorithm to guarantee the claimed r, we show that it will have to use the advice so as to "choose" one of the sequences $X_1, \ldots X_{K-1}$, say X_j. Then in the next round, the adversary will choose an appropriate u that is adversarial for X_j. The crux of the proof is to show that the algorithm's only response is to choose a sequence of index higher than j. Eventually, the only remaining choice for the algorithm is strategy X_K; moreover, we can show that throughout the execution of the algorithm the adversarial u is comparable to $x_{K,i-1}$, in particular, we show that $u \leq e^{1/3} x_{K,i-1}$. To conclude, the above argument shows that

$$r \geq \sup_{i \geq i_0} \frac{\sum_{j=1}^{i} x_{K,j}}{e^{\frac{1}{3}} x_{K,i-1}} = \frac{1}{e^{\frac{1}{3}}} \sup_{i=1} \frac{\sum_{j=1}^{i} x_{K,j}}{x_{K,i-1}} \geq \frac{4}{e^{\frac{1}{3}}} > 2 + \frac{1}{3K}.$$

where we used the fact that $\sup_{i=1} \frac{\sum_{j=1}^{i} x_{K,j}}{x_{K,i-1}} = 4$, since X_K is 4-competitive. ◀

4 Online bin packing

4.1 Background

An instance of the online bin packing problem consists of a sequence of items with different *sizes* in the range $(0, 1]$, and the objective is to pack these items into a minimum number of bins, each with a capacity of 1. For each arriving item, the algorithm must place it in one of the current bins or open a new bin for the item. We say that algorithm A has an asymptotic competitive ratio r if, on every sequence σ, the number of opened bins satisfies $A(\sigma) \leq r \cdot \textsc{Opt}(\sigma) + c$, where c is a constant. As standard in the analysis of bin packing problems, throughout this section, by "competitive ratio" we mean "asymptotic competitive ratio". The First Fit [15] algorithm maintains bins in the same order that they have been opened, and places an item into the first bin with enough free space; if no such bin exists, it opens a new bin. First Fit has a competitive ratio of 1.7 [15] while the best online algorithm has a competitive ratio of at least 1.54278 [5] and at most 1.57829 [4]. Online bin packing has also been studied in the advice setting [10, 24, 3]. In particular, it is possible to achieve a competitive ratio of 1.4702 with only a constant number of (trusted) advice bits [3].

In this section, we introduce an algorithm named Robust-Reserve-Critical (Rrc) which has a parameter $\alpha \in [0, 1]$, indicating how much the algorithm relies on the advice. Provided with $O(1)$ bits of advice, the algorithm is asymptotically $(r_{\text{Rrc}}, w_{\text{Rrc}})$-competitive for $r_{\text{Rrc}} = 1.5 + \frac{1-\alpha}{4-3\alpha}$ and $w_{\text{Rrc}} = 1.5 + \max\{\frac{1}{4}, \frac{9\alpha}{8-6\alpha}\}$. If the advice is reliable, we set $\alpha = 1$ and the algorithm is asymptotically $(1.5, 6)$-competitive; otherwise, we set α to a smaller value.

4.2 The Reserve-Critical algorithm

Our solution uses an algorithm introduced by Boyar et al. [10] which achieves a competitive ratio of 1.5 using $O(\log n)$ bits of advice [10]. We refer to this algorithm as Reserve-Critical in this paper and describe it briefly. The algorithm classifies items according to their size. Tiny items have their size in the range $(0, 1/3]$, small items in $(1/3, 1/2]$, critical items in $(1/2, 2/3]$, and large items in $(2/3, 1]$. In addition, the algorithm has four kinds of bins, called tiny, small, critical and large bins. Large items are placed alone in large bins, which are opened at each arrival. Small items are placed in pairs in small bins, which are opened every other arrival. Critical bins contain a single critical item, and tiny items up to a total size of $1/3$ per bin, while tiny bins contain only tiny items. The algorithm receives as advice the number of critical items, denoted by c, and opens c *critical bins* at the beginning. Inside each critical bin, a space of $2/3$ is reserved for a critical item, and tiny items are placed using First-Fit into the remaining space of these bins possibly opening new bins dedicated to tiny items. Each critical item is placed in one of the critical bins. Note that the algorithm is heavily dependent on the advice being trusted. Imagine that the encoded advice is strictly larger than the real number of critical items. This results in critical bins which contain only tiny items. The worst case is reached when all tiny items have size slightly more than $1/6$ while there is no critical item. In this case, all critical bins are filled up to a level slightly more than $1/6$. Hence, untrusted advice can result in a competitive ratio as bad as 6.

4.3 The Robust-Reserve-Critical (RRC) algorithm

Let t be the number of tiny bins opened by the Reserved-Critical algorithm. Recall that c is the number of critical bins. We call the fraction $c/(c+t)$ the *critical ratio*. The advice for Rrc is a fraction γ, integer multiple of $1/2^k$, that is encoded in k bits such that if the advice is trusted then $\gamma \leq c/(c+t) \leq \gamma + 1/2^k$. In case $c/(c+t)$ is a positive integer multiple of

$1/2^k$, we break the tie towards $\gamma < c/(c+t)$. Note that for sufficiently large, yet constant, number of bits, γ provides a good approximation of the critical ratio. Indeed having γ as advice is sufficient to achieve a competitive ratio that approaches 1.5 in the trusted advice model, as shown in [3].

The RRC algorithm has a parameter $0 \leq \alpha \leq 1$, which together with the advice γ can be used to define a fraction $\beta = \min\{\alpha, \gamma\}$. The algorithm maintains a proportion close to β of critical bins among critical and tiny bins. Formally, on the arrival of a critical item, the algorithm places it in a critical bin, opening a new one if necessary. Each arriving tiny item x is packed in the first critical bin which has enough space, with the restriction that the tiny items don't exceed a fraction $1/3$ in these bins. If this fails, the algorithm tries to pack x in a tiny bin using First-Fit strategy (this time on tiny bins). If this fails as well, a new bin B is opened for x. Now, B should be declared as a critical or a tiny bin. Let c' and t' denote the number of critical and tiny bins before opening B. If $c' + t' > 0$ and $\frac{c'}{c'+t'} < \beta$, then B is declared a critical bin; otherwise, B is declared a tiny bin. Large and small items are placed similarly to the Reserved-Critical algorithm (one large item in each large bin and two small items in each small bin).

4.4 Analysis

Intuitively, RRC works similarly to Reserved-Critical except that it might not open as many critical bins as suggested in the advice. The algorithm is more "conservative" in the sense that it does not keep two-third of many (critical) bins open for critical items that might never arrive. The smaller the value of α is, the more conservative the algorithm is. Our analysis is based on two possibilities in the final packing of the algorithm. In the first case (case I), all critical bins receive a critical item, while in the second case (case II) some of them have their reserved space empty. In case I, we show the number of bins in the packing of RRC is within a factor $1.5 + \frac{1-\beta}{4-3\beta}$ of the number of bins in the optimal packing. Note that this ratio decreases as the value of α (and β) grows. This implies a less conservative algorithm would be better packing in this case. Case II happens only if the advice is untrusted. In this case, the number of bins in the RRC packing is within a factor $1.5 + \frac{9\beta}{8-6\beta}$ of the number of bins in an optimal packing. This ratio increases with α (and β). This implies a more conservative algorithm would be better in this case as it would open less critical bins and, thus, would have fewer without critical items.

Assume the advice is trusted. Then either $\gamma \leq \alpha$ or $\gamma > \alpha$. In the former case, the algorithm maintains the same ratio as suggested by advice, and a result from [3] indicates that the competitive ratio is at most $1.5 + \frac{15}{2^{k/2+1}}$. In the former case, the algorithm maintains a smaller number of critical items than what the advice suggested; all these bins receive critical items and the final packing will be in Case I. Consequently, when the advice is trusted, the competitive ratio is at most $1.5 + \max\{\frac{1-\alpha}{4-3\alpha}, \frac{15}{2^{k/2+1}}\}$. If the advice is untrusted, both case I and case II can be realized for the final packing. The competitive ratio will be at most $1.5 + \max\{\frac{1}{4}, \frac{9\alpha}{8-6\alpha}\}$. We can conclude with the following theorem:

▶ **Theorem 8.** *Algorithm Robust-Reserve-Critical with parameter $\alpha \in [0, 1]$ and k bits of advice achieves a competitive ratio of $r_{\text{RRC}} \leq 1.5 + \max\{\frac{1-\alpha}{4-3\alpha}, \frac{15}{2^{k/2+1}}\}$ when the advice is trusted and a competitive ratio of $w_{\text{RRC}} \leq 1.5 + \max\{\frac{1}{4}, \frac{9\alpha}{8-6\alpha}\}$ when the advice is untrusted.*

Assuming the size k of the advice is a sufficiently large constant, we conclude the following.

▶ **Corollary 9.** *For bin packing with untrusted advice, there is a $(r, f(r))$-competitive algorithm where $r \geq 1.5$ and $f(r) = \max\{33 - 18r, 7/4\}$.*

5 List update

5.1 Background

The list update problem consists of a list of items of length m, and a sequence of n requests that should be served with minimum total cost. Every request corresponds to an "access" to an item in the list. If the item is at position i of the list then its access cost is i. After accessing the item, the algorithm can move it closer to the front of the list with no cost using a "free exchange". In addition, at any point, the algorithm can swap the position of any two consecutive items in the list using a "paid exchange" which has a cost of 1. Throughout this section, we adopt the standard assumption that m is a large integer but still a constant with respect to n.

Move-to-Front (MTF) is an algorithm that moves every accessed item to the front of the list using a free exchange. MTF has a competitive ratio of at most 2 [27], which is the best that a deterministic algorithm can achieve [14]. Timestamp [1] is another algorithm that achieves the optimal competitive ratio of 2. This algorithm uses a free exchange to move an accessed item x to the front of the first item that has been accessed at most once since the last access to x. Move-To-Front-Every-Other-Access (MTF2) is a class of algorithms which maintain a bit for each item in the list. Upon accessing an item x, the bit of x is flipped, and x is moved to front if its bit is 0 after the flip (otherwise the list is not updated). If all bits are 0 at the beginning, MTF2 is called called Move-To-Front-Even (MTFE), and if all bits are 1 at the beginning, MTF2 is called Move-To-Front-Odd (MTFO). Both MTFE and MTFO algorithms have a competitive ratio of 5/2 [9]. In [9] it is shown that, for any request sequence, at least one of Timestamp, MTFO, and MTFE has a competitive ratio of at most 5/3. For a given request sequence, the best option among the three algorithms can be indicated with two bits of advice, giving a 5/3-competitive algorithm. However, if the advice is untrusted, the competitive ratio can be as bad as 5/2.

To address this issue, we introduce an algorithm named Toggle (TOG) that has a parameter $\beta \in [0, 1/2]$, and uses 2 advice bits to select one of the algorithms Timestamp, MTFE or MTFO. This algorithm achieves a competitive ratio of $r_{\text{TOG}} = 5/3 + \frac{5\beta}{6+3\beta}$ when the advice is trusted and a competitive ratio of at most $w_{\text{TOG}} = 2 + 2/(4 + 5\beta)$ when the advice is untrusted. The parameter β can be tuned and should be smaller when the advice is more reliable. In particular, when $\beta = 0$, we get a $(5/3, 2.5)$-competitive algorithm.

5.2 The Toggle algorithm

Given the parameter β, the Toggle algorithm (TOG) works as follows. If the advice indicates Timestamp, the algorithm runs Timestamp. If the advice indicates either MTFO or MTFE, the algorithm will proceed in phases (the length of which partially depend on β) alternating ("toggling") between running MTFE or MTFO, and MTF. In what follows, we use MTF2 to represent the algorithm indicated by the advice. The algorithm TOG will initially begin with MTF2 until the cost of the accesses of the phase reaches a certain threshold, then a new phase begins and TOG switches to MTF. This new phase ends when the access cost of the phase reaches a certain threshold, and TOG switches back to MTF2. This alternating pattern continues as TOG serves the requests. As such, TOG will use MTF2 for the odd phases which we will call *trusting phases*, and MTF for the even phases which we will call *ignoring phases*. The actions during each phase are formally defined below.

Trusting phase: In a trusting phase, TOG will use MTF2 to serve the requests. Let σ_i be the first request of some trusting phase j for $1 \leq i \leq n$ and an odd $j \geq 1$. Before serving σ_i, TOG modifies the list with paid exchanges to match the list configuration that would result

from running MTF2 on the request sequence $\sigma_1, \ldots, \sigma_i$. The number of paid exchanges will be less than m^2. In addition, TOG will set the bits of items in the list to the same value as at the end of this hypothetical run. As such, during a trusting phase, TOG incurs the same access cost as MTF2. The trusting phase continues until the cost to access a request σ_ℓ, $i < \ell \leq n$, for TOG would cause the total access cost for the phase to become at least m^3 (or the request sequence ends). The next phase, which will be an ignoring phase, begins with request $\sigma_{\ell+1}$.

Ignoring phase: In an ignoring phase, TOG will use the MTF rule to serve the request. Unlike the trusting phase, TOG does not use paid exchanges to match another list configuration. Let σ_i be the first request of some ignoring phase j for $1 \leq i \leq n$ and an even $j \geq 1$. The ignoring phase continues until the cost to access a request σ_ℓ, $i < \ell \leq n$, for TOG would cause the total access cost for the phase to exceed $\beta \cdot m^3$ (or the request sequence ends). The next phase, which will be a trusting phase, begins with request $\sigma_{\ell+1}$.

5.3 Analysis

The access cost in each phase of TOG is $\Theta(m^3)$, while the cost that it might pay at the beginning of each phase is $O(m^2)$. Consequently, the cost of the algorithm in a trusting phase is roughly $m^3 + o(m^3)$ and $\beta m^3 + o(m^3)$ for an ignoring phase. Moreover, the optimal algorithm incurs a cost of at least $m^3/2.5 - m^2$ in a trusting phase and at least $\beta m^3/2 - m^2$ in an ignoring phase; these results follow from the upper bounds of respectively 2.5 and 2 for the competitive ratios of MTF and MTF2, and the fact that the discrepancy in the initial configuration of each phase changes the cost of OPT in that phase by at most m^2. We can extend these results to show that, for sufficiently long lists, the competitive ratio of TOG (regardless of the advice being trusted or not) converges to at most $2 + \frac{2}{4+5\beta}$. We conclude that, when the advice is untrusted, the competitive ratio of TOG is at most $2 + \frac{2}{4+5\beta}$.

Now, assume the advice is trusted. If the advice indicates Timestamp as the best algorithm among MTFE, MTFO, and Timestamp, the algorithm uses Timestamp to serve the entire sequence, and since the advice is right, the competitive ratio will be at most 5/3 [9]. If the advice indicates MTF2 (either MTFE or MTFO), we compare the cost of TOG with that of MTF2 in each phase. A careful phase analysis, similar to the one for the competitive ratio, shows that the ratio between the costs of the two algorithms converges to at most $2 + \frac{2}{4+5\beta}$. We conclude that, when the advice is trusted, the competitive ratio of the TOG algorithm converges to $5/3 + \frac{5\beta}{6+3\beta}$ for sufficiently long lists. We can state the following theorem:

▶ **Theorem 10.** *Algorithm* TOG *with parameter* $\beta \in [0, 1/2]$ *and* k *bits of advice achieves a competitive ratio of at most* $5/3 + \frac{5\beta}{6+3\beta}$ *when the advice is trusted and a competitive ratio of at most* $2 + \frac{2}{4+5\beta}$ *when the advice is untrusted.*

▶ **Corollary 11.** *For list update with untrusted advice, there is a* $(r, f(r))$*-competitive algorithm where* $r \in [5/3, 2]$ *and* $f(r) = 2 + \frac{10-3r}{9r-5}$.

6 Randomized online algorithms with untrusted advice

The discussion in all previous sections pertains to deterministic online algorithms. In this section we focus on randomization and its impact on online computation with untrusted advice. We will assume, as standard in the analysis of randomized algorithms, that the source of randomness is trusted (unlike the advice). Given a randomized algorithm A, its trusted and untrusted competitive ratios are defined as in (1), with the difference that the cost $A(\sigma, \phi)$ is now replaced by the expected cost $\mathbb{E}(A(\sigma, \phi))$.

First, we will argue that randomization can improve the competitiveness of the ski rental problem. For this, we note that [23] gave a randomized algorithm with a single advice bit for this problem which is $\left(\frac{\lambda}{1-e^{-\lambda}}, \frac{1}{1-e^{-(\lambda-1/B)}}\right)$-competitive, where $\lambda \in (1/B, 1)$ is a parameter of the algorithm. For simplicity, we may assume that B is large, hence this algorithm is $\left(\frac{\lambda}{1-e^{-\lambda}}, \frac{1}{1-e^{-\lambda}}\right)$-competitive, which we can write in the equivalent form $(w \ln \frac{w}{w-1}, w)$. In contrast, Theorem 2 shows that any deterministic Pareto-optimal algorithm with advice of any size is $(1+\lambda, 1+1/\lambda)$-competitive, or equivalently $(\frac{w}{w-1}, w)$-competitive. Standard calculus shows that $w \ln \frac{w}{w-1} < \frac{w}{w-1}$; therefore we conclude that the randomized algorithm Pareto-dominates any deterministic algorithm, even when the latter is allowed unbounded advice.

A second issue we address in this section is related to the comparison of random bits and advice bits as resource. More specifically, in the standard model in which advice is always trustworthy, an advice bit can be at least as powerful as a random bit since the former can simulate the efficient choice of the latter, and thus provide a "no-loss" derandomization. However, in the setting of untrusted advice, the interplay between advice and randomization is much more intricate. This is because random bits, unlike advice bits, are assumed to be trusted.

We show, using online bidding as an example, that there are situations in which a deterministic algorithm with $L+1$ advice bits is Pareto-incomparable to a randomized algorithm with 1 random bit and L advice bits. In particular we focus on the *bounded online bidding* problem, in which $u \leq B$, for some given B.

▶ **Theorem 12.** *For every $\epsilon > 0$ there exist sufficiently large B and L such that there is a randomized algorithm for bounded online bidding with L advice bits and 1 random bit, and which is $(\frac{1+\rho_1}{2}\rho_1 + \epsilon, \frac{1+\rho_1}{2\rho_1}w + \epsilon)$-competitive for all $w > 4$, where $\rho_1 = \frac{w-\sqrt{w^2-4w}}{2}$.*

Note that when $B, L \to \infty$, the competitiveness of the best deterministic algorithm with L advice bits approaches the one of X_u^*, as expressed in Theorem 5, namely (ρ_1, w). Thus, Theorem 12 shows that randomization improves upon the deterministic untrusted ratio w by a factor $\frac{1+\rho_1}{2\rho_1} > 1$, at the expense of a degradation of the trusted competitive ratio by a factor $\frac{1+\rho_1}{2} > 1$. For instance, if w is close to 4, then the randomized algorithm has untrusted competitive ratio less than 4, and thus better than any deterministic strategy.

References

1. Susanne Albers. Improved Randomized On-Line Algorithms for the List Update Problem. *SIAM J. Comput.*, 27:682–693, 1998.
2. Spyros Angelopoulos, Christoph Dürr, Shendan Jin, Shahin Kamali, and Marc P. Renault. Online Computation with Untrusted Advice. *CoRR*, abs/1905.05655, 2019. `arXiv:1905.05655`.
3. Spyros Angelopoulos, Christoph Dürr, Shahin Kamali, Marc P. Renault, and Adi Rosén. Online Bin Packing with Advice of Small Size. *Theory Comput. Syst.*, 62(8):2006–2034, 2018.
4. János Balogh, József Békési, György Dósa, Leah Epstein, and Asaf Levin. A New and Improved Algorithm for Online Bin Packing. In *26th Annual European Symposium on Algorithms (ESA 2018)*, pages 5:1–5:14, 2018.
5. János Balogh, József Békési, György Dósa, Leah Epstein, and Asaf Levin. A new lower bound for classic online bin packing. *CoRR*, abs/1807.05554, 2018. `arXiv:1807.05554`.
6. Hans-Joachim Böckenhauer, Dennis Komm, Rastislav Královič, Richard Královič, and Tobias Mömke. On the Advice Complexity of Online Problems. In *Proceedings of the 20th International Symposium on Algorithms and Computation (ISAAC)*, pages 331–340, 2009.
7. Joan Boyar, Lene M. Favrholdt, Christian Kudahl, Kim S. Larsen, and Jesper W. Mikkelsen. Online Algorithms with Advice: A Survey. *SIGACT News*, 47(3):93–129, 2016.

8 Joan Boyar, Lene M Favrholdt, Christian Kudahl, and Jesper W Mikkelsen. Advice complexity for a class of online problems. In *32nd International Symposium on Theoretical Aspects of Computer Science (STACS 2015)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2015.

9 Joan Boyar, Shahin Kamali, Kim S. Larsen, and Alejandro López-Ortiz. On the List Update Problem with Advice. *Information and Computation*, 253:411–423, 2016.

10 Joan Boyar, Shahin Kamali, Kim S. Larsen, and Alejandro López-Ortiz. Online Bin Packing with Advice. *Algorithmica*, 74(1):507–527, 2016.

11 Marek Chrobak and Claire Kenyon. Competitiveness via Doubling. *SIGACT News*, pages 115–126, 2006.

12 Stefan Dobrev, Rastislav Královič, and Dana Pardubská. Measuring the problem-relevant information in input. *RAIRO Inform. Theor. Appl.*, 43(3):585–613, 2009.

13 Yuval Emek, Pierre Fraigniaud, Amos Korman, and Adi Rosén. Online computation with advice. *Theoret. Comput. Sci.*, 412(24):2642–2656, 2011.

14 Sandy Irani. Two results on the list update problem. *Inform. Process. Lett.*, 38:301–306, 1991.

15 David S. Johnson. *Near-optimal bin packing algorithms*. PhD thesis, MIT, Cambridge, MA, 1973.

16 Mikkelsen J.W. Randomization Can Be as Helpful as a Glimpse of the Future in Online Computation. In *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, pages 39:1–39:14, 2016.

17 Shahin Kamali and Alejandro López-Ortiz. Better Compression through Better List Update Algorithms. In *Data Compression Conference*, pages 372–381, 2014.

18 Anna Karlin, Mark Manasse, Larry Rudolph, and Daniel Sleator. Competitive snoopy caching. *Algorithmica*, 3:79–119, 1988.

19 Anna R Karlin, Claire Kenyon, and Dana Randall. Dynamic TCP Acknowledgment and Other Stories about e/(e- 1). *Algorithmica*, 36:209–224, 2003.

20 Alejandro López-Ortiz, Spryos Angelopoulos, and Angèle M. Hamel. Optimal Scheduling of Contract Algorithms for Anytime Problems. *Journal of Artificial Intelligence Research*, 51:533–554, 2014.

21 Thodoris Lykouris and Sergei Vassilvitskii. Competitive Caching with Machine Learned Advice. In *Proceedings of the 35th International Conference on Machine Learning, ICML*, pages 3302–3311, 2018.

22 Adam Meyerson. The Parking Permit Problem. In *46th Annual IEEE Symposium on Foundations of Computer Science*, pages 274–284. IEEE, 2005.

23 Manish Purohit, Zoya Svitkina, and Ravi Kumar. Improving Online Algorithms via ML Predictions. In *Advances in Neural Information Processing Systems*, volume 31, pages 9661–9670, 2018.

24 Marc P. Renault, Adi Rosén, and Rob van Stee. Online algorithms with advice for bin packing and scheduling problems. *Theor. Comput. Sci.*, 600:155–170, 2015. doi:10.1016/j.tcs.2015.07.050.

25 Stuart J. Russell and Shlomo Zilberstein. Composing real-time systems. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 212–217, 1991.

26 Daniel Sleator and Robert E. Tarjan. Amortized efficiency of list update and paging rules. *Commun. ACM*, 28:202–208, 1985.

27 Daniel D. Sleator and Robert Endre Tarjan. Self-adjusting Binary Search Trees. *Jour. of the ACM*, 32:652–686, 1985.

Monochromatic Triangles, Intermediate Matrix Products, and Convolutions

Andrea Lincoln
CSAIL, MIT, Cambridge, MA, USA
andreali@mit.edu

Adam Polak
Institute of Theoretical Computer Science, Faculty of Mathematics and Computer Science, Jagiellonian University, Krakow, Poland
polak@tcs.uj.edu.pl

Virginia Vassilevska Williams
CSAIL, MIT, Cambridge, MA, USA
virgi@mit.edu

Abstract

The most studied linear algebraic operation, matrix multiplication, has surprisingly fast $O(n^\omega)$ time algorithms for $\omega < 2.373$. On the other hand, the $(\min, +)$ matrix product which is at the heart of many fundamental graph problems such as All-Pairs Shortest Paths, has received only minor $n^{o(1)}$ improvements over its brute-force cubic running time and is widely conjectured to require $n^{3-o(1)}$ time. There is a plethora of matrix products and graph problems whose complexity seems to lie in the middle of these two problems. For instance, the Min-Max matrix product, the Minimum Witness matrix product, All-Pairs Shortest Paths in directed unweighted graphs and determining whether an edge-colored graph contains a monochromatic triangle, can all be solved in $\widetilde{O}(n^{(3+\omega)/2})$ time. While slight improvements are sometimes possible using rectangular matrix multiplication, if $\omega = 2$, the best runtimes for these "intermediate" problems are all $\widetilde{O}(n^{2.5})$.

A similar phenomenon occurs for convolution problems. Here, using the FFT, the usual $(+, \times)$-convolution of two n-length sequences can be solved in $O(n \log n)$ time, while the $(\min, +)$ convolution is conjectured to require $n^{2-o(1)}$ time, the brute force running time for convolution problems. There are analogous intermediate problems that can be solved in $O(n^{1.5})$ time, but seemingly not much faster: Min-Max convolution, Minimum Witness convolution, etc.

Can one improve upon the running times for these intermediate problems, in either the matrix product or the convolution world? Or, alternatively, can one relate these problems to each other and to other key problems in a meaningful way?

This paper makes progress on these questions by providing a network of fine-grained reductions. We show for instance that APSP in directed unweighted graphs and Minimum Witness product can be reduced to both the Min-Max product and a variant of the monochromatic triangle problem, so that a significant improvement over $n^{(3+\omega)/2}$ time for any of the latter problems would result in a similar improvement for both of the former problems. We also show that a natural convolution variant of monochromatic triangle is fine-grained equivalent to the famous 3SUM problem. As this variant is solvable in $O(n^{1.5})$ time and 3SUM is in $O(n^2)$ time (and is conjectured to require $n^{2-o(1)}$ time), our result gives the first fine-grained equivalence between natural problems of different running times. We also relate 3SUM to monochromatic triangle, and a coin change problem to monochromatic convolution, and thus to 3SUM.

2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness

Keywords and phrases 3SUM, fine-grained complexity, matrix multiplication, monochromatic triangle

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.53

Funding *Andrea Lincoln*: Partially supported by NSF Grant CCF-1909429.
Adam Polak: Partially supported by the National Science Center, Poland under grants 2017/27/N/ST6/01334 and 2018/28/T/ST6/00305.

Virginia Vassilevska Williams: Supported by an NSF CAREER Award, NSF Grants CCF-1528078, CCF-1514339 and CCF-1909429, a BSF Grant BSF:2012338, a Google Research Fellowship and a Sloan Research Fellowship.

Acknowledgements We would like to thank Amir Abboud for fruitful discussions at an early stage of our research. Part of the research was done when the second author was visiting MIT.

1 Introduction

Matrix multiplication is arguably the most fundamental linear algebraic operation. It is an important primitive for an enormous variety of applications. Within algorithmic research it has a very special role since it is one of the few problems for which we have surprisingly fast and completely counter-intuitive algorithms. Starting with Strassen's breakthrough [36] in 1969, a long line of research culminated in the current bound $\omega < 2.373$ [42, 30], where ω is the smallest real number so that $n \times n$ matrix multiplication can be performed in $\mathcal{O}(n^{\omega+\varepsilon})$ time for all $\varepsilon > 0$.

In many applications, one needs to compute matrix products that are a bit different (often called funny [2]) from the usual definition of matrix multiplication over a ring such as the integers ($C_{ij} = \sum_k A_{ik} \cdot B_{kj}$). Such examples include matrix products over semirings such as the $(\min, +)$-product (often called distance product) which is over the tropical $((\min, +))$ semiring and the Max-Min product which is over the $(\max, \min)$-semiring. Both these products are equivalent to certain types of path optimization problems in graphs. The distance product of $n \times n$ matrices is equivalent to the All-Pairs Shortest Paths (APSP) problem in n-node graphs, so that a $T(n)$ time algorithm for one problem would imply an $\mathcal{O}(T(n))$ time algorithm for the other [21]. Similarly, the Max-Min product is equivalent to the so called All-Pairs Bottleneck Paths (APBP) in graphs (e.g. [35]).

There seems to be a distinct complexity difference between APSP and APBP (and hence the corresponding matrix products), however. The fastest algorithms for APSP and the distance product run in $n^3/\exp(\sqrt{\log n})$ time [46], which is only better by an $n^{o(1)}$ factor than the trivial cubic time algorithm for the distance product. Meanwhile, as was first shown by [38, 39], APBP and the Max-Min product admit a much faster than cubic time algorithm via a reduction to (normal) matrix multiplication; the fastest running time is $\mathcal{O}(n^{(3+\omega)/2})$ [19].

APSP is in fact conjectured to not admit any truly subcubic, $\mathcal{O}(n^{3-\varepsilon})$ time algorithms for $\varepsilon > 0$. Fine-grained complexity has strengthened this hypothesis by providing a large class of problems that are equivalent to APSP and the distance product, via fine-grained subcubic reductions. Thus the reason why distance product is seemingly so difficult is because there are many problems that are equivalent to it and researchers from different communities have all failed to solve these problems faster.

The best known running time for the $n \times n$ Max-Min product, $\widetilde{\mathcal{O}}(n^{(3+\omega)/2})$, while nontrivially subcubic, seems difficult to improve upon. In fact, $\widetilde{\mathcal{O}}(n^{(3+\omega)/2})$ is the best known running time for many other matrix and graph problems besides the Max-Min product: the Dominance product [32] and Equality product [47, 29], All-Pairs Nondecreasing Paths (APNP) and the $(\min, \leqslant)$-product [37, 41, 18]. For some of these problems [49, 24] one can obtain slightly improved running times using rectangular matrix multiplication [23]. However, the closer ω is to 2, the smaller the improvement, and when $\omega = 2$, the $\widetilde{\mathcal{O}}(n^{(3+\omega)/2}) = \widetilde{\mathcal{O}}(n^{2.5})$ running time is the best known for all of these problems. Since their running time exponent is essentially the average of the brute-force exponent 3 and the fast matrix multiplication exponent ω, we will call these problems "intermediate".

There are two additional problems that are intermediate if $\omega = 2$: the Minimum Witness product, which is equivalent to the problem of computing All-Pairs Least Common Ancestors in a DAG, and All-Pairs Shortest Paths (APSP) in unweighted directed graphs. For both problems we know algorithms running in $\widetilde{\mathcal{O}}(n^{(3+\omega)/2}) \leqslant \widetilde{\mathcal{O}}(n^{2.687})$ time [6, 2], and both algorithms can be improved upon, by using rectangular matrix multiplication [15, 50]. The improvement is already seen in a naive implementation, i.e. cutting rectangular matrices into square blocks, which gives an $\widetilde{\mathcal{O}}(n^{2+1/(4-\omega)}) \leqslant \widetilde{\mathcal{O}}(n^{2.615})$ time. Employing a specialized rectangular matrix multiplication algorithm [23], brings the runtime down to $\widetilde{\mathcal{O}}(n^{2.529})$. When $\omega = 2$, however, all the improvements vanish and those running times become $\widetilde{\mathcal{O}}(n^{2.5})$.

Is the 2.5 *running time exponent (for* $\omega = 2$*) for all of these problems a coincidence, or can we relate all of them via fine-grained reductions, and use plausible hypotheses to explain it?*

This is a question that many have asked, but unfortunately there are only two partial answers: First, it is known that Equality product and Dominance product are equivalent ([47, 29], also follows from Proposition 3.4 in [45]), and that they are equivalent to All-Pairs ℓ_{2p+1} Distances [29]. The second result is that the Max-Min product is equivalent to approximate APSP in weighted graphs without scaling [10]. The main question above remains *wide open*.

Parallel to the world of matrix products, there is a very similar landscape of *convolution* problems. While it is well-known that the $(+,\times)$-convolution[1] of two n-length vectors can be computed in $\mathcal{O}(n \log n)$ time using the Fast Fourier Transform (FFT), these techniques no longer work for the $(\min,+)$-convolution, and this problem is conjectured to require $n^{2-o(1)}$ time (see e.g. [14]). Similar to the "intermediate" matrix product problems, there are analogous "intermediate" convolution problems, all in $\widetilde{\mathcal{O}}(n^{3/2})$ time[2]: Max-Min convolution, Dominance convolution, Minimum Witness convolution, etc.

The convolution landscape is even somewhat cleaner than the matrix product one. As the normal convolution $((+,\times))$ is already in (near-)linear time, there are no analogues of rectangular matrix multiplication speedups, and all intermediate problems happen to have exactly the same running time (up to polylogarithmic factors). Still, there is no real formal explanation of why they have the same running time. The only reductions between these convolutions are analogous to the matrix product ones: Dominance convolution is equivalent to Equality convolution [29], and approximate $(\min,+)$-convolution is equivalent to exact Max-Min convolution [10].

1.1 Our contributions

In this paper we provide new fine-grained reductions between several intermediate matrix product and all-pairs graph problems, and between intermediate convolution problems, also relating these to other key problems from fine-grained complexity such as 3SUM. See Figure 1 for a pictorial representation of our results.

Reductions for Graph Problems and Matrix Products

Several of our reductions concern the All-Edges Monochromatic Triangle (AE-MonoΔ) problem: Given an n-node graph in which each edge has a color from 1 to n^2, decide for each edge whether it belongs to a *monochromatic* triangle, a triangle whose all three edges

[1] The $(+,\times)$-convolution of two vectors a and b is the vector c such that $c_i = \sum_j a_j b_{i-j}$.

[2] The exponent $(3+\omega)/2$ for intermediate matrix products is the average of the fast matrix multiplication exponent and the brute force matrix product exponent, and the exponent $3/2$ for intermediate convolution problems is the average of the fast convolution exponent 1 and the brute force exponent 2.

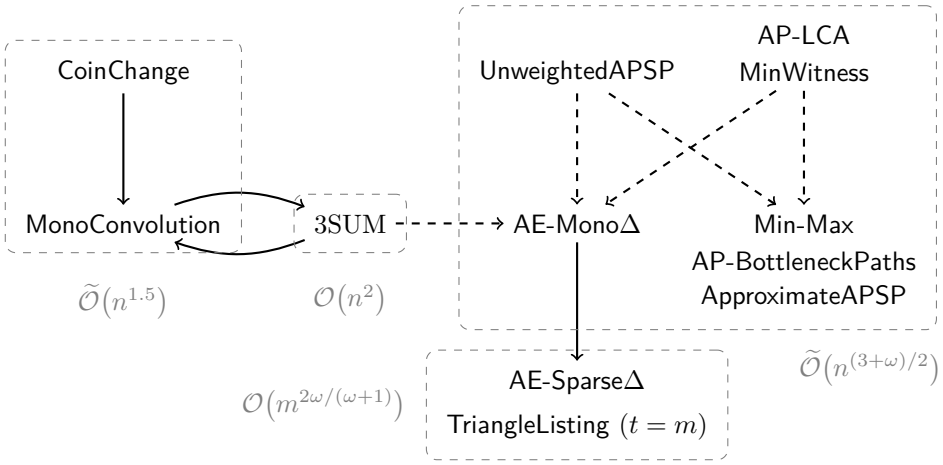


Figure 1 Our results. An arrow pointing from problem A to problem B means that problem A reduces to problem B in the fine-grained sense. Dashed arrows denote reductions which become tight when $\omega = 2$.

have the same color. Vassilevska, Williams and Yuster [40] studied the decision variant of AE-MonoΔ in which one asks whether the given graph contains a monochromatic triangle. They provided an $\mathcal{O}(n^{(3+\omega)/2})$ time algorithm for the decision problem, but that algorithm is in fact strong enough to also solve the all-edges variant AE-MonoΔ, making AE-MonoΔ one of the "intermediate" problems of interest.

To obtain their $\mathcal{O}(n^{(3+\omega)/2})$ time algorithm, Vassilevska, Williams and Yuster [40] implicitly reduce AE-MonoΔ (in a black-box way) to the AE-SparseΔ problem of deciding for every edge e in an m-edge graph whether e is in a triangle. The fastest known algorithm for AE-SparseΔ is by Alon, Yuster and Zwick [3], running in $\mathcal{O}(m^{2\omega/(\omega+1)})$ time, and the problem is known to be runtime equivalent to the problem of *listing* up to m triangles in an m-edge graph [20]. The black-box reduction of [40] from AE-MonoΔ to AE-SparseΔ implies that a significant improvement over the $\mathcal{O}(m^{2\omega/(\omega+1)})$ time for AE-SparseΔ would translate to an improvement over $\mathcal{O}(n^{(3+\omega)/2})$ for AE-MonoΔ.

▶ **Theorem 1** (implicit in [40]). *If* AE-SparseΔ *is in* $\mathcal{O}(m^{2\omega/(\omega+1)-\varepsilon})$ *time, for some* $\varepsilon > 0$, *then* AE-MonoΔ *is in* $\mathcal{O}(n^{(3+\omega)/2-\delta})$ *time, for some* $\delta > 0$.

Our first set of results shows that AE-MonoΔ is powerful enough to capture two well-studied intermediate problems: the Minimum Witness product of two Boolean matrices and the All-Pairs Shortest Paths problem in directed unweighted graphs.

The Minimum Witness product (MinWitness) C of two Boolean matrices A and B is defined as $C_{ij} = \min\{k \mid A_{ik} = B_{kj} = 1\}$ (where the minimum is defined to be ∞ if there is no witness k). It is well-known [15] that MinWitness is equivalent to determining for every pair u, v of vertices in a DAG, the least common ancestor of u and v, i.e. solving the All-Pairs Least Common Ancestors (AP-LCA) problem. The fastest known algorithm for MinWitness runs in $\mathcal{O}(n^{2.529})$ time using rectangular matrix multiplication, and in $\mathcal{O}(n^{2+1/(4-\omega)})$ time just using square matrix multiplication [15].

The All-Pairs Shortest Paths (APSP) problem in unweighted graphs is very well-studied. While in undirected graphs, the problem is known to be solvable in $\widetilde{\mathcal{O}}(n^\omega)$ time [34], the problem in directed graphs is one of our intermediate problems. Its fastest algorithm (similarly to MinWitness) runs in $\mathcal{O}(n^{2.529})$ time using rectangular matrix multiplication, and in $\widetilde{\mathcal{O}}(n^{2+1/(4-\omega)})$ time just using square matrix multiplication [50]. We will refer to the APSP problem in directed unweighted graphs as UnweightedAPSP.

We present reductions from MinWitness and UnweightedAPSP to AE-MonoΔ with only polylogarithmic overhead.

▶ **Theorem 2.** *If* AE-MonoΔ *is in* $T(n)$ *time, then* MinWitness *is in* $\mathcal{O}(T(n)\log n)$ *time.*

▶ **Theorem 3.** *If* AE-MonoΔ *is in* $T(n)$ *time, then* UnweightedAPSP *is in* $\mathcal{O}\big(T(n)\log^2 n\big)$ *time.*

The above reductions tightly relate MinWitness and UnweightedAPSP to AE-MonoΔ if $\omega = 2$, showing that any improvement over the 2.5 exponent for AE-MonoΔ, gives the same improvement for MinWitness and UnweightedAPSP. Due to the tight reduction (Theorem 1) from AE-MonoΔ to AE-SparseΔ, we also obtain that an $\mathcal{O}\big(m^{4/3-\varepsilon}\big)$ time algorithm, with $\varepsilon > 0$, for AE-SparseΔ would give $\mathcal{O}\big(n^{2.5-\delta}\big)$ time algorithms, for $\delta > 0$, for MinWitness and UnweightedAPSP, presenting another tight relationship for the case when $\omega = 2$.

Our next result is that improving over the exponent 2.5 for AE-MonoΔ is at least as hard as obtaining a truly subquadratic time algorithm for the 3SUM problem.

▶ **Theorem 4.** *If* AE-MonoΔ *is in* $\mathcal{O}\big(n^{5/2-\varepsilon}\big)$ *time, then* 3*SUM is in (randomized)* $\widetilde{\mathcal{O}}\Big(n^{2-\frac{4}{5}\varepsilon}\Big)$ *time.*

In 3SUM one is given n integers and is asked whether three of them sum to 0. The problem is easy to solve in $\mathcal{O}\big(n^2\big)$ time, and slightly subquadratic time algorithms exist [4, 11]. 3SUM is a central problem in fine-grained complexity [43]. It is hypothesized to require $n^{2-o(1)}$ time (on a word-RAM with $\mathcal{O}(\log n)$ bit words), and many fine-grained hardness results are conditioned on this hypothesis (see [22, 43]). Our reduction shows that, under the 3SUM Hypothesis, the exponent 2.5 for AE-MonoΔ cannot be beaten, and this is tight if $\omega = 2$. We note that before our work no intermediate matrix, graph, or convolution problem was known to be 3SUM-hard.

Next, we consider the Min-Max product (Min-Max) of two matrices A and B, defined as $C_{ij} = \min_k \max(A_{ik}, B_{kj})$. The Min-Max product is equivalent to the aforementioned Max-Min product (just negate the matrix entries) and the All-Pairs Bottleneck Paths problem, and is thus solvable in $\mathcal{O}\big(n^{(3+\omega)/2}\big)$ time [19].

A very simple folklore reduction shows that Min-Max on $n \times n$ integer matrices is at least as hard as MinWitness on $n \times n$ Boolean matrices, giving a tight relationship when $\omega = 2$.

▶ **Theorem 5** (folklore)**.** *If* Min-Max *is in* $T(n)$ *time, then* MinWitness *is in* $\mathcal{O}(T(n))$ *time.*

Our next result states that All-Pairs Shortest Paths in directed unweighted graphs (UnweightedAPSP) is also tightly reducible to Min-Max. This gives a second intermediate problem that is at least as hard as both MinWitness and UnweightedAPSP.

▶ **Theorem 6.** *If* Min-Max *is in* $T(n)$ *time, then* UnweightedAPSP *is in* $\mathcal{O}(T(n)\log n)$ *time.*

The above theorem also follows from a recent independent result by Barr, Kopelowitz, Porat and Roditty [5]. In particular, they reduce All-Pairs Shortest Paths in directed graphs with edge weights from $\{-1, 0, 1\}$ to Min-Max. Interestingly, they use a substantially different approach than ours. While their argument can be seen as inspired by Seidel's algorithm for unweighted APSP in undirected graphs [34], ours resembles Zwick's algorithm for directed graphs [50].

Reductions for Convolution Problems

Our main result for convolution problems regards the convolution version of AE-MonoΔ, which we call MonoConvolution: Given three integer sequences a, b, c, decide for each index i if there exists j such that $a_j = b_{i-j} = c_i$. We show that MonoConvolution is actually fine-grained equivalent to 3SUM.

▶ **Theorem 7.** *If* MonoConvolution *is in* $\mathcal{O}\left(n^{3/2-\varepsilon}\right)$ *time, then* 3SUM *is in (randomized)* $\widetilde{\mathcal{O}}\left(n^{2-\frac{4}{3}\varepsilon}\right)$ *time.*

▶ **Theorem 8.** *If* 3SUM *is in* $\mathcal{O}\left(n^{2-\varepsilon}\right)$ *time, then* MonoConvolution *is in* $\widetilde{\mathcal{O}}\left(n^{3/2-\varepsilon/4}\right)$ *time.*

This equivalence is arguably the first fine-grained equivalence between natural problems with *different* running time complexities: MonoConvolution is a problem in $\mathcal{O}\left(n^{3/2}\right)$ time, whereas 3SUM is in $\mathcal{O}\left(n^2\right)$ time, and a polynomial improvement on one of these running times would result in a polynomial improvement over the other. All previous fine-grained equivalences were between problems with the same running time exponent: the problems equivalent to APSP [44, 1] are all solvable in $\mathcal{O}\left(N^{1.5}\right)$ time where N is the size of their input, the problems equivalent to Orthogonal Vectors [12] or to $(\min, +)$-convolution [14] are all in quadratic time, the problems equivalent to CNF-SAT [13] are all in $\mathcal{O}(2^n)$ time, etc. While tight fine-grained reductions between problems with different running times are well-known, there was no such equivalence until our result, largely since it often seems difficult to reduce a problem with a smaller asymptotic running time to one with a larger running time, something our Theorem 8 overcomes. Note that the same apparent difficulty is overcome by the reduction from AE-MonoΔ to AE-SparseΔ in Theorem 1, as well as by the reductions from MinWitness and UnweightedAPSP to AE-SparseΔ, which follow from combining Theorems 2 and 3 with Theorem 1.

Theorem 8 together with Theorem 4 give a reduction from MonoConvolution to AE-MonoΔ. Previously reductions from a convolution to the corresponding graph/matrix problem were known only for problems with best known algorithms running in brute-force time, i.e. quadratic time for convolution and cubic time for product, e.g. $(\min, +)$-convolution and $(\min, +)$-product [7].

Finally, we relate MonoConvolution to an unweighted variant of a coin change problem [48, 28] that is related to the minimum word break problem [8]. Given a set of coin values from $\{1, 2, \ldots, n\}$, the CoinChange problem asks to determine for each integer value up to n what is the minimum number of coins (allowing repetitions) that sum to that value. We reduce CoinChange to MonoConvolution with only a polylogarithmic overhead. CoinChange is solvable in $\widetilde{\mathcal{O}}\left(n^{3/2}\right)$ time [9], and our reduction implies that any improvement over the known running times of MonoConvolution or 3SUM would also improve the running time for CoinChange.

▶ **Theorem 9.** *If* MonoConvolution *is in* $T(n)$ *time, then* CoinChange *is in* $\mathcal{O}\left(T(n)\log^2 n\right)$ *time.*

2 Preliminaries

In this section we recall formal definitions of all the problems involved in the reductions presented in the paper. We split these problems by their time complexity.

2.1 Problems in $\widetilde{\mathcal{O}}\left(n^{(3+\omega)/2}\right)$ time

▶ **Definition 10** (All-Edges Monochromatic Triangle, AE-MonoΔ). *Given an n-node graph G in which each edge has a color from 1 to n^2, decide for each edge whether it belongs to a* monochromatic *triangle, a triangle where all three edges have the same color.*

▶ **Definition 11** (Min-Max matrix product, Min-Max). *Given two $n \times n$ matrices A and B, compute matrix C such that*

$$C_{ij} = \min_k \max(A_{ik}, B_{kj}).$$

▶ **Definition 12** (Minimum Witness matrix product, MinWitness). *Given two $n \times n$ Boolean matrices A and B, compute matrix C such that*

$$C_{ij} = \min(\{k \mid A_{ik} = B_{kj} = 1\} \cup \{\infty\}).$$

▶ **Definition 13** (All-Pairs Shortest Paths in directed unweighted graphs, UnweightedAPSP). *Given an n-node unweighted directed graph $G = (V, E)$, compute for each pair of vertices $u, v \in V$ the length of a shortest path from u to v. Note that all path lengths will be in $\{0, 1, \ldots, n-1\} \cup \{\infty\}$.*

2.2 Problems in $\mathcal{O}\left(m^{2\omega/(\omega+1)}\right)$ time

▶ **Definition 14** (All-Edges Sparse Triangle, AE-SparseΔ). *Given an m-edge graph G decide for each edge whether it belongs to a triangle.*

2.3 Problems in $\mathcal{O}(n^2)$ time

▶ **Definition 15** (3SUM). *Given three lists, A, B and C, of n integers, determine if there exist $a \in A$, $b \in B$, and $c \in C$ such that $a + b = c$.*

Let us note that the 3SUM problem is defined in several different ways in literature. They differ as to whether the input is split into three list or all the numbers are in a single list, and whether one looks for $a + b = c$ or $a + b + c = 0$. All these variants are equivalent by simple folklore reductions.

2.4 Problems in $\widetilde{\mathcal{O}}(n^{1.5})$ time

▶ **Definition 16** (MonoConvolution). *Given three sequences a, b, c, all of length n, compute the sequence d such that*

$$d_i = \begin{cases} 1 & \text{if } \exists_j\ a_j = b_{i-j} = c_i, \\ 0 & \text{otherwise.} \end{cases}$$

▶ **Definition 17** (CoinChange). *Given a set of coin values $C \subseteq \{1, 2, \ldots, n\}$, assume you have for each $c \in C$ an infinite supply of coins of value c, and determine for each $v \in \{1, 2, \ldots, n\}$ the minimum number of coins that sums up to v.*

CoinChange can be solved in $\widetilde{\mathcal{O}}(n^{1.5})$ time [9]. The algorithm splits the coins into heavy coins, with weight at least $\sqrt{n}$, and light coins, with weight less than $\sqrt{n}$. The minimum sum for a value can use at most $\sqrt{n}$ heavy coins. By running FFT $\sqrt{n}$ times the algorithm produces a vector with the minimum number of heavy coins needed to sum to every value. That takes $\mathcal{O}(n^{1.5} \log n)$ time in total. Then a classical dynamic programming algorithm is run for the $\sqrt{n}$ light coins and n values, in $\mathcal{O}(n^{1.5})$ time.

2.5 Self-reducibility of 3SUM

In our proofs of Theorems 4 and 7 we use the following fact about 3SUM.

▶ **Lemma 18.** *For any $\alpha \in [0,1]$, a single instance of 3SUM of size n can be reduced to $\mathcal{O}(n^{2\alpha})$ instances of 3SUM of size $\mathcal{O}(n^{1-\alpha})$ each. The reduction runs in time linear in the total size of produced instances, and the original instance is a yes-instance if and only if at least one of the produced instances is a yes-instance.*

This fact appears in many 3SUM-related papers, e.g. [4, 26, 27, 31, 33]. In [4] it was proved using a randomized almost linear hashing scheme [17]. An alternative proof – using a domination argument to provide a deterministic reduction – appeared e.g. in [31, 25], and is based on ideas of [16].

3 Reductions for Graph and Matrix Problems

First, let us recall the algorithm of Vassilevska, Williams and Yuster [40] for AE-MonoΔ. We rephrase the argument so that it not only shows how to solve AE-MonoΔ in $\mathcal{O}(n^{(3+\omega)/2})$ time, but also proves that any polynomial improvement over the $\mathcal{O}(m^{2\omega/(\omega+1)})$ time algorithm of Alon, Yuster and Zwick [3] for AE-SparseΔ translates to a polynomial improvement for AE-MonoΔ.

▶ **Theorem 1** (implicit in [40])**.** *If* AE-SparseΔ *is in $\mathcal{O}(m^{2\omega/(\omega+1)-\varepsilon})$ time, for some $\varepsilon > 0$, then* AE-MonoΔ *is in $\mathcal{O}(n^{(3+\omega)/2-\delta})$ time, for some $\delta > 0$.*

Proof. Assume AE-SparseΔ is in $\mathcal{O}(m^\alpha)$ time. Take an AE-MonoΔ instance. For each color consider the subgraph composed of all the edges of that color. Each such subgraph constitutes an independent instance of AE-SparseΔ. However, simply using the $\mathcal{O}(m^\alpha)$ time algorithm on all of these instances is not efficient enough. Intuitively, some of the instances might be too dense.

Instead, for a parameter t to be determined later, take the t largest subgraphs (in terms of the number of edges). For each of them solve the problem by using fast matrix multiplication to compute the square of the adjacency matrix. This takes $\mathcal{O}(tn^\omega)$ time in total. Let m_i denote the number of edges in the i-th of the remaining subgraphs. Clearly, $\forall_i\ m_i \leqslant n^2/t$, and $\sum_i m_i \leqslant n^2$. On each of those subgraphs use the $\mathcal{O}(m^\alpha)$ time AE-SparseΔ algorithm. This takes an order of

$$\sum_i m_i^\alpha = \sum_i m_i \cdot m_i^{\alpha-1} \leqslant \sum_i m_i \cdot (n^2/t)^{\alpha-1} \leqslant n^2 \cdot (n^2/t)^{\alpha-1}$$

time. The total runtime is thus $\mathcal{O}(tn^\omega + n^{2\alpha}/t^{\alpha-1})$. Optimize by setting $t = n^{(2\alpha-\omega)/\alpha}$, and get an $\mathcal{O}(n^{\omega+2-(\omega/\alpha)})$ time.

Observe that for $\alpha = 2\omega/(\omega+1)$ the runtime is $\mathcal{O}(n^{(3+\omega)/2})$. Moreover, for $\alpha < 2\omega/(\omega+1)$ the exponent in the runtime becomes strictly smaller. ◀

Now, we proceed to show how to use AE-MonoΔ to solve two popular graph problems. We start with the problem of finding All-Pairs Least Common Ancestors in directed acyclic graphs, which is runtime-equivalent to MinWitness [15]. We reduce a single instance of MinWitness to $\log n$ instances of AE-MonoΔ.

▶ **Theorem 2.** *If* AE-MonoΔ *is in $T(n)$ time, then* MinWitness *is in $\mathcal{O}(T(n)\log n)$ time.*

Proof. The main idea is to use a parallel binary search. For each entry of the output matrix C we will keep an interval which that entry is guaranteed to lie in. With a single call to AE-MonoΔ we will be able to halve all the intervals.

W.l.o.g. assume the last column of A and last row of B are all ones, so that the output is always finite. For $\ell \in [\log n]$, let $C^{(\ell)}$ denote the matrix pointing to 2^ℓ-length intervals in which entries of C lie, that is $C^{(\ell)}_{ij}$ is the unique integer such that $C_{ij} \in \left[2^\ell \cdot C^{(\ell)}_{ij}, 2^\ell \cdot (C^{(\ell)}_{ij}+1)\right)$.

We will compute $C^{(\ell)}$ for $\ell = \lceil \log n \rceil, \ldots, 1, 0$. Observe that $C^{(\lceil \log n \rceil)}$ is the zero matrix. Knowing $C^{(\ell+1)}$, we compute $C^{(\ell)}$ as follows. We create a tripartite graph $G = (I \cup J \cup K, E)$, with each of I, J, K containing n vertices. We add edges between I and K according to the matrix A. Edges from the k-th column get the label $\lfloor k/2^\ell \rfloor$. We add edges between K and J according to the matrix B. Edges from the k-th row get the label $\lfloor k/2^\ell \rfloor$. Finally, we add the full bipartite clique between I and J. The edge between the i-th vertex of I and the j-th vertex of J gets the label $2 \cdot C^{(\ell+1)}$. That edge forms a monochromatic triangle if and only if $C_{ij} \in \left[2^\ell \cdot 2 \cdot C^{(\ell+1)}_{ij}, 2^\ell \cdot (2 \cdot C^{(\ell+1)}_{ij} + 1)\right)$, i.e. $C^{(\ell)}_{ij} = 2 \cdot C^{(\ell+1)}_{ij}$. Otherwise, it must be that $C_{ij} \in \left[2^\ell \cdot (2 \cdot C^{(\ell+1)}_{ij} + 1), 2^\ell \cdot (2 \cdot C^{(\ell+1)}_{ij} + 2)\right)$, i.e. $C^{(\ell)}_{ij} = 2 \cdot C^{(\ell+1)}_{ij} + 1$. Therefore, solving AE-MonoΔ on G suffices to compute $C^{(\ell)}$. Finally, observe that $C = C^{(0)}$. ◀

With a slightly more involved argument we show how to solve UnweightedAPSP with $\mathcal{O}(\log^2 n)$ calls to AE-MonoΔ.

▶ **Theorem 3.** *If* AE-MonoΔ *is in* $T(n)$ *time, then* UnweightedAPSP *is in* $\mathcal{O}(T(n) \log^2 n)$ *time.*

Proof. We solve UnweightedAPSP in $\log n$ rounds, in the i-th round we compute matrix $D^{\leqslant 2^i}$ of lengths of shortest paths of length up to 2^i (other entries equal to ∞). Each round will consist of a parallel binary search, similar to the one we use in our reduction from MinWitness to AE-MonoΔ (Theorem 2). The algorithm is based on the fact that in unweighted graphs every path can be split roughly in half, i.e. if the distance from u to v equals to k, then there must exist a vertex w such that the distances from u to w and from w to v equal to $\lfloor k/2 \rfloor + \{0, 1\}$.

To start, note that $D^{\leqslant 2^0}$ is a $\{0, 1, \infty\}$-matrix that can be easily obtained from the adjacency matrix of the input graph. Now, assume we already computed $D^{\leqslant 2^i}$ and let us proceed to compute $D^{\leqslant 2^{i+1}}$. To avoid excessive indexing, let A denote $D^{\leqslant 2^i}$, and B denote $D^{\leqslant 2^{i+1}}$. For each entry of the output matrix B we will keep an interval which that entry is guaranteed to lie in. With a single call to AE-MonoΔ we will be able to halve all the intervals.

For $\ell \in \{0, 1, \ldots, i+2\}$, let $B^{(\ell)}$ denote the matrix pointing to 2^ℓ-length intervals in which entries of B lie, that is $B^{(\ell)}_{uv}$ equals to the unique integer such that $B_{uv} \in \left[2^\ell \cdot B^{(\ell)}_{uv}, 2^\ell \cdot (B^{(\ell)}_{uv} + 1) - 1\right]$, or to infinity in case B_{uv} is infinite.

We will iterate over ℓ from $i+2$ down to 0. First, we need to compute $B^{(i+2)}$, whose entries are either zeros or infinities. Recall that we already know the matrix $A = D^{\leqslant 2^i}$. Consider a pair of nodes u and v that are at distance at most 2^{i+1}. There must exist a node w such that $A_{uw} \leqslant 2^i$ and $A_{wv} \leqslant 2^i$, that is, equivalently both A_{uw} and A_{wv} are finite. We obtain the matrix $B^{(i+2)}$ by squaring the $(0, 1)$ matrix obtained from A by putting ones at the finite entries and zeros elsewhere. That single matrix multiplication can be easily simulated by a single call to AE-MonoΔ, using just two colors.

Once we have the matrix $B^{(\ell+1)}$ we want to compute $B^{(\ell)}$. For this we first note that if $B^{(\ell+1)}_{uv} = j$ then $B^{(\ell)}_{uv}$ is either $2j$ or $2j+1$. If $B^{(\ell)}_{uv} = 2j$, then there must exist a vertex w such that

$$A_{uw} \in \big[2^{\ell-1}\cdot(2j), 2^{\ell-1}\cdot(2j+1)\big), \quad \text{and} \quad A_{wv} \in \big[2^{\ell-1}\cdot(2j), 2^{\ell-1}\cdot(2j+1)\big]. \tag{1}$$

Furthermore, if $B^{(\ell)}_{uv} > 2j$, then there is no w such that the above condition holds. This will allow us to distinguish between the $2j$ and $2j+1$ cases by coloring the matrix A based on which range the entries fall in. Note that the ranges in Condition (1) do not overlap with corresponding ranges for different integer values $j' \neq j$. Thus we will be able to use a single call to AE-MonoΔ to check in parallel for all values of $B^{(\ell)}_{uv}$ if they are the smaller even value $2\cdot B^{(\ell+1)}_{uv}$ or the larger odd value $2\cdot B^{(\ell+1)}_{uv}+1$.

We construct an AE-MonoΔ instance with a tripartite graph with the vertex set $U \sqcup V \sqcup W$ where U, V and W are disjoint copies of the original vertex set. The edges between U and V correspond to our desired output. If $B^{(\ell+1)}_{uv} = j$ then we color the edge $(u,v) \in U \times V$ with j. The edges between U and W correspond to the first part of Condition (1), i.e. if $A_{uw} \in \big[2^{\ell-1}\cdot(2j), 2^{\ell-1}\cdot(2j+1)\big)$, then we add the edge (u,w) to $U \times W$ with color j. The edges between W and V correspond to the second part of Condition (1), i.e. if $A_{wv} \in \big[2^{\ell-1}\cdot(2j), 2^{\ell-1}\cdot(2j+1)\big]$, then we add the edge (w,v) to $W \times V$ with color j. Any edge (u,v) in $U \times V$ that is in a monochromatic triangle implies $B^{(\ell)}_{uv} = 2\cdot B^{(\ell+1)}_{uv}$. Conversely, any edge (u,v) that is not a part of any monochromatic triangle implies $B^{(\ell)}_{uv} = 2\cdot B^{(\ell+1)}_{uv}+1$.

We iterate down until $B^{(0)}$, and observe that $B^{(0)} = B$. Thus, with $\mathcal{O}(\log n)$ calls we can compute $B = D^{\leqslant 2^{i+1}}$ from $A = D^{\leqslant 2^i}$. To solve UnweightedAPSP the total number of calls we need to make to AE-MonoΔ is $\mathcal{O}\big(\log^2(n)\big)$. Therefore, if AE-Mono$\Delta$ can be solved in $T(n)$ time then UnweightedAPSP can be solved in $\mathcal{O}\big(T(n)\log^2(n)\big)$ time. ◀

Now we show that AE-MonoΔ is 3SUM-hard. In our proof we use as a black-box the following reduction from 3SUM to AE-SparseΔ.

▶ **Lemma 19** (Kopelowitz, Pettie, Porat [27]). *A single instance of 3SUM of size n can be reduced to a single instance of* AE-SparseΔ *with $\Theta(n\log n)$ vertices and $\Theta\big(n^{3/2}\log n\big)$ edges.*

▶ **Theorem 4.** *If* AE-MonoΔ *is in $\mathcal{O}\big(n^{5/2-\varepsilon}\big)$ time, then 3SUM is in (randomized) $\widetilde{\mathcal{O}}\Big(n^{2-\frac{4}{5}\varepsilon}\Big)$ time.*

Proof. Given an instance of 3SUM of size N, we use the self-reduction (Lemma 18), and reduce it to $\mathcal{O}\big(N^{2/5}\big)$ instances of size $\mathcal{O}\big(N^{4/5}\big)$ each. Then, we reduce each of these instances to an AE-SparseΔ instance with $n = \Theta\big(N^{4/5}\log N\big)$ vertices and $m = \Theta\big(N^{6/5}\log N\big)$ edges, using Lemma 19. Now we will show how to combine these $\mathcal{O}\big(N^{2/5}\big)$ AE-SparseΔ instances to form polylogarithmically many AE-MonoΔ instances, each with $\mathcal{O}\big(N^{4/5}\log N\big)$ vertices, which will finish the proof.

Assume w.l.o.g. that all the created graphs are over the same vertex set $[n]$. If we were lucky enough and the edge sets of the created AE-SparseΔ instances were disjoint, the reduction would be essentially done. Indeed, we could simply union the edge sets to create a single graph, and use colors to track from which graph every edge originates. Solving that one AE-MonoΔ instance would provide answers to all AE-SparseΔ instances. Sadly, the chances of such a favorable collision-free scenario are very slim. The remaining part of the proof shows how to deal with multiple AE-SparseΔ instances containing the same edge.

We randomly permute the vertex sets, for each graph independently. For a fixed $(u,v) \in [n]^2$, the probability that a fixed graph contains the edge (u,v) equals to $p = m/\binom{n}{2} = \mathcal{O}\big((N^{2/5}\log N)^{-1}\big)$. The expected number of (u,v) edges across all graphs is $\mathcal{O}\big(N^{2/5}\cdot p\big) = \mathcal{O}(1/\log N)$. By a Chernoff bound, the probability that the number of (u,v) edges exceeds

$c \log n$ is less than $e^{\Theta(c \log n)}$. We take c large enough so that, by union bound, with probability at least $1/2$ no edge appears more than $c \log n$ times across all graphs. For each $(u, v) \in [n]^2$ we arbitrarily number all (u, v) edges with consecutive positive integers from 1 up to at most $c \log n$. We iterate over all triples $(i, j, k) \in [c \log n]^3$. For every triple we create a tripartite graph with the vertex set $V_1 \sqcup V_2 \sqcup V_3$, for $V_1 = V_2 = V_3 = [n]$. We create an edge (u, v) between V_1 and V_2 if there exists an edge (u, v) with number i assigned to it in any of the AE-SparseΔ instances. Note that there can be at most one such instance. We set the color of the newly created edge to the identifier of the instance it originates from. Similarly, we create edges between V_2 and V_3 using edges with number j assigned, and between V_3 and V_1 using number k. That gives us $(c \log n)^3$ instances of AE-MonoΔ. Note that every triangle present in any of the AE-SparseΔ instance corresponds to a single monochromatic in one of the AE-MonoΔ instances, and vice versa. We solve all AE-MonoΔ instances and combine the outputs in order to get the output for all AE-SparseΔ instances, and eventually for the 3SUM instance. ◀

The next two theorems use techniques similar to Theorems 2 and 3 to give reductions to Min-Max.

▶ **Theorem 5** (folklore). *If* Min-Max *is in* $T(n)$ *time, then* MinWitness *is in* $\mathcal{O}(T(n))$ *time.*

Proof. Given two $(0, 1)$ matrices A and B, we construct matrices A' and B' such that

$$A'_{ik} = \begin{cases} k & \text{if } A_{ik} = 1, \\ \infty & \text{if } A_{ik} = 0, \end{cases} \quad \text{and} \quad B'_{kj} = \begin{cases} k & \text{if } B_{kj} = 1, \\ \infty & \text{if } B_{kj} = 0. \end{cases}$$

Observe that the $(\min, \max)$-product of A' and B' equals to the minimum witness product of A and B. ◀

▶ **Theorem 6.** *If* Min-Max *is in* $T(n)$ *time, then* UnweightedAPSP *is in* $\mathcal{O}(T(n) \log n)$ *time.*

Proof. The reduction is similar to the reduction from UnweightedAPSP to AE-MonoΔ (Theorem 3) in that we also have $\log n$ rounds, and in the i-th round we compute matrix $D^{\leqslant 2^i}$ of lengths of shortest paths of length up to 2^i (other entries equal to ∞). The key difference is that, in each round, instead of performing a binary search and issuing $\log n$ calls to AE-MonoΔ, we issue just two calls to Min-Max.

As before, first note that $D^{\leqslant 2^0}$ is a $\{0, 1, \infty\}$-matrix that can be easily obtained from the adjacency matrix of the input graph. Now, assume we already computed $D^{\leqslant 2^i}$ and let us proceed to compute $D^{\leqslant 2^{i+1}}$. Let $\ell = 2^i$. Naturally, $D^{\leqslant 2\ell}$ is the $(\min, +)$-product of $D^{\leqslant \ell}$ with itself, but this sole observation is not enough for our purposes. We will exploit the fact that $D^{\leqslant \ell}$ is not an arbitrary matrix – but a (truncated) matrix of shortest paths in an unweighted graph – in order to compute that specific $(\min, +)$-product using a Min-Max algorithm. Let $A \mathbin{⍉} B$ denote the $(\min, \max)$-product of matrices A and B.

First, we handle even-length paths. We compute $E = 2 \cdot (D^{\leqslant \ell} \mathbin{⍉} D^{\leqslant \ell})$. Note that $D^{\leqslant 2\ell}_{uv} \leqslant E_{uv}$ for all $u, v \in V$, because for any two integers a, b we have $a + b \leqslant 2 \cdot \max(a, b)$. Moreover, if $D^{\leqslant 2\ell}_{uv} = 2k$, then there must exist $w \in V$ such that $D^{\leqslant \ell}_{uw} = D^{\leqslant \ell}_{wv} = k$, and thus $D^{\leqslant \ell}_{uw} + D^{\leqslant \ell}_{wv} = 2 \cdot \max(D^{\leqslant \ell}_{uw}, D^{\leqslant \ell}_{wv})$ and $D^{\leqslant 2\ell}_{uv} = E_{uv}$.

For odd-length paths we proceed in a similar manner, just the formulas become slightly more obscure. We compute $O = 2 \cdot (D^{\leqslant \ell} \mathbin{⍉} (D^{\leqslant \ell} - 1)) + 1$. Note that $D^{\leqslant 2\ell}_{uv} \leqslant O_{uv}$ for all $u, v \in V$, because for any two integers a, b we have $a + b \leqslant 2 \cdot \max(a, b - 1) + 1$. Moreover, if $D^{\leqslant 2\ell}_{uv} = 2k + 1$, then there must exist $w \in V$ such that $D^{\leqslant \ell}_{uw} = k$ and $D^{\leqslant \ell}_{wv} = k + 1$, and thus $D^{\leqslant \ell}_{uw} + D^{\leqslant \ell}_{wv} = 2 \cdot \max(D^{\leqslant \ell}_{uw}, D^{\leqslant \ell}_{wv} - 1) + 1$ and $D^{\leqslant 2\ell}_{uv} = O_{uv}$.

Consequently, we compute $D^{\leqslant 2\ell}_{uv} = \min(E_{uv}, O_{uv})$, for all $u, v \in V$. ◀

4 Reductions for Convolution Problems

In this section we provide two reductions which together show that MonoConvolution is fine-grained equivalent to 3SUM. Recall that the best known algorithms for MonoConvolution require time $n^{3/2-o(1)}$, and the best algorithms for 3SUM require time $n^{2-o(1)}$, so this is an equivalence between problems of different time complexity. At the end of the section we reduce CoinChange to MonoConvolution.

First, let us recall the All-Integers variant of 3SUM, which parallels the All-Edges variants of our graph problems. That variant is easier to work with than the original 3SUM problem for our purposes. Luckily if either variant has a subquadratic algorithm then they both do [44].

▶ **Definition 20** (All-Integers 3SUM). *Given three lists A, B, C of n integers each, output the list of all integers $c \in C$ such that there exist $a \in A$ and $b \in B$ such that $a + b = c$.*

▶ **Lemma 21** (Vassilevska Williams, Williams [44]). *If 3SUM is in $\mathcal{O}(n^{2-\varepsilon})$ time, then All-Integers 3SUM is in $\mathcal{O}(n^{2-\varepsilon/2})$ time.*

An important ingredient of our reduction from 3SUM to AE-MonoΔ (Theorem 7) is the following range reduction for 3SUM.

▶ **Lemma 22** (Baran, Demaine, Pătraşcu, rephrased, see Section 2.1 of [4]). *For every positive integer output size s, there exists a family of hash functions H such that:*

1. *Every hash function $h \in H$ hashes to the range $\{0, 1, \ldots, R-1\}$ for $R = 2^s$.*
2. *For all integers $a, b, c \in \mathbb{Z}$ and all hash functions $h \in H$, if $a + b = c$, then*

$$h(a) + h(b) \equiv h(c) + \{-1, 0, 1\} \mod R.$$

3. *Given an integer c and two lists of n integers A and B such that there are no $a \in A, b \in B$ with $a + b = c$, the probability, over hash functions h drawn uniformly at random from H, that there exist $a \in A, b \in B$ such that $h(a) + h(b) \equiv h(c) + \{-1, 0, 1\} \mod R$ is at most $\mathcal{O}(n^2/R)$.*

We are now ready to show that 3SUM can be solved efficiently with a MonoConvolution algorithm. Our reduction uses the fact that we can re-write a 3SUM instance with n integers in $\{-R, \ldots, R\}$ as a convolution of $\mathcal{O}(R)$-length $(0, 1)$ vectors, where a one in the i-th position corresponds to the number i in the original 3SUM instance. We will combine several such instances into one MonoConvolution instance by giving each instance its own number. A one in position i in a convolution instance labelled j will result in the MonoConvolution instance having j in position i.

▶ **Theorem 7.** *If MonoConvolution is in $\mathcal{O}(n^{3/2-\varepsilon})$ time, then 3SUM is in (randomized) $\widetilde{\mathcal{O}}\left(n^{2-\frac{4}{3}\varepsilon}\right)$ time.*

Proof. Given an instance of 3SUM of size n, we reduce it to $\mathcal{O}(n^{2/3})$ instances of size $\mathcal{O}(n^{2/3})$ each, using the self-reduction (Lemma 18). Although for the self-reduction itself it would be sufficient just to solve 3SUM on each of these instances – i.e. decide if there exist a, b, c with $a + b = c$ – we are going to solve the All-Integers 3SUM variant – i.e. decide for each c if there exist a and b with $a + b = c$.

To each created instance we apply a hashing scheme of Lemma 22 in order to reduce the universe size down to $R = n^{4/3}$. This introduces false positives for each element with probability $\mathcal{O}((n^{2/3})^2)/R = \mathcal{O}(1)$. Note that the hashing has one-sided error, i.e. if for some

element c there are no a and b such that $h(a)+h(b) \equiv h(c)+\{-1,0,1\} \mod R$, then with certainty there are no a and b such that $a+b=c$. To mitigate the effect of false positives we create $\mathcal{O}(\log n)$ copies of each instance, each copy with an independently drawn hash function. Note that for every fixed element c, if there are no a, b with $a+b=c$, then the probability that in each of the independent $\mathcal{O}(\log n)$ copies we detect that $h(a)+h(b) \equiv h(c)+\{-1,0,1\} \mod R$ for some $h(a), h(b)$ is $1/\text{poly}(n)$, and we can make the degree of the polynomial arbitrarily large by choosing an appropriate multiplicative constant for the number of copies. Therefore we can use the union bound to argue that with at least $2/3$ probability there are no false positives across all instances and all elements.

Suppose that for some subinstance A, B, C of size $n^{2/3}$ we obtained for every one of the $\mathcal{O}(\log n)$ hashed instance copies, for every $t \in [R]$ for which there is some c with $h(c)=t$, whether there are some $h(a), h(b)$ with $h(a)+h(b) \equiv h(c)+\{-1,0,1\} \mod R$. Then, we can go through every $c \in C$ and if for every copy the answer for $h(c)$ was YES, we can conclude that (whp) $\exists a \in A, b \in B$ with $a+b=c$, and if the answer was NO at least once, then we can conclude that there is no pair that sums to c.

Here an important point is that we need to solve *all* of the $\mathcal{O}\left(n^{2/3}\log n\right)$ instances of All-Integers 3SUM above on $n^{2/3}$ integers each over a range $[\mathcal{O}\left(n^{4/3}\right)]$. We will embed solving all instances simultaneously into solving a small number of MonoConvolution instances.

Each of the above $\mathcal{O}\left(n^{2/3}\log n\right)$ instances of All-Integers 3SUM easily reduces to an (OR, AND)-convolution of $(0,1)$ vectors of length $\mathcal{O}\left(n^{4/3}\right)$, each with only $\mathcal{O}\left(n^{2/3}\right)$ nonzero entries, and with only $\mathcal{O}\left(n^{2/3}\right)$ relevant output coordinates one needs to compute. If only we had no collisions – i.e. two instances with the same nonzero input coordinate or the same relevant output coordinate – we could easily combine all the convolution instances into a single instance of MonoConvolution, with $\mathcal{O}\left(n^{2/3}\log n\right)$ different colors/values. However, the collisions are unavoidable. In order to circumvent these collisions, we will add small random shifts, and use a similar analysis as in the 3SUM to AE-MonoΔ reduction of Theorem 4.

Specifically, for each instance we chose a shift s uniformly at random from a range of size $\mathcal{O}\left(n^{4/3}\right)$, we add s to all elements in A, add s to all elements in B, and add $-2s$ to all elements in C. Let the numbers after the shift lie in $\{-R', \ldots, R'\}$ where $R' = \mathcal{O}\left(n^{4/3}\right)$.

These shifts do not change whether for a given triplet a, b, c the condition $a+b=c$ holds or not. For a fixed value $v \in \{-R', \ldots, R'\}$ the expected number of instances containing v is $\mathcal{O}(\log n)$: for each particular instance, the probability that one of its numbers lands at v after the shift is $\mathcal{O}\left(n^{2/3}/R'\right) = \mathcal{O}\left(1/n^{2/3}\right)$; then summing over all the instances gives an expectation of $\mathcal{O}(\log n)$.

Since the shifts are independent, we can use a Chernoff bound to bound the probability that the number of instances containing v exceeds $c\log n$ by $\leqslant e^{\Theta(c\log n)}$. We take c large enough so that, by union bound, the probability that no value is contained in more than $c\log n$ instances is at least $2/3$.

Then, once again following the example of Theorem 4, we reduce the problem to $(c\log n)^3$ instances of MonoConvolution as follows.

For each value $r \in \{-R', \ldots, R'\}$, let the instances that contain r in their A sets be $in_A(r)[1], \ldots, in_A(r)[c\log n]$. Define $in_B(r)[1], \ldots, in_B(r)[c\log n]$ and $in_C(r)[1], \ldots, in_C(r)[c\log n]$ analogously.

We now create an instance of MonoConvolution for each choice of $(x,y,z) \in [c\log n]^3$. In instance (x,y,z) we create vectors a, b, c, where for each $r \in \{-R', \ldots, R'\}$, we set $a_r = in_A(r)[x]$, $b_r = in_B(r)[y]$ and $c_r = in_C(r)[z]$.

Then for any instance i that contains r in A, s in B and t in C, we would have $in_A(r)[x] = in_B(s)[y] = in_C(t)[z] = i$ for some x, y, z and so we will place i in a_r, b_s, and c_t for that choice of x, y, z. Thus all value collisions will be handled. ◀

This next reduction finishes the equivalence between MonoConvolution and 3SUM. It uses a high-frequency/low-frequency split. For elements that appear at a high frequency we use FFT. For elements of low frequency we make calls to All-Integers 3SUM. Recall that a subquadratic algorithm for 3SUM implies a subquadratic algorithm for All-Integers 3SUM.

▶ **Theorem 8.** *If 3SUM is in $\mathcal{O}(n^{2-\varepsilon})$ time, then* MonoConvolution *is in $\widetilde{\mathcal{O}}(n^{3/2-\varepsilon/4})$ time.*

Proof. For a parameter t to be determined later, consider the t most frequent values. For each of these values use FFT to calculate the standard $(+,\times)$-convolution of two $(0,1)$ vectors formed from vectors a and b by putting ones everywhere that value appears, and zeros everywhere else. Examine, where the outputs of these convolutions are nonzero, in order to determine the part of output to MonoConvolution corresponding to occurrences of the frequent values in vector c. This takes $\widetilde{\mathcal{O}}(tn)$ time in total.

Let n_i denote the number of occurrences of the i-th of the remaining values in all three sequences. Clearly, $\forall_i\ n_i \leqslant 3n/t$, and $\sum_i n_i \leqslant 3n$. For each value v out of those remaining values construct sets of indices at which it appears in vectors a, b, c, i.e. $A = \{j : a_j = v\}$, $B = \{j : b_j = v\}$, $C = \{j : c_j = v\}$, and solve All-Integers 3SUM on these sets. For each element j reported by the All-Integers 3SUM algorithm assign the corresponding output of MonoConvolution $d_j = 1$. By Lemma 21, solving these All-Integers 3SUM instances takes an order of

$$\sum_i n_i^{2-\varepsilon/2} = \sum_i n_i \cdot n_i^{1-\varepsilon/2} \leqslant \sum_i n_i \cdot (3n/t)^{1-\varepsilon/2} \leqslant 3n \cdot (3n/t)^{1-\varepsilon/2}$$

time. The total time is thus $\widetilde{\mathcal{O}}\big(tn + n \cdot (n/t)^{1-\varepsilon/2}\big)$. Optimize by setting $t = n^{1/2-\varepsilon/4}$, and get the desired runtime. ◀

The following theorem connects the CoinChange problem to our network of reduction. The proof uses the same structure and techniques as the reduction from UnweightedAPSP to AE-MonoΔ in Theorem 3.

▶ **Theorem 9.** *If* MonoConvolution *is in $T(n)$ time, then* CoinChange *is in $\mathcal{O}\big(T(n)\log^2 n\big)$ time.*

Proof. Let S denote the array of output values, i.e. $S[v]$ equals to the minimum number of coins that sum to v. Parallel to the proof of Theorem 3, let $S^{\leqslant 2^i}[v]$ be infinity if $S[v] > 2^i$, and otherwise equal to $S[v]$. We solve CoinChange in $\log n$ rounds, in the i-th round we compute $S^{\leqslant 2^i}$.

Note that $S^{\leqslant 2^0}[0] = 0$, and, for $v \geqslant 1$, $S^{\leqslant 2^0}[v] = 1$ if $v \in C$ and $S^{\leqslant 2^0}[v] = \infty$ otherwise. Further note that $S = S^{\leqslant 2^{\log n}}$. We will show how to compute $S^{\leqslant 2^{i+1}}$ given $S^{\leqslant 2^i}$. We will then iterate i from 0 up to $\log n$.

Following the style of Theorem 3, we avoid overparameterizing by setting $A = S^{\leqslant 2^i}$ and $B = S^{\leqslant 2^{i+1}}$. Let $B^{(\ell)}$ be an array pointing to 2^ℓ-length intervals in which entries of B lie, i.e. $B^{(\ell)}[v] = j$ if $B[v] \in [2^\ell j, 2^\ell(j+1) - 1]$. We will iterate ℓ from $i+2$ down to 0 to compute B from A.

First we show how to compute $B^{(i+2)}$ from A. If there is a way to sum to v with at most 2^{i+1} coins, then there must be a $u \in [0, n]$ such that both $A[u]$ and $A[v-u]$ are at most 2^i. Conversely, if there is no way to sum to v with at most 2^i coins, then there will be no u that meets the above criteria. Therefore we create a $(0,1)$ vector a with $a_v = 1$ if and only if $A[v]$ is finite. Then, we compute the $(+,\times)$-convolution of a with itself, in near-linear time using FFT. We set $B[v] = 0$ where the convolution output is non-zero and $B[v] = \infty$ everywhere else.

Now we show how to compute $B^{(\ell)}$ from A and $B^{(\ell+1)}$. Note that if $B^{(\ell+1)}[v] = j$ then $B^{(\ell)}[v] \in \{2j, 2j+1\}$. Next, note that if $B^{(\ell)}[v] = 2j$, then there must exist an integer $u \in [0, n]$ such that

$$A[u] \in \big[2^{\ell-1} \cdot (2j), 2^{\ell-1} \cdot (2j+1)\big), \quad \text{and} \quad A[v-u] \in \big[2^{\ell-1} \cdot (2j), 2^{\ell-1} \cdot (2j+1)\big]. \quad (2)$$

Furthermore, if $B^{(\ell)}[v] > 2j$, then there is no u such that the above condition holds. This will allow us to distinguish between the $2j$ and $2j+1$ cases. Note that the ranges in Condition (2) do not overlap with corresponding ranges for different integer values $j' \neq j$. Thus, we will be able to use a single call to MonoConvolution to check in parallel for all values v if $B^{(\ell)}[v]$ is the smaller even value $2B^{(\ell+1)}[v]$ or the larger odd value $2B^{(\ell+1)}[v] + 1$.

We construct a MonoConvolution instance with three input vectors a, b, c. The first input vector corresponds to the first part of Condition (2), i.e. if $A[v] \in \big[2^{\ell-1} \cdot (2j), 2^{\ell-1} \cdot (2j+1)\big)$, then $a_v = j$. Any entries a_v unset by this condition are given the special value $a_v = -1$. The second vector corresponds to the second part of Condition (2), i.e. if $A[v] \in \big[2^{\ell-1} \cdot (2j), 2^{\ell-1} \cdot (2j+1)\big]$, then $b_v = j$. Similarly, any entries b_v unset by this condition are given the special value $b_v = -1$. The last vector corresponds to our desired output, i.e. $c_v = B^{(\ell+1)}[v]$. Let d denote the vector output by this MonoConvolution call. Now, if $d_v = 1$ then $B^{(\ell)}[v] = 2B^{(\ell+1)}[v]$, else $B^{(\ell)}[v] = 2B^{(\ell+1)}[v] + 1$.

We iterate down until $B^{(0)}$, and observe that $B^{(0)} = B$. Thus, with $\mathcal{O}(\log n)$ calls to MonoConvolution we can compute $B = S^{\leqslant 2^{i+1}}$ from $A = S^{\leqslant 2^i}$. To solve CoinChange the total number of MonoConvolution calls is $\mathcal{O}\big(\log^2 n\big)$. Therefore if MonoConvolution can be solved in $T(n)$ time, then CoinChange can be solved in $\mathcal{O}\big(T(n) \log^2 n\big)$ time. ◀

References

1. Amir Abboud, Fabrizio Grandoni, and Virginia Vassilevska Williams. Subcubic Equivalences Between Graph Centrality Problems, APSP and Diameter. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 1681–1697, 2015. doi:10.1137/1.9781611973730.112.
2. Noga Alon, Zvi Galil, and Oded Margalit. On the Exponent of the All Pairs Shortest Path Problem. *Journal of Computer and System Sciences*, 54(2):255–262, 1997. doi:10.1006/jcss.1997.1388.
3. Noga Alon, Raphael Yuster, and Uri Zwick. Finding and counting given length cycles. *Algorithmica*, 17(3):209–223, March 1997. doi:10.1007/BF02523189.
4. Ilya Baran, Erik D. Demaine, and Mihai Pătraşcu. Subquadratic Algorithms for 3SUM. *Algorithmica*, 50(4):584–596, April 2008. doi:10.1007/s00453-007-9036-3.
5. Hodaya Barr, Tsvi Kopelowitz, Ely Porat, and Liam Roditty. $\{-1, 0, 1\}$-APSP and (min,max)-product problems, 2019. arXiv:1911.06132.
6. Michael A. Bender, Giridhar Pemmasani, Steven Skiena, and Pavel Sumazin. Finding least common ancestors in directed acyclic graphs. In *Proceedings of the Twelfth Annual Symposium on Discrete Algorithms, January 7-9, 2001, Washington, DC, USA.*, pages 845–854, 2001. URL: http://dl.acm.org/citation.cfm?id=365411.365795.
7. David Bremner, Timothy M. Chan, Erik D. Demaine, Jeff Erickson, Ferran Hurtado, John Iacono, Stefan Langerman, and Perouz Taslakian. Necklaces, Convolutions, and X + Y. In *Proceedings of the 14th Conference on Annual European Symposium - Volume 14*, ESA'06, pages 160–171, London, UK, UK, 2006. Springer-Verlag. doi:10.1007/11841036_17.
8. Karl Bringmann, Allan Grønlund, and Kasper Green Larsen. A Dichotomy for Regular Expression Membership Testing. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 307–318, 2017. doi:10.1109/FOCS.2017.36.

9 Karl Bringmann and Tomasz Kociumaka. Personal communication, 2019.

10 Karl Bringmann, Marvin Künnemann, and Karol Wegrzycki. Approximating APSP Without Scaling: Equivalence of Approximate Min-plus and Exact Min-max. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2019, pages 943–954, New York, NY, USA, 2019. ACM. doi:10.1145/3313276.3316373.

11 Timothy M. Chan. More Logarithmic-Factor Speedups for 3SUM, (median, +)-Convolution, and Some Geometric 3SUM-Hard Problems. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 881–897, 2018. doi:10.1137/1.9781611975031.57.

12 Lijie Chen and Ryan Williams. An Equivalence Class for Orthogonal Vectors. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 21–40, 2019. doi:10.1137/1.9781611975482.2.

13 Marek Cygan, Holger Dell, Daniel Lokshtanov, Dániel Marx, Jesper Nederlof, Yoshio Okamoto, Ramamohan Paturi, Saket Saurabh, and Magnus Wahlström. On Problems as Hard as CNF-SAT. *ACM Transactions on Algorithms*, 12(3):41:1–41:24, 2016. doi:10.1145/2925416.

14 Marek Cygan, Marcin Mucha, Karol Wegrzycki, and Michal Wlodarczyk. On Problems Equivalent to (Min,+)-Convolution. *ACM Transactions on Algorithms*, 15(1):14:1–14:25, January 2019. doi:10.1145/3293465.

15 Artur Czumaj, Mirosław Kowaluk, and Andrzej Lingas. Faster algorithms for finding lowest common ancestors in directed acyclic graphs. *Theoretical Computer Science*, 380(1):37–46, 2007. Automata, Languages and Programming. doi:10.1016/j.tcs.2007.02.053.

16 Artur Czumaj and Andrzej Lingas. Finding a Heaviest Triangle is Not Harder Than Matrix Multiplication. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, pages 986–994, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics. URL: http://dl.acm.org/citation.cfm?id=1283383.1283489.

17 Martin Dietzfelbinger. Universal hashing and k-wise independent random variables via integer arithmetic without primes. In *STACS 96*, pages 567–580, Berlin, Heidelberg, 1996. Springer Berlin Heidelberg. doi:10.1007/3-540-60922-9_46.

18 Ran Duan, Ce Jin, and Hongxun Wu. Faster Algorithms for All Pairs Non-Decreasing Paths Problem. In *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece.*, pages 48:1–48:13, 2019. doi:10.4230/LIPIcs.ICALP.2019.48.

19 Ran Duan and Seth Pettie. Fast algorithms for (max, min)-matrix multiplication and bottleneck shortest paths. In *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2009, New York, NY, USA, January 4-6, 2009*, pages 384–391, 2009. doi:10.1137/1.9781611973068.43.

20 Lech Duraj, Krzysztof Kleiner, Adam Polak, and Virginia Vassilevska Williams. Equivalences between triangle and range query problems. *arXiv e-prints*, page arXiv:1908.11819, August 2019. arXiv:1908.11819.

21 Michael J Fischer and Albert R Meyer. Boolean matrix multiplication and transitive closure. In *12th Annual Symposium on Switching and Automata Theory, SWAT 1971*, pages 129–131. IEEE, 1971. doi:10.1109/SWAT.1971.4.

22 Anka Gajentaan and Mark H. Overmars. On a Class of $O(n^2)$ Problems in Computational Geometry. *Computational Geometry*, 5:165–185, 1995. doi:10.1016/0925-7721(95)00022-2.

23 Francois Le Gall and Florent Urrutia. Improved Rectangular Matrix Multiplication using Powers of the Coppersmith-Winograd Tensor. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 1029–1046, 2018. doi:10.1137/1.9781611975031.67.

24 Omer Gold and Micha Sharir. Dominance Product and High-Dimensional Closest Pair under L_∞. In Yoshio Okamoto and Takeshi Tokuyama, editors, *28th International Symposium on Algorithms and Computation (ISAAC 2017)*, volume 92 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 39:1–39:12, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.ISAAC.2017.39.

25 Allan Grønlund and Seth Pettie. Threesomes, Degenerates, and Love Triangles. *Journal of the ACM*, 65(4):22:1–22:25, April 2018. doi:10.1145/3185378.

26 MohammadTaghi Hajiaghayi, Silvio Lattanzi, Saeed Seddighin, and Cliff Stein. MapReduce meets fine-grained complexity: Mapreduce algorithms for APSP, matrix multiplication, 3-SUM, and beyond, 2019. arXiv:1905.01748.

27 Tsvi Kopelowitz, Seth Pettie, and Ely Porat. Higher Lower Bounds from the 3SUM Conjecture. In *Proceedings of the Twenty-seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '16, pages 1272–1287, Philadelphia, PA, USA, 2016. Society for Industrial and Applied Mathematics. doi:10.1137/1.9781611974331.ch89.

28 Marvin Künnemann, Ramamohan Paturi, and Stefan Schneider. On the Fine-Grained Complexity of One-Dimensional Dynamic Programming. In Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl, editors, *44th International Colloquium on Automata, Languages, and Programming (ICALP 2017)*, volume 80 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 21:1–21:15, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.ICALP.2017.21.

29 Karim Labib, Przemyslaw Uznanski, and Daniel Wolleb-Graf. Hamming Distance Completeness. In Nadia Pisanti and Solon P. Pissis, editors, *30th Annual Symposium on Combinatorial Pattern Matching (CPM 2019)*, volume 128 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 14:1–14:17, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.CPM.2019.14.

30 François Le Gall. Powers of Tensors and Fast Matrix Multiplication. In *Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation*, ISSAC '14, pages 296–303, New York, NY, USA, 2014. ACM. doi:10.1145/2608628.2608664.

31 Andrea Lincoln, Virginia Vassilevska Williams, Joshua R. Wang, and R. Ryan Williams. Deterministic Time-Space Trade-Offs for k-SUM. In *43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016)*, volume 55 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 58:1–58:14, Dagstuhl, Germany, 2016. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.ICALP.2016.58.

32 Jiří Matoušek. Computing Dominances in E^n. *Information Processing Letters*, 38(5):277–278, 1991. doi:10.1016/0020-0190(91)90071-O.

33 Mihai Patrascu. Towards Polynomial Lower Bounds for Dynamic Problems. In *Proceedings of the Forty-second ACM Symposium on Theory of Computing*, STOC '10, pages 603–610, New York, NY, USA, 2010. ACM. doi:10.1145/1806689.1806772.

34 Raimund Seidel. On the All-Pairs-Shortest-Path Problem in Unweighted Undirected Graphs. *Journal of Computer and System Sciences*, 51(3):400–403, 1995. doi:10.1006/jcss.1995.1078.

35 Asaf Shapira, Raphael Yuster, and Uri Zwick. All-Pairs Bottleneck Paths in Vertex Weighted Graphs. *Algorithmica*, 59(4):621–633, 2011. doi:10.1007/s00453-009-9328-x.

36 Volker Strassen. Gaussian elimination is not optimal. *Numerische mathematik*, 13(4):354–356, 1969. doi:10.1007/BF02165411.

37 Virginia Vassilevska. Nondecreasing paths in a weighted graph or: how to optimally read a train schedule. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2008, San Francisco, California, USA, January 20-22, 2008*, pages 465–472, 2008. URL: http://dl.acm.org/citation.cfm?id=1347082.1347133.

38 Virginia Vassilevska, Ryan Williams, and Raphael Yuster. All-pairs bottleneck paths for general graphs in truly sub-cubic time. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007*, pages 585–589, 2007. doi:10.1145/1250790.1250876.

39 Virginia Vassilevska, Ryan Williams, and Raphael Yuster. All Pairs Bottleneck Paths and Max-Min Matrix Products in Truly Subcubic Time. *Theory of Computing*, 5(1):173–189, 2009. doi:10.4086/toc.2009.v005a009.

40 Virginia Vassilevska, Ryan Williams, and Raphael Yuster. Finding Heaviest H-subgraphs in Real Weighted Graphs, with Applications. *ACM Transactions on Algorithms*, 6(3):44:1–44:23, July 2010. doi:10.1145/1798596.1798597.

41 Virginia Vassilevska Williams. Nondecreasing paths in a weighted graph or: How to optimally read a train schedule. *ACM Transactions on Algorithms*, 6(4):70:1–70:24, 2010. doi:10.1145/1824777.1824790.

42 Virginia Vassilevska Williams. Multiplying Matrices Faster Than Coppersmith-Winograd. In *Proceedings of the Forty-fourth Annual ACM Symposium on Theory of Computing*, STOC '12, pages 887–898, 2012. doi:10.1145/2213977.2214056.

43 Virginia Vassilevska Williams. On some fine-grained questions in algorithms and complexity. In *Proceedings of the International Congress of Mathematicians (ICM 2018)*, pages 3447–3487, 2018. doi:10.1142/9789813272880_0188.

44 Virginia Vassilevska Williams and R. Ryan Williams. Subcubic Equivalences Between Path, Matrix, and Triangle Problems. *Journal of the ACM*, 65(5):27:1–27:38, August 2018. doi:10.1145/3186893.

45 Virginia Vassilevska Williams and Ryan Williams. Finding, Minimizing, and Counting Weighted Subgraphs. *SIAM Journal on Computing*, 42(3):831–854, 2013. doi:10.1137/09076619X.

46 Ryan Williams. Faster All-pairs Shortest Paths via Circuit Complexity. In *Proceedings of the Forty-sixth Annual ACM Symposium on Theory of Computing*, STOC '14, pages 664–673, New York, NY, USA, 2014. ACM. doi:10.1145/2591796.2591811.

47 Virginia V. Williams. Problem Set 2 in Stanford's class CS367, Oct. 15, 2015. http://theory.stanford.edu/~virgi/cs367/hw2.pdf, 2015.

48 J. W. Wright. The Change-Making Problem. *Journal of the ACM*, 22(1):125–128, January 1975. doi:10.1145/321864.321874.

49 Raphael Yuster. Efficient algorithms on sets of permutations, dominance, and real-weighted APSP. In *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2009, New York, NY, USA, January 4-6, 2009*, pages 950–957, 2009. doi:10.1137/1.9781611973068.103.

50 Uri Zwick. All Pairs Shortest Paths Using Bridging Sets and Rectangular Matrix Multiplication. *Journal of the ACM*, 49(3):289–317, May 2002. doi:10.1145/567112.567114.

Matching Is as Easy as the Decision Problem, in the NC Model

Nima Anari
Computer Science Department, Stanford University, CA, United States
https://nimaanari.com
anari@cs.stanford.edu

Vijay V. Vazirani
Computer Science Department, University of California, Irvine, CA, United States
https://www.ics.uci.edu/~vazirani/
vazirani@ics.uci.edu

Abstract

Is matching in NC, i.e., is there a deterministic fast parallel algorithm for it? This has been an outstanding open question in TCS for over three decades, ever since the discovery of randomized NC matching algorithms [17, 27]. Over the last five years, the theoretical computer science community has launched a relentless attack on this question, leading to the discovery of several powerful ideas. We give what appears to be the culmination of this line of work: An NC algorithm for finding a minimum-weight perfect matching in a general graph with polynomially bounded edge weights, provided it is given an oracle for the decision problem. Consequently, for settling the main open problem, it suffices to obtain an NC algorithm for the decision problem. We believe this new fact has qualitatively changed the nature of this open problem.

All known efficient matching algorithms for general graphs follow one of two approaches: given by [6] and [20]. Our oracle-based algorithm follows a new approach and uses many of ideas discovered in the last five years.

The difficulty of obtaining an NC perfect matching algorithm led researchers to study matching vis-a-vis clever relaxations of the class NC. In this vein, recently [10] gave a pseudo-deterministic RNC algorithm for finding a perfect matching in a bipartite graph, i.e., an RNC algorithm with the additional requirement that on the same graph, it should return the same (i.e., unique) perfect matching for almost all choices of random bits. A corollary of our reduction is an analogous algorithm for general graphs.

2012 ACM Subject Classification Theory of computation → Parallel algorithms; Theory of computation → Graph algorithms analysis; Theory of computation → Pseudorandomness and derandomization

Keywords and phrases Parallel Algorithm, Pseudo-Deterministic, Perfect Matching, Tutte Matrix

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.54

Related Version A full version of the paper is available at https://arxiv.org/abs/1901.10387.

Funding *Vijay V. Vazirani*: Supported in part by NSF grant CCF-1815901.

1 Introduction

Is matching in NC, i.e., is there a deterministic fast parallel[1] algorithm for finding a perfect or, more generally, a maximum matching in a general graph? This has been an outstanding open question in theoretical computer science for over three decades, ever since the discovery of RNC matching algorithms [17, 27]. Over the last five years, the TCS community has launched a relentless attack on this question, leading to the discovery of numerous powerful

[1] That runs in polylogarithmic time using polynomially many processors.

ideas [8, 32, 10, 1, 31]. We give what appears to be the culmination of this line of work: An NC algorithm for finding a minimum weight perfect matching in a general graph with polynomially bounded edge weights, provided it is given an oracle, say $\mathcal{O}$, for the decision problem. Consequently, for settling the main open problem, it suffices to obtain an NC algorithm for the decision problem. We believe this new fact has qualitatively changed the nature of this open problem. Henceforth, by *small weights* we will mean *polynomially bounded edge weights* and *acronym MWPM* will be short for *minimum weight perfect matching.*

The difficulty of obtaining an NC matching algorithm led researchers to study matching vis-a-vis certain clever relaxations of the class NC. One such relaxation is pseudo-deterministic RNC. This is an RNC algorithm with the additional property that on the same graph, it must return the same (i.e., unique) solution for almost all choices of random bits [9, 10]. Recently, [10] gave such an algorithm for perfect matching in bipartite graphs. A second relaxation of NC is quasi-NC, under which the algorithm must run in polylogarithmic time, though it can use $O(n^{\log^{O(1)} n})$ processors; see Section 1.1 for results obtained for this model.

A corollary of our result extends [10] to general graphs as follows: The precise decision problem for our result is: Given a graph G with small weights and a number W, is there a perfect matching of weight at most W in G. Clearly, this is NC equivalent to: Find the weight of a minimum weight perfect matching in G. This question is easy to answer in RNC with inverse-polynomial probability of error using the algorithm of [27]. Therefore, using this RNC algorithm in place of the oracle we get an RNC matching algorithm with the property that in a run, all queries to the decision problem will be answered correctly with overwhelming probability, i.e., this is a pseudo-deterministic RNC matching algorithm.

All known efficient matching algorithms for general graphs follow one of two approaches: given by [6] and [20]. Our oracle-based algorithm follows a new approach and uses many of ideas discovered in the last five years. In particular, it uses the overall structure of the recent NC algorithm of [1] for finding a perfect matching in planar graphs. Since oracle $\mathcal{O}$ can be implemented in NC for planar graphs, our current paper yields a simpler NC algorithm for finding a perfect matching in planar graphs. The second key ingredient which made our current result possible is an NC algorithm for finding a maximal laminar family of tight odd sets in a given face of the perfect matching polytope. This follows from the works of [4] and [31].

Our main result is:

▶ **Theorem 1.** *There is an* NC *algorithm for finding a minimum weight perfect matching in general graphs with small weights, provided the algorithm is given access to oracle $\mathcal{O}$ for the decision problem. The latter is: Given a graph G with small weights and a target weight W, is there a perfect matching of weight at most W in G?*

▶ **Corollary 2.** *There is an* NC *algorithm for finding a maximum matching in general graphs, provided the algorithm is given access to the oracle $\mathcal{O}$.*

▶ **Corollary 3.** *There is a pseudo-deterministic* RNC *algorithm for finding a minimum weight perfect matching in general graphs with small weights.*

We further show that our algorithms only need to call the decision oracle for minors of the input graph.

▶ **Theorem 4.** *Let $\mathcal{F}$ be a minor-closed family of graphs. If there is an* NC *algorithm for deciding whether a perfect matching of weight at most W exists in graphs from $\mathcal{F}$, weighted with polynomially small weights, then there is also an* NC *algorithm for* finding *a MWPM in such graphs.*

1.1 Related work and a brief history of parallel matching algorithms

The notion of a pseudo-deterministic algorithm with polynomial expected running time was given by [9]. Such an algorithm runs in expected polynomial time and is required to output the same (i.e., unique) solution on a given instance on each run with high probability. Hence, in this sense, it resembles a deterministic algorithm. [9] gave pseudo-deterministic polynomial expected running time algorithms for several number theoretic and cryptographic problems. The notion of pseudo-deterministic RNC algorithms was defined by [10].

An RNC algorithm for the decision problem, of determining if a graph has a perfect matching, was obtained by [20], using the Tutte matrix of the graph. The first RNC algorithm for the search problem, of actually finding a perfect matching, was obtained by [18]. This was followed by a simpler and more versatile algorithm due to [27]; besides perfect matching, it also yielded RNC algorithms for the problem of exact matching (see Section 8) and for finding a MWPM in a graph with small weights. The latter fact is crucially used for obtaining pseudo-deterministic RNC algorithms for bipartite graphs [10] and general graphs (current paper). The "philosophy" behind [27] will be useful for dealing with a difficulty that arises in the design of the current algorithm as well, so it is recalled below[2].

The matching problem occupies an especially distinguished position in the theory of algorithms: Some of the most central notions and powerful tools within this theory were discovered in the context of an algorithmic study of this problem, including the notion of polynomial time solvability [6], the counting class #P [33] and a polynomial time equivalence between random generation and approximate counting for self-reducible problems [15], which lies at the core of the Markov chain Monte Carlo method. The perspective of parallel algorithms has also led to such a gain, namely the Isolation Lemma [27], which has found several applications in complexity theory and algorithms. Considering the fundamental insights gained from an algorithmic study of perfect matchings, the problem of obtaining an NC algorithm for it has remained a premier open question ever since the 1980s.

The first substantial progress on this question was made for the case of planar bipartite graphs by [26] via a flow-based approach, followed by [24] using the fact that there is an NC algorithm for counting perfect matchings in planar graphs. The long-standing problem of extending this result to non-bipartite planar graphs was resolved by [1]. Subsequently, [31] also got the same result using different ideas. [1] also extended their algorithm to constant genus graphs. Subsequently, [7] gave an NC algorithm for perfect matching in one-crossing-minor-free graphs, which include K_5-free graphs and $K_{3,3}$-free graphs; the resolution of the latter class settles a thirty-year-old open problem asked in [34].

The quasi-NC algorithms for matching and its generalizations, mentioned above, work by achieving a partial derandomization of the Isolation Lemma. First, [8] gave a quasi-NC algorithm for perfect matching in bipartite graphs, which was followed by the algorithm of [32] for general graphs. Algorithms were also obtained for the generalization of bipartite matching to the linear matroid intersection problem by [11], and to a further generalization of isolating a vertex of a polytope with faces given by totally unimodular constraints, by [12].

[2] Under the NC model, any one processor does not even have enough time to read the entire input, and hence can perform only local computations. On the other hand, a perfect matching is a global object, unlike say, a maximal independent set. Further difficulties arise from the fact that the number of perfect matchings in a graph can vary widely, all the way from one to exponentially many (assuming it has at least one). If there were a unique perfect matching in the graph, the algorithm's task would become a lot simpler. [27] achieve uniqueness via a powerful probabilistic fact, the Isolating Lemma: under an assignment of randomly chosen small weights to the edges it claims that the MWPM will be unique with high probability.

1.2 What is the "right" decision problem?

Consider the following two decision problems for perfect matching:

- Given a graph G with small weights and a target W, is there a perfect matching of weight at most W in G?
- Does graph G have a perfect matching?

Clearly, the second can be reduced to the first and is therefore "easier". This leads to a legitimate question: why not attempt to reduce, in NC, the search problem to the second decision problem? Our experience suggests that the first problem is much more basic for the setting at hand. We next provide evidence to this effect.

Seeking a MWPM in a graph with small weights was the central problem in the work of [27]. The Isolating Lemma helped find small weights under which there was a unique MWPM. The second half of [27] gave an NC algorithm for finding this (unique) perfect matching, using the Tutte matrix of the graph and matrix inversion; the latter is known to be in NC [3]. Ever since then, perhaps the most popular avenue for obtaining an NC matching algorithm has been to derandomize the Isolating Lemma. This would deterministically yield small weights under which there is a unique MWPM, and it could be found using the second half of [27].

The question of MWPM in a graph with small weights plays a central role in NC-type approaches to all non-bipartite, and even some bipartite, perfect matching algorithms: partial derandomization leading to quasi-NC algorithms [8, 32], resolution of the open problem of non-bipartite planar graphs [1, 31], and quasi-deterministic RNC algorithms for bipartite [10] and general graphs (current paper).

In mathematics, sometimes solving the harder problem turns out to be easier than solving the easier one, if the former has a better "behavior". Our belief is that this is the case here.

1.3 Bipartite vs non-bipartite matching: An intriguing phenomenon

Decades of algorithmic work on the matching problem, from numerous perspectives, exhibits the following intriguing phenomenon: The bipartite case gets solved first. Then, using much more elaborate machinery, the general graph case also follows and yields the exact same result! This phenomenon is made all the more fascinating by the fact that the "elaborate machinery" consists not of one fact but numerous different structural properties and mathematical facts which happen to be just right for the problem at hand! We give a number of examples below.

The duality between maximum matching and minimum vertex cover for bipartite graphs extends to general graphs via the notion of an odd set cover, see [21]. The formulation of the perfect matching polytope for bipartite graphs extends by introducing constraints corresponding to odd sets [6]. Polynomial time algorithms for maximum matching and maximum weight matching in bipartite graphs generalize via the notion of blossoms [21]. The most efficient known algorithm for maximum matching in bipartite graphs [13, 19] obtained via an alternating breadth first search, extends via a much more elaborate algorithm with the same running time via the graph search procedure of double depth first search [25] and blossoms defined from the perspective of minimum length alternating paths [35]. The RNC matching algorithms [18, 27] use Tutte's theorem to extend to general graphs. The randomized matching algorithm of [30] uses Tutte's theorem and a theorem of Frobenius about ranks of sub-matrices of skew-symmetric matrices.

More recent work exhibits this phenomenon as well. The quasi-NC algorithm of [8] for bipartite graphs extends by handling tight odd cuts appropriately [32]. The NC algorithm of [24] for planar bipartite graphs was extended to non-bipartite graphs via Edmonds' formulation of the perfect matching polytope [6], an NC algorithm for max-flow in planar

graphs [16], and a result of [28] proving that the Gomory-Hu tree of a graph must contain a tight odd cut, and an elaborate NC algorithm for uncrossing tight odd cuts [1]. In the same vein, the current paper is extending the pseudo-deterministic RNC bipartite algorithm of [10] by giving a way of dealing with tight odd cuts in Edmonds' formulation of the perfect matching polytope [6] and using an NC procedure for finding a maximal laminar family of tight odd cuts [4, 31].

2 Overview and Technical Ideas

Most of this paper will concentrate on the problem of finding a perfect matching in a general graph in NC, given oracle $\mathcal{O}$. In Section 6.1 we will extend our ideas to finding a MWPM for small weights. Then, an algorithm for finding a maximum matching in a general graph in NC will easily follow. In this section, we will also give a number of key definitions which will be used throughout the paper.

2.1 The bipartite case

For ease of comprehension, we will first give an outline of a proof of Theorem 1 for the case of bipartite graphs. Such a proof can be gleaned from the paper of [10]; however, to the best of our knowledge, this important fact was not derived so far. Below, we build on the quasi-NC algorithm of [8] to obtain a somewhat simpler proof of this result.

The algorithm of [8] first starts with the perfect matching polytope and then iteratively moves to lower dimensional faces of this polytope, terminating when a vertex of the polytope is reached; this will be a perfect matching.

▶ **Definition 5.** *In a general graph $G = (V, E)$ with edge weight function w, an edge e is called an* allowed edge *if it participates in MWPM. Let $E[w]$ denote the set of all allowed edges. Edges in the complement of this set will be called* disallowed edges.

Assume w are small weights and let $\mathrm{PM}[w]$ denote the face of the polytope containing all fractional and integral MWPMs w.r.t. w. Since we are in the bipartite case, $\mathrm{PM}[w]$ has a simple description: It is defined by the set of disallowed edges, since they are set to zero, or equivalently its complement, i.e., the set of allowed edges, $E[w]$. The description of the algorithm given above can be refined to: Iteratively modify the weight vector w so that the dimension of face $\mathrm{PM}[w]$ keeps dropping, and equivalently $E[w]$ keeps getting sparser, until $E[w]$ is a perfect matching.

As argued earlier, using oracle $\mathcal{O}$, we can find the weight of a MWPM in G. Further, it is easy to see that for a given edge e, we can determine in NC if e participates in a MWPM, i.e., if $e \in E[w]$. Repeating for all edges in parallel, we can compute $E[w]$ in NC. The following is a fundamental notion in all recent NC-type matching algorithms:

▶ **Definition 6** ([5])**.** *Number the edges of an even cycle C in a general graph G with edge-weights w starting from an arbitrary edge. The* circulation of cycle C *is the absolute value of the difference of the sum of weights of odd-numbered and even-numbered edges and is denoted* $\mathrm{circ}_w(C)$.

It is easy to prove that if the MWPM in G is not unique, then any cycle in the symmetric difference of two such matchings must have zero circulation[3].

[3] Hence, if we find a weight vector w such that each cycle in G has nonzero circulation, then the MWPM must be unique and can be found in NC.

▶ **Proposition 7** ([8]). *In a bipartite graph, let cycle $C \subseteq E[w]$ have $\mathrm{circ}_w(C) = 0$. Let G denote the graph on edge set $E[w]$. Assign small weights w' to edges $E[w]$ so that $\mathrm{circ}'_w(C) > 0$. Then C will not be present in $E[w']$, i.e., at least one of its edges will be dropped in going from $E[w]$ to $E[w']$. We will say that C got* destroyed.

Hence, if we find a weight vector that destroys all cycles of G, we would be done. However, G may have exponentially many cycles, so this is non-trivial. One of the key ideas of [8] is a systematic way of destroying cycles: They iteratively destroy cycles of length $4, 8, 16, \ldots, n$; clearly, the number of iterations needed is $O(\log n)$. In the first round, G has at most $O(n^4)$ cycles of length 4. [8] show that if all cycles of length at most 2^i have already been destroyed, then there are at most $O(n^4)$ cycles of length at most 2^{i+1} left. Hence, in each iteration only $O(n^4)$ cycles need to be destroyed.

Suppose the current iteration starts with small weights w under which all cycles of length at most 2^i have already been destroyed. In this iteration, the algorithm finds a weight vector w' for the edges in $E[w]$ under which all cycles of length at most 2^{i+1} are destroyed. The following fact will play a central role in the current paper as well:

▶ **Proposition 8** ([8]). *In order to destroy any set of s cycles, it suffices to try certain well-chosen $O(n^2 s)$ integral weight vectors each of which uses numbers that are $O(n^2 s)$; one of these vectors is sure to work.*

Since in the current iteration $s = O(n^4)$, at most $O(n^6)$ weight vectors suffice. The algorithm for choosing a weight vector that works is as follows. In parallel, for each of the $O(n^6)$ weight vectors, y, compute $E[y]$ and find the girth of the resulting graph; this can easily be done in NC. Pick the lexicographically first weight vector, say w', such that $E[w']$ has girth $> 2^{i+1}$. Clearly, w' destroys all cycles of length at most 2^{i+1}.

2.2 Extension to general graphs

Matching algorithms for general graphs, in different computational models, are far harder because they need to handle odd cycles in special ways. The set of constraints capturing the perfect matching polytope is also more complex: it includes exponentially many odd set constraints. An odd set $S \subset V$ which satisfies this constraint with equality is called a *tight odd set*. The description of face $\mathrm{PM}[w]$ is also much more involved: in addition to edges $E[w]$, we need a maximal laminar family of tight odd sets, say $\mathcal{L}$; see Section 3.2.

The "engine" underlying our algorithm

Analogous to the bipartite case, there is an "engine" underlying our algorithm as well – it iteratively reduces the size of the graph. This engine can be thought of as composed of three components which draw on different domains to establish structural facts and algorithms.

2.2.1 Component based on the structure of the perfect matching polytope

Proposition 7, which yielded the "engine" for the bipartite case, does not hold in general graphs. Thus a non-bipartite graph may have an even cycle $C \subseteq E[w]$ with $\mathrm{circ}_w(C) > 0$. The reason for this is the presence of a tight odd set. As a result, Proposition 7 needs to be enhanced to the fact stated below. We will say that a cycle C *crosses* a tight odd set S if C has vertices in S as well as in $(V - S)$. Similarly, edge e *crosses* S if one of its endpoints is in S and the other is in $V - S$.

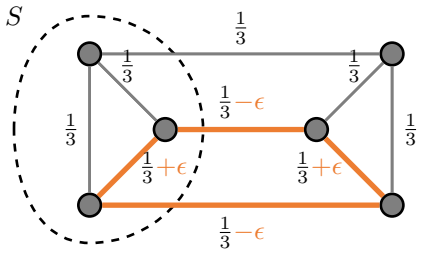


Figure 1 The orange even cycle crosses tight odd set S; example due to [8, 32].

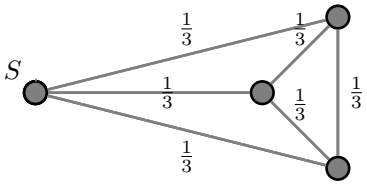


Figure 2 Resulting graph after shrinking tight odd set S.

▶ **Proposition 9** ([32]). *In a general graph G, suppose even cycle $C \subseteq E[w]$ has* $\text{circ}_w(C) > 0$. *Then, there must be a tight odd set S such that C* crosses *S.*

This is illustrated in Figure 1. In this graph, the three edges in $\delta(S)$ have weight 1 and the rest have weight 0. Observe that each edge participates in a MWPM and hence $E[w]$ consists of all edges. The cycle consisting of the four orange edges, say C, has positive circulation even though it is contained in $E[w]$. Cycle C crosses tight odd set S.

▶ **Definition 10.** *Assume that even cycle C* crosses *tight odd set S. Number the edges of C starting from an arbitrary edge. Let n_o and n_e denote the number of odd-numbered and even-numbered edges, respectively, that cross S. Then the* mismatch of C and S, *denoted* $\text{mismatch}(C, S)$, *is $|n_o - n_e|$.*

Note that in Figure 1, $\text{mismatch}(C, S) = 2$. Observe that if the MWPM is not unique and C is a cycle in the symmetric difference of two such perfect matchings then the following must hold:

- $\text{circ}_w(C) = 0$.
- If C crosses a tight odd set S, then $\text{mismatch}(C, S) = 0$; the reason is that each perfect matching crosses each tight set exactly once.

▶ **Proposition 11** (Lemma 25). *Consider a general graph G with weights w and even cycle $C \subseteq E[w]$ with* $\text{circ}_w(C) > 0$. *Let S be a tight odd set such that C crosses S. Then* $\text{mismatch}(C, S) > 0$ *and at least one edge of C has both its endpoints in S.*

Our strategy for dealing with cycle C having $\text{circ}_w(C) > 0$ is to *shrink* the tight odd set S it crosses; this is illustrated in Figure 2. By Proposition 11, this will shrink at least one edge of C, hence resulting in a smaller graph. Our overall strategy is as follow: Suppose w.r.t. weight vector w, $\text{circ}_w(C) = 0$. Let w' be a weight vector such that $\text{circ}_{w'}(C) > 0$. If so, [32] show that either C must lose an edge in going from $E[w]$ to $E[w']$ or a new odd set S goes tight w.r.t. w' such that C crosses S. In the latter case, we shrink S. In either case we will obtain a smaller graph and in both cases we will say that C is *destroyed*.

2.2.2 Component based on graph-theoretic facts

As stated in the Introduction, the overall structure of our algorithm is similar to that of [1]. Both algorithms require in each iteration a large enough number of edge-disjoint even cycles whose destruction will result in the removal of a corresponding number of edges. However, in both cases, the graph may have not such cycles. The recourse is to resort to even walks.

▶ **Definition 12** ([32]). *We call an ordered list of an even number of edges $C = (e_1, \ldots, e_{2k})$, not necessarily distinct, that start and end at the same vertex, an* even walk *if this list traverses either a simple even cycle or two odd cycles with a path joining them; in the latter case, the cycles are traversed once each and the path twice, once in each direction.*

The list C of edges of an even walk contains each edge either once or twice, and if it contains an edge e twice, then both copies will have the same parity. The notions of circulation and mismatch can be extended to even walks in a natural way by taking into consideration multiplicity of edges. Thus if e occurs twice in walk C, is odd-numbered and crosses tight odd set S, then it contributes 2 to n_o in the computation of mismatch(C, S) (see Definition 10) and it contributes $2w_e$ to the sum of odd-numbered edges in the computation of $\text{circ}_w(C)$ (see Definition 6). As shown in [32], all statements made above about destroying even cycles carry over to even walks as well.

[1] critically used Euler's formula and the planar dual of G for first finding a large number of edge-disjoint cycles in NC. If more than half were even, they sufficed. Otherwise, they paired up odd cycles and found paths connecting each pair to obtain even walks. This was done in a such a manner that the resulting even walks were edge-disjoint.

Finding edge-disjoint cycles in a general graph in NC appears to be quite difficult. Instead, we take a cue from the bipartite case, which finessed the issue of finding edge-disjoint cycles by using Proposition 7. As a result, showing the *existence* of cycles sufficed! However, there is a subtle difference: in the bipartite case, we needed to upper bound the number of cycles that needed to be destroyed in each iteration, whereas here we need to lower bound them; the latter is the case in [1] as well.

Using ideas from [2] we show that if the graph $G = (V, E)$ is not very sparse (see Definition 31), then it contains $\Omega\left(\frac{|E|}{\log^2|V|}\right)$ edge-disjoint even cycles. Then, using ideas from [1], we show how to pair up odd cycles to form walks. Unlike [1], the walks don't need to be found explicitly – establishing existence suffices.

If in an iteration the graph is very sparse, it will not have the required number of edge-disjoint cycles. For this case, we define the notion of a *triad* in Definition 21; this is a tight odd set consisting of three vertices. We show that the graph has sufficiently many disjoint triads, and a maximal independent set algorithm can find a large enough subset of these in NC. These can be shrunk simultaneously.

2.2.3 Component based on facts from matching theory

Suppose that in a certain iteration our algorithm is trying weight function w, as per Proposition 7. We will need to find in NC a description of face PM$[w]$, which involves, in addition to edges $E[w]$, a maximal laminar family of tight odd sets, say $\mathcal{L}$. Computing $E[w]$ using oracle $\mathcal{O}$ is straightforward. However, finding family $\mathcal{L}$ in NC is a difficult question. The difficulty is similar to that of finding a perfect matching in a graph, i.e., the presence of a plethora of solutions. Recall the "philosophy" of [27] given in Section 1.1, for dealing with this issue for perfect matching, namely attempt to narrow down the choices to one. Clearly unlike [27], randomization is not a resource we can use for this purpose. The solution involves imposing more and more restrictions on the family of tight odd sets until it becomes unique! These restrictions arise from deep structural facts from matching theory. Additional facts lead to an NC algorithm for computing $\mathcal{L}$ with the help of $\mathcal{O}$. These ideas are from [4] and [31] and are given in Section 3.2.

For the "correct" weight function, say w, among the set of even walks being handled in this iteration, some will be destroyed by losing an edge and some by crossing a tight odd set. By updating the edge set to $E[w]$, we can accrue the advantage from the first set of walks. For obtaining advantage from the second set of walks, for each such walk, say C, we need to shrink a tight odd set, say S, that it crosses. A major obstacle is that our algorithm does not "know" *any* of the walks! The way we finesse this difficulty is to shrink all outermost sets of L, which are clearly be disjoint, in the graph on edge set $E[w]$.

Finally, among all weight functions, we will pick the one, say w, that yields a graph with the smallest number of edges. There is no guarantee that w would have destroyed all s walks which we had established the existence of up-front. However, at least one of the weight functions must have done so and therefore led to a decrease of at least s edges. Hence, w must also decrease at least s edges, and that suffices for making progress. As shown in Lemma 42, the number of non-isolated edges gets reduced by a factor of $1 - \Omega(1/\log^2|V|)$ in each iteration.

2.3 The final idea: balanced viable set

Our current strategy is to iteratively reduce the number of edges until a perfect matching remains. After picking its edges, we need to recursively find a perfect matching in each of the shrunk sets (after removing its matched vertex). The resulting algorithm would have polylogarithmic depth; however, it does not run in polylogarithmic time because of the following inherent sequentiality: Perfect matchings in shrunk sets can be found only *after* finding a perfect matching in the shrunk graph, because the algorithm needs to know the vertex in S that is matched outside S. Moreover, perfect matchings in the shrunk graph and the shrunk sets need to be found via a recursive application of the full algorithm described so far.

The exact same issue arose in [1] as well. The solution proposed there was meant for general graphs and hence it works here as well. The solution is quite elaborate and hence is not repeated here; instead, we direct the reader to Section 4.2 in [1]. We note that the task is somewhat easier here because we have recourse to oracle $\mathcal{O}$; [1] had to resort to computing Pfaffians orientations, etc. We give a short, high-level summary below.

An odd set S is *viable* if there is at least one perfect matching in G which picks exactly one edge from $\delta(S)$. A set S is *balanced* if both S and its complement contain a constant fraction of the vertices. [1] show how to find in NC a balanced viable odd set. Let S be such a set. Clearly, using oracle $\mathcal{O}$, we can find an edge $e \in \delta(S)$ which is the unique edge in a perfect matching from this cut. Now we are done by a simple divide-and-conquer strategy: match e, remove its end-points and find perfect matchings in the two sides of the cut recursively, in parallel. Observe that even though perfect matchings in the two sides can be found only after finding the matched edge e, the latter can be done without any recursive calls, hence, leading to a polylogarithmic running time.

3 Preliminaries

We represent undirected graphs by $G = (V, E)$, where V is the set of vertices and E is the set of edges. Unless otherwise specified, we only work with graphs that have no loops, i.e., an edge from a vertex to itself. An edge between vertices u and v is represented as $\{u, v\}$. For a set $S \subseteq V$, we use $\delta(S)$ to denote the cut between S and its complement, i.e., $\delta(S) = \{\{u, v\} \in E \mid u \in S, v \notin S\}$. When S is a singleton, i.e., $\{v\}$ for some $v \in V$, we use the shorthand $\delta(v) = \delta(\{v\})$. A perfect matching is a subset of edges $M \subseteq E$ such that for all $v \in V$ we have $|M \cap \delta(v)| = 1$.

▶ **Definition 13.** *We call an edge* $e = \{u, v\}$ isolated *if* $\deg(u) = \deg(v) = 1$.

By this definition a graph is a perfect matching if it has no isolated vertices and all of its edges are isolated.

For a set $S \subseteq E$ of edges we use $\mathbb{1}_S \in \mathbb{R}^E$ to denote the indicator of S. We use the shorthand $\mathbb{1}_e$ to denote the e-th element of the standard basis for $\mathbb{R}^E$, where $e \in E$. We denote the standard inner product between vectors $w, x \in \mathbb{R}^E$ by $\langle w, x \rangle$.

Given a convex polytope $P \subseteq \mathbb{R}^E$, and a weight vector $w \in \mathbb{R}^E$, we use $P[w]$ to denote the set of points minimizing the weight function $x \mapsto \langle w, x \rangle$:

$$P[w] = \{x \in P \mid \forall y \in P : \langle w, x \rangle \leq \langle w, y \rangle\}.$$

Note that $P[w]$ is a face of P; all faces of P can be obtained as $P[w]$ for appropriately chosen w.

3.1 The perfect matching polytope

Given a graph $G = (V, E)$, we call a subset of edges $M \subseteq E$ a perfect matching if it contains exactly one edge in every degree cut, i.e., $|M \cap \delta(v)| = 1$ for all v. We call a graph matching-covered if any of its edges can be extended to a perfect matching.

▶ **Definition 14.** *A graph* $G = (V, E)$ *is* matching-covered *if for every edge* $e \in E$*, there exists a perfect matching* M *such that* $e \in M$*.*

The perfect matching polytope for $G = (V, E)$ is the convex hull of all perfect matchings of G in $\mathbb{R}^E$. Thus,

$$\mathrm{PM}_G = \mathrm{conv}\{\mathbb{1}_M \mid M \subseteq E \text{ is a perfect matching of } G\}.$$

Clearly the perfect matchings of G are in one-to-one correspondence with the vertices of this polytope.

When G is clear from context, we simply use PM to refer to this polytope. PM is alternatively described by the following set of linear equalities and inequalities [6]:

$$\mathrm{PM} = \left\{ x \in \mathbb{R}^E \;\middle|\; \begin{array}{ll} \langle \mathbb{1}_{\delta(v)}, x \rangle = 1 & \forall v \in V, \\ \langle \mathbb{1}_{\delta(S)}, x \rangle \geq 1 & \forall S \subseteq V, \text{ with } |S| \text{ odd}, \\ \langle \mathbb{1}_e, x \rangle \geq 0 & \forall e \in E. \end{array} \right\}. \tag{1}$$

Any face F of PM can be either described by a weight vector w, i.e., $F = \mathrm{PM}[w]$, or it can be alternatively described by the set of inequalities turned into equalities in Equation (1). These correspond to odd sets S and edges e. When face F is clear from context, we call odd sets whose inequalities have been turned into equalities, *tight* odd sets. We call an edge e *allowed* if $x_e > 0$ for some $x \in F$, i.e., if the inequality corresponding to e in Equation (1) has not been turned into equality. We use $E[w]$ or $E[F]$ to denote the set of allowed edges in the face $F = \mathrm{PM}[w]$. Putting it all together, to describe a face F it is enough to describe the set of allowed edges as well as tight odd sets.

3.2 Finding a description of a face

A key step in our oracle-based algorithm is: given small weights w, compute a description of the face $F = \mathrm{PM}[w]$. As stated before, using oracle $\mathcal{O}$, $E[w]$ can be computed in NC. However, as far as tight odd sets go, there are typically exponentially many choices of a

family of such sets that suffice. At this point, it will be useful to recall the "philosophy" of [27] given in Section 1.1, namely when designing an NC algorithm, faced with a plethora of solutions, one should attempt to narrow down the choices to one. Clearly unlike [27], randomization is not a resource we can use for this purpose. The solution to this puzzle is indeed one of the keys that enables our result and is described below. It involves imposing more and more structure on the family of tight odd sets we seek until it becomes unique! It turns out that the latter can be computed in NC with the help of $\mathcal{O}$.

Two tight odd sets $S_1, S_2 \subseteq V$ are said to *cross* if they are not disjoint and neither is a subset of the other. A family of these sets $\mathcal{L} \subseteq 2^V$ is said to be *laminar* if no pair of sets in it cross. It is well-known that each face F of the perfect matching polytope can be described by the set of allowed edges and a laminar family of tight odd sets $\mathcal{L}$:

$$F = \left\{ x \in \text{PM} \;\middle|\; \begin{array}{ll} \langle \mathbb{1}_{\delta(S)}, x \rangle = 1 & \forall S \in \mathcal{L}, \\ \langle 1_e, x \rangle = 0 & \forall e \notin E[F]. \end{array} \right\}.$$

In fact, $\mathcal{L}$ can be taken to be any *maximal* laminar family of tight odd sets for the given face F (see, e.g., [32, Lemma 2.2]). Note that we will always include all singletons $\{v\}$ in the laminar family $\mathcal{L}$ since the equalities $\langle \mathbb{1}_{\delta(v)}, x \rangle = 1$ are automatically satisfied over all of PM. However, there are still potentially many choices for the laminar family $\mathcal{L}$ describing face F, so we impose more conditions on $\mathcal{L}$.

▶ **Definition 15.** *Suppose we are given a face $F = \text{PM}[w]$. A* laminar optimal dual solution *is a laminar family $\mathcal{L}$ of tight odd sets, including all singletons, together with a function $\pi : \mathcal{L} \to \mathbb{R}$ such that for $S \in \mathcal{L}$, $\pi(S) > 0$ whenever $|S| > 1$ and for all edges e*

$$w_e \geq \sum_{S \in \mathcal{L} : e \in \delta(S)} \pi(S),$$

with equality for allowed edges.

This definition gives dual solutions for the linear program $\min\{\langle w, x \rangle \mid x \in \text{PM}\}$ that satisfy complimentary slackness and are in *laminar* form. By complimentary slackness, for any such solution, $\sum_{S \in \mathcal{L}} \pi(S)$ is equal to the weight of a MWPM. Laminar optimal dual solutions exist but are still not unique.

[4] showed that extra conditions can be imposed on laminar optimal dual solution to make it unique. They studied the notion of *balanaced critical dual solutions* and they showed how this unique $\mathcal{L}$ can be found by computing *primal* solutions to the MWPM problem. [31] used this procedure to design an alternative NC algorithm for planar graph perfect matching. We describe this procedure below. For more details see the work of [4]. Note that we will not use these rather complex and elaborate extra conditions in any other context, so we will not state them explicitly.

The following was shown by [4, Lemma 28].

▶ **Lemma 16** ([4])**.** *If $E[w]$ is connected, then a balanced critical dual is unique and Algorithm 1 finds its support, the laminar family $\mathcal{L}$.*

It was observed by [31] that all steps of Algorithm 1 can be performed in NC except for finding allowed edges $E[w]$ and the computation of $\mu(v)$'s. We note that using oracle $\mathcal{O}$, both these steps can be also be performed in NC.

▶ Remark 17. When $E[w]$ is not connected, Algorithm 1 still works but should be run in parallel for *each connected component* of $E[w]$.

Algorithm 1 Finding a balanced critical dual.

$\mathcal{L} \leftarrow \{\{v\} \mid v \in V\}$.
for $v \in V$ **in parallel do**
  $\mu(v) \leftarrow \min\{\langle w, \mathbb{1}_M \rangle \mid M \subseteq$
    $E[w]$ is a perfect matching on $V \setminus \{u, v\}$ for some vertex $u\}$.
end
Let $w'_e \leftarrow w_e + \mu(u) + \mu(v)$ for each $e \in E[w]$.
for $t \in \{w'_e \mid e \in E[w]\}$ **in parallel do**
  Find the connected components of the graph $(V, \{e \in E[w] \mid w'_e \leq t\})$.
  Add each nontrivial connected component to $\mathcal{L}$.
end
return $\mathcal{L}$.

3.3 Contraction of tight odd sets, matching minors, and triads

[6] observed that if a collection of tight odd sets are disjoint, one can shrink each one to a single node and obtain a smaller graph whose perfect matchings can be extended to perfect matchings in the original graph. For the sake of completeness we state and prove this fact here.

▶ **Proposition 18.** *Suppose that $F = \mathrm{PM}[w]$ is a face of the matching polytope for $G = (V, E)$ and $S_1, \ldots, S_k$ are disjoint tight odd sets w.r.t. F. Let H be obtained from G by removing disallowed edges and contracting each S_i to a single node. Then any perfect matching in H can be extended to a perfect matching in G.*

Proof. Suppose that M is a perfect matching in H. We can think of edges in M as edges in E as well; in fact $M \subseteq E[w]$, because we remove disallowed edges to obtain H. Because M is a perfect matching in H, for each S_i, there is a unique $e_i \in M \cap \delta_G(S_i)$. Now since e_i is an allowed edge, there must be some perfect matching M_i of G such that $e_i \in M_i$ and $\mathbb{1}_{M_i} \in F$. Since S_i is a tight odd set, M_i cannot have any other edge in $\delta(S_i)$, except for e_i. So if we look at $\{\{u, v\} \in M_i \mid u, v \in S_i\}$, we must have a matching covering all vertices of S_i except for the endpoint of e_i. Combining all of these matchings for $i = 1, \ldots, k$ together with M will give us a perfect matching in G as desired. ◀

Note that the graph H obtained above is a minor of the graph G. But it is not an arbitrary minor. It has the additional property that every perfect matching of it can be extended back to a perfect matching of the original graph. For convenience we name these minors, matching minors.

▶ **Definition 19.** *A matching minor H of a graph G, is a graph that can be obtained by a sequence of the following operations: Pick a face of the matching polytope and a collection of disjoint tight odd sets. Remove disallowed edges, and contract each tight odd set into a single node.*

The following statement follows directly from Proposition 18.

▶ **Lemma 20.** *If H is a matching minor of the graph G, then every perfect matching in H can be extended to a perfect matching in G.*

In our algorithms, we use the simple observation that a path of length 2 on vertices of degree 2 yields a tight odd set for the entire matching polytope. We call these paths triads.

▶ **Definition 21.** *A* triad *in graph $G = (V, E)$ is a set of three vertices $\{a, b, c\}$ such that* $\deg(a) = \deg(b) = \deg(c) = 2$*, and* $\{a, b\}, \{b, c\} \in E$*.*

▶ **Lemma 22.** *A triad $\{a, b, c\}$ is a tight odd set for the matching polytope and all of its faces.*

Proof. The only two neighbors of b are a, c. So in every perfect matching, b must be matched to one of them. The other vertex must have an edge to an outside vertex, and in fact that is the only possible edge in $\delta(\{a, b, c\})$. ◀

▶ Remark 23. Note that the proof of Lemma 22 does not use the assumptions $\deg(a) = \deg(c) = 2$ and only uses $\deg(b) = 2$. We will use these extra assumptions elsewhere, to prove that in certain situations, we can find many triads in our graph.

3.4 Even walks and weight vectors

Even walks were defined in Definition 12. For an even walk C, define the *signature* of C to be the vector:

$$\text{sign}(C) = \sum_{i=1}^{2k} (-1)^i \mathbb{1}_{e_i}.$$

The notions of circulation and mismatch can be stated in terms of signature:

$$\text{circ}_w(C) = |\langle w, \text{sign}(C)\rangle|$$

$$\text{mismatch}(C, S) = |\langle \mathbb{1}_{\delta(S)}, \text{sign}(C)\rangle|$$

Now, there cannot be two distinct points $x, y \in \text{PM}[w]$ whose difference $x - y$ is a multiple of $\text{sign}(C)$, since otherwise we would have $\langle w, x\rangle \neq \langle w, y\rangle$. Another way of stating this is that if $x \in \text{PM}[w]$, then $x + \epsilon\,\text{sign}(C) \notin \text{PM}[w]$ for any $\epsilon \neq 0$. So, some inequality or equality describing $\text{PM}[w]$ must be violated for this point. If we pick x to be in the relative interior of the face $\text{PM}[w]$ we will have some slack for non-tight inequalities describing $\text{PM}[w]$. So the violated constraint for $x + \epsilon\,\text{sign}(C)$ must be a constraint that is tight for the entire face $\text{PM}[w]$. This implies that:

▶ **Lemma 24.** *Let C be an even walk with* $\text{circ}_w(C) > 0$*. Then either there is an edge $e \in C$ that is disallowed, i.e., $e \notin E[w]$, or for any laminar dual $(\mathcal{L}, \pi)$ describing* $\text{PM}[w]$*, there is some set S such that* $\text{mismatch}(C, S) > 0$*.*

For a more detailed proof of this, see [1, 32]. Note that if $\text{mismatch}(C, S) > 0$, then C must have an edge with both endpoints inside S.

▶ **Lemma 25.** *If C is an even walk and S is a tight odd set such that* $\text{mismatch}(C, S) > 0$*, then there is an edge $e = \{u, v\} \in C$ such that $u, v \in S$.*

Proof. If this is not true, then every time C enters S it must immediately exit. So if we compute $\text{mismatch}(C, S)$ by looking at edges that cross S, we always get a $+1$ followed by a -1, and a -1 followed by a $+1$. So the entire sum would be 0 which is a contradiction. ◀

We also borrow from [8] the following important result, which is also stated in [32] and as Proposition 7 in this paper.

▶ **Lemma 26** ([8]). *There exists a polynomial sized family of polynomially bounded weight vectors $\mathcal{W}$, such that for any set of edge disjoint even walks $C_1, \ldots, C_k$, there is some $w \in \mathcal{W}$ which ensures*

$$\forall i : \mathrm{circ}_w(C) > 0.$$

Proof. This lemma is actually proved in [8, 32] for any collection of nonzero vectors, not just $\mathrm{sign}(C_i)$'s, as long as there is both a polynomial bound on the number of vectors and the absolute value of their coordinates. Edge-disjointness of even walks automatically puts a bound of $|E|$ on their number, and the coordinates of our even walks are always bounded in absolute value by 2. ◀

3.5 Maximal independent sets

Given a graph $G = (V, E)$, we call a subset $S \subseteq V$ independent if no edge $e \in E$ has both endpoints in S. We call an independent set maximal if no strict superset $T \supsetneq S$ is independent. We will crucially use the fact that maximal independent sets can be found in NC.

▶ **Theorem 27** ([22]). *There is a deterministic* NC *algorithm that on input graph $G = (V, E)$ returns a maximal independent set $S \subseteq V$.*

We usually want a large, rather than a maximal, independent set. We will use the fact that in bounded degree graphs, any maximal independent set is automatically large.

▶ **Proposition 28.** *If $G = (V, E)$ is a graph with $\deg(v) \leq \Delta$ for all $v \in V$, then any maximal independent set $S \subseteq V$ satisfies*

$$|S| \geq \frac{|V|}{\Delta + 1}.$$

4 The Decision Oracle

We will assume that our algorithm is equipped with an oracle $\mathcal{O}$ which answers the following type of queries: Given a graph $G = (V, E)$ and small weights $w \in \mathbb{Z}^E$, what is the weight of a MWPM in G? We denote the answer by

$$\mathcal{O}(G, w) = \min\{\langle w, x\rangle \mid x \in \mathrm{PM}_G\}.$$

We now list several deterministic NC primitives based on $\mathcal{O}$. Versions of these two lemmas appear implicitly, stated for planar graphs, in [31], but we prove them for the sake of completeness.

▶ **Lemma 29.** *Given access to $\mathcal{O}$, for polynomially bounded $w \in \mathbb{Z}^E$, one can find $E[w]$ in* NC.

Proof. An edge $e = \{u, v\}$ can be in a MWPM if and only if $\mathcal{O}(G, w) = w_e + \mathcal{O}(G - \{u\} - \{v\}, w)$, where $G - \{u\} - \{v\}$ is obtained from G by removing vertices u, v. This can be checked in parallel for all edges e. ◀

▶ **Lemma 30.** *Given access to $\mathcal{O}$, for polynomially bounded $w \in \mathbb{Z}^E$, one can run Algorithm 1 in* NC.

Proof. As was observed by [31], all steps of Algorithm 1 can be run in NC except for finding $E[w]$ and computing $\mu(v)$. Given access to $\mathcal{O}$, we can find $E[w]$ in NC by Lemma 29. Furthermore observe that for any $v \in V$

$$\mu(v) = \min\{\mathcal{O}(G - \{u\} - \{v\}, w) \mid u \in V - \{v\}\},$$

which can be computed by making all queries $\mathcal{O}(G - \{u\} - \{v\}, w)$ in parallel and then taking the minimum. ◀

An implementation for the oracle, in RNC with arbitrarily small inverse polynomial probability of error for general graphs, follows from [27], since they give an RNC algorithm for finding a MWPM for small weights. Since $\mathcal{O}$ is promised to be called at most polynomially many times, the probability of error over the entire run of the algorithm can be made inverse polynomially small.

5 Structural Facts

Our algorithm requires two structural facts, one for the case that the graph G is very sparse and the other for the complementary case. They are encapsulated in Lemmas 32 and 34.

▶ **Definition 31.** *A connected graph $G = (V, E)$ is said to be* very sparse *if $|E| < |V|/(1-\epsilon)$, for some constant $\epsilon < 1/9$.*

▶ **Lemma 32.** *If $G = (V, E)$ is a matching-covered, very sparse graph, then the number of triads in any maximal set of node-disjoint triads in G is at least $c_1|E|$, for some constant $c_1(\epsilon) > 0$.*

The proof of this lemma involves two steps: first, we prove that the total number of triads is large and second, that a maximal node-disjoint set of triads must also be large. The first step is accomplished in the following lemma.

▶ **Lemma 33.** *Suppose that $G = (V, E)$ is a graph with no vertices of degree 0 or 1. Then the number of triads in G is at least $9|V| - 8|E|$.*

Proof. Consider a charging scheme, where we allocate a budget of 1 to each edge, and the edge distributes its budget between its two endpoints. We then sum up the charge on all vertices and use the fact that this sum is exactly $|E|$.

Let $e = \{u, v\}$ be an edge. If neither u nor v is of degree 2, let the edge give $1/2$ to u, and $1/2$ to v. If both u and v are of degree 2, we allocate the budget the same way by splitting it equally between u and v. The only remaining case is when one of u and v has degree 2 and the other has degree at least 3; by symmetry let us assume that $\deg(u) = 2$ and $\deg(v) \geq 3$. Then we allocate $5/8$ to u and $3/8$ to v.

Now let us lower bound the charge that each vertex v receives. Note that the minimum amount v receives from any of its adjacent edges is $3/8$, so an obvious lower bound is $3\deg(v)/8$. If $\deg(v) \geq 3$, this is at least $9/8$. Now consider the case when $\deg(v) = 2$. Then v receives at least $1/2$ from each of its adjacent edges. If one of the neighbors of v is not of degree 2, then the charge that v receives will be at least $1/2 + 5/8 = 9/8$. The only possible case where v does not receive at least $9/8$ is when it is of degree 2, and both of its neighbors are also of degree 2 (the center of a triad), in which case it receives 1.

Now let k be the number of triads. Then, by the above argument the total charge on all the vertices is at least

$$\frac{9}{8}(|V| - k) + k \leq |E|.$$

Rearranging yields $k \geq 9|V| - 8|E|$. ◀

Proof of Lemma 32. We know that the number of triads is at least $9|V|-8|E| = (1-9\epsilon)|E|$. Now consider the conflict graph of triads, where nodes represent triads, and edges represent having an intersection. It is easy to see that any triad can only intersect at most 4 other triads. So the degrees in this conflict graph are bounded by 4. By Proposition 28, any maximal node-disjoint set of triads will contain at least $(1-9\epsilon)|E|/5$ many triads. So we can take $c_1(\epsilon) = (1-9\epsilon)/5$ which is positive for $\epsilon < 1/9$. ◀

▶ **Lemma 34.** *If $G = (V, E)$ is a matching-covered graph on $|V| > 2$ vertices that is not very sparse, then there exist $c_2|E|/\log^2|V|$ edge-disjoint even walks in G, for some constant $c_2(\epsilon) > 0$.*

We first show that there are many edge-disjoint cycles in a non-sparse graph. If at least half of them are even, we are done. Otherwise, we show how to pair up odd cycles and connect them via suitable paths to get sufficiently many edge-disjoint even walks. A proof of the next lemma can be found in [2]; however, for the sake of completeness we provide it here.

▶ **Lemma 35.** *In a graph $G = (V, E)$ there exists a collection of edge-disjoint cycles with at least the following number of cycles:*

$$\frac{|E| - |V|}{2\log_2|V|}.$$

Proof. We prove this by induction on $|V| + |E|$. We have several cases:

- **i)** If there are any loops in the graph, we extract that as one of our cycles, and remove the edge from the graph. The promised quantity goes down by $1/(2\log_2|V|)$ which is $\leq 1/2$. So from now on we assume that G has no loops.
- **ii)** If there are any two parallel edges e, e', we extract those as a cycle of length 2, and remove both from the graph. The promised number of edge-disjoint cycles goes down by $2/(2\log_2|V|) \leq 1$. So adding the cycle we extracted fulfills the promise. From now on we assume that G is simple.
- **iii)** If G has any vertices of degree 0: We can simply remove it and the promised quantity grows.
- **iv)** If G has a vertex of degree 1: We can also remove this vertex. This operation does not change the numerator but shrinks the denominator, which results in a larger promised quantity.
- **v)** If G has a vertex v of degree 2: Let e, e' be the two adjacent edges to v. Remove v, e, e' from the graph, and place a new edge e'' between the two former neighbors of v. By doing this, both $|V|$ and $|E|$ go down by 1. So now the promised number of edge-disjoint cycles becomes larger. By induction we find them, and now we replace the edge e'' if it is used at all in a cycle, by the path of length two consisting of e, e'. Since e'' appears in at most one cycle, this operation preserves edge-disjointness.
- **vi)** Finally if G is a simple graph with no vertices of degree ≤ 2, it must have a cycle of length at most $2\log_2|V|$. If we prove this, we are done by induction, because we can remove the edges of this cycle and the promised quantity goes down by at most 1. Now to prove the existence of this cycle, assume the contrary, that the length of the minimum cycle of the graph is at least $2\log_2|V| + 1$. Pick a vertex v and look at all simple paths of length at most $\log_2|V|$ going out of v. The number of paths of length i is at least twice the number of paths of length $i-1$. This is because every path of length $i-1$ ending at a vertex u can be extended in at least $\deg(u) - 1 \geq 2$ ways, and none of these extensions will intersect themselves, otherwise we would get a cycle of length $\log_2|V| + 1$.

So in the end, the total number of such paths will be $> 2^{\log_2|V|} = |V|$, which means that two of the paths must share an endpoint. But now from the union of these two paths, we can extract a cycle of length at most $\log_2|V| + \log_2|V| = 2\log_2|V|$. ◀

If at least half of the cycles guaranteed by Lemma 35 are odd, we need to pair them up and connect them with paths. We use a spanning tree to do this.

▶ **Proposition 36** ([1, Lemma 20]). *Consider a tree T with an even number of tokens placed on its vertices, with possibly multiple tokens on each vertex. There is a pairing, i.e., a partitioning of tokens into partitions of size two, such that the unique tree paths connecting each pair are all edge-disjoint.*

▶ **Lemma 37.** *Suppose that there are 2ℓ edge-disjoint cycles of odd length in a matching-covered connected graph $G = (V, E)$. Then G contains at least $\Omega(\ell^2/|E|)$ edge-disjoint even walks.*

Proof. We will pair up the odd cycles by paths connecting each pair. This will create ℓ even walks, but they might not be edge-disjoint. We will then show how to extract $\Omega(\ell^2/|E|)$ *edge-disjoint* even walks out of them.

Consider a spanning tree T of G. For each of the 2ℓ odd cycles, pick an arbitrary vertex, and put a token on that vertex. Now we have an even number of tokens on the vertices. We can pair up these tokens, so that the unique tree paths (of possibly length 0) connecting each pair are edge-disjoint, see Proposition 36.

Now for each pair of odd cycles C_1, C_2 whose tokens got paired up, we create an even walk. Let P be the tree path connecting tokens from C_1 and C_2. If P has no common edges with C_1, C_2 we can simply create our even walk, but this is not guaranteed to happen. So instead, traverse P from C_1's token to C_2's token and look at the last exit from C_1; afterwards look for the first time any vertex of C_2 is visited. This portion of P is a subpath connecting C_1 and C_2 having no common edged with either. We use C_1, C_2 and this subpath of P to create our even walk.

So far we have created ℓ even walks, but they might not be edge-disjoint. The odd cycles are edge-disjoint, as are the paths connecting them, but one of the paths might share an edge with an unrelated odd cycle. This also means that no edge e can be shared between more than two even walks; e can be used once as part of an odd cycle, and once as part of a path.

Now consider the number of edges in each even walk. If we sum this over all even walks, we get at most $2|E|$, since each edge can appear in at most two even walks. So the average number of edges in an even walk is $\leq 2|E|/\ell$. By Markov's inequality at least half of the even walks, $\ell/2$ of them, will have at most twice this average number of edges, $4|E|/\ell$. Now create a conflict graph where nodes represent these $\ell/2$ even walks, and an edge is placed when the two even walks share an edge. The degree of each node is at most $4|E|/\ell$. So if pick a maximal independent set in this conflict graph, it will consist of at least $\Omega(\ell^2/|E|)$ many even walks. ◀

We are finally ready to prove Lemma 34.

Proof of Lemma 34. First note that if our graph is not an isolated edge and is matching-covered it must contain at least one even cycle. This is so because there must be at least two perfect matchings in the graph, and in their symmetric difference, we can find one such cycle.

Because we are guaranteed to have at least 1 cycle, we can simply show that asymptotically we can extract $\Omega(|E|/\log^2|V|)$ edge-disjoint even walks. Then the asymptotic statement translates to the more concrete bound of $c_2|E|/\log^2|V|$.

If $(1-\epsilon)|E| \geq |V|$, by Lemma 35, we have at least $\epsilon|E|/2\log_2|V| = \Omega(|E|/\log|V|)$ cycles. If at least half of them are of even length, we are done. Otherwise we get $\Omega(|E|/\log|V|)$ odd cycles. Perhaps by throwing away one of them, we can assume the number of odd cycles we have is even. Then we can apply Lemma 37 to obtain $\Omega(|E|/\log^2|V|)$ edge-disjoint walks. This completes the proof. ◀

6 The Oracle-Based Algorithm

In this section we describe our oracle-based algorithm for finding a perfect matching. In Section 6.1, we will extend this to finding a *minimum weight* perfect matching for small weights.

On input $G = (V, E)$, our algorithm proceeds by finding smaller and smaller matching minors H of G, until H has a unique perfect matching, or in other words is a perfect matching. Then we pick the edges in H as a partial matching in G and extend this partial matching to a perfect matching independently and in parallel for the preimage of each node in H. That is for each node s in H, we take the set $S \subseteq V$ that got shrunk to s, remove the single endpoint of the partial matching from S, and recursively find a perfect matching in S. In the end we return the results of all these recursive calls along with the edges of H as the final answer.

We crucially make sure that the pre-image of nodes in H never contain more than a constant fraction of V. This makes sure that our recursive calls end in $O(\log|V|)$ steps.

In all of our algorithms, when we construct matching minors, we implicitly maintain the mapping from the resulting edges to the original edges, and the mapping from original vertices to the minor's vertices. These are trivial to maintain in NC, but for clarity we avoid explicitly mentioning them. We also keep node weights for matching minors, where the *weight of a node* is simply the number of original vertices that got shrunk to it.

Algorithm 2 Divide-and-conquer algorithm for finding a perfect matching.

```
PerfectMatching(G = (V, E))
if V = ∅ then
    return ∅.
else
    Call PartialMatching(G), and let H be the matching minor returned.
    Let M ⊆ E be the edges of H.
    for each node s of H in parallel do
        Let S ⊆ V be the nodes of G that are shrunk to s.
        Let v be the unique endpoint of the unique edge of M in δ(S).
        Let G_s be the induced graph on S − {v}.
        M ← M ∪ PerfectMatching(G_s).
    end
    return M.
end
```

The pseudocode for the main algorithm PerfectMatching can be seen in Algorithm 2. On input G, the algorithm calls PartialMatching to find a matching minor H of G which itself is a perfect matching. Then the edges of H, which form a partial matching in G, are extended to a perfect matching independently and in parallel in the preimage of each node from H. Since H is a matching minor, this extension can always be performed by Lemma 20.

The pseudocode for PartialMatching can be seen in Algorithm 3. This algorithm keeps a node-weighted matching minor of the input graph G. It tries several ways of obtaining a smaller matching minor, where *size of a matching minor* is measured in terms of the number

Algorithm 3 Find a matching minor of the input graph that is itself a perfect matching.

PartialMatching($G = (V, E)$)
Assign node weight 1 to each node $v \in V$.
while *G is not a perfect matching* **do**
  if *any node v of G has at least $1/6$ of the total node weight* **then**
    Remove disallowed edges $e \notin E[0]$ from G.
    Contract the complement of $\{v\}$ to a single node. If there are parallel edges, remove all except for an arbitrary one.
    return G.
  end
  Find a maximal set of node-disjoint triads in G.
  Let H be obtained from G by removing disallowed edges and contracting each triad into a single node.
  $U \leftarrow \{H\}$.
  for $w \in \mathcal{W}$ **in parallel do**
    Call Reduce(G, w) and let the result be H.
    $U \leftarrow U \cup \{H\}$.
  end
  Find the graph $H \in U$ with the minimum number of non-isolated edges.
  $G \leftarrow H$.
end
return G.

Algorithm 4 Remove disallowed edges and contract certain tight odd sets.

Reduce($G = (V, E)$, w) ; // The graph G has node weights.
Remove disallowed edges $e \notin E[w]$ from G.
Find all connected components of G.
for *each connected component C of G* **in parallel do**
  Run Algorithm 1 on C to find a laminar family of tight odd sets $\mathcal{L}$.
  for $S \in \mathcal{L}$ **in parallel do**
    if *node weight of S is more than half of the node weight of C* **then**
      Replace S in $\mathcal{L}$ with $C - S$.
    end
  end
  Find the inclusion-wise maximal sets in $\mathcal{L}$ and shrink each one to a single node.
end
return G.

of non-isolated edges, see Definition 13. One way of obtaining a smaller matching minor is by picking a maximal node-disjoint set of triads and shrinking them simultaneously. By Lemma 22, this produces a matching minor. Also note that the maximal set of node-disjoint triads can be found in NC by enumerating all triads and using Theorem 27.

Another way of obtaining smaller matching minors is by trying weights from the set of weight vectors $\mathcal{W}$ and calling Reduce to remove disallowed edges $e \notin E[w]$ and shrinking top-level sets of a laminar family of tight odd sets w.r.t. w.

Finally. the pseudocode for Reduce can be seen in Algorithm 4. This algorithm is simply fed a graph $G = (V, E)$ and a weight vector w. It removes disallowed edges $e \notin E[w]$ and shrinks the maximal sets of a laminar family of tight odd set. The laminar family is found using Algorithm 1, but is modified to make sure that no shrunk set becomes too large; to be more precise no shrunk vertex in the end will have node weight more than half of the total node weight.

6.1 Finding a minimum weight perfect matching

We extend our algorithm so it returns not just any perfect matching, but rather a minimum weight perfect matching, for small weights.

Given an input graph $G = (V, E)$ and a weight vector w, we can remove disallowed edges $e \notin E[w]$, and find a laminar family of tight odd sets $\mathcal{L}$ w.r.t. w, by calling Algorithm 1 on each connected component of G. By complementary slackness, any perfect matching that has only one edge in $\delta(S)$ for each $S \in \mathcal{L}$ will automatically be of minimum weight, see Definition 15. We can simply contract the top level sets in $\mathcal{L}$, use Algorithm 2 to find a perfect matching in the shrunk graph, and recursively extend this to a minimum weight perfect matching in each shrunk piece. Following an almost identical argument as in the proof of Proposition 18, the perfect matching in the shrunk graph can be extended to a minimum weight perfect matching.

The only problem with this method is that the recursion depth is not guaranteed to be polylogarithmic. However we can fix that by making sure that tight odd sets $S \in \mathcal{L}$ do not have more than half of the vertices in the graph; if they do, we replace them by their complements and we will see in Lemma 40 why this operation preserves laminarity.

6.2 Minor-closed families of graphs

Throughout our algorithm we only call the decision oracle on graphs obtained from the original through a sequence of edge and vertex removals and contractions. In this section we will prove that the decision oracle is only called on minors of the original graph, that is those graphs obtained by vertex and edge removals and contractions of *connected* subgraphs.

▶ **Lemma 38.** *Algorithms 2 to 4 call the decision oracle on minors of their input graph only.*

This lemma is all we need to prove Theorem 4. Note that there are several minor-closed families of graphs where the decision problem can be solved in NC by using a counting oracle. In particular we can count perfect matchings in graphs embedded on surfaces of genus at most $O(\log n)$, and therefore solve the decision problem, all in NC. This improves upon the genus bound of $O(\sqrt{\log n})$ given by [1].

▶ **Corollary 39.** *For graphs embedded on a surface of genus at most $O(\log n)$ and weighted with polynomially bounded edge weights, there is an NC algorithm to find a minimum weight perfect matching.*

Another consequence of Theorem 4 is an alternative algorithm for $K_{3,3}$-free graphs, which was resolved earlier by [7].

Now we prove Lemma 38.

Proof of Lemma 38. First we prove this for Algorithm 4. In this algorithm, we only remove edges from the input graph, and shrink tight odd sets in connected components. We just have to show that what we shrink is already connected. Consider a tight odd set S in a connected component C. If it is not internally connected, then one of its internal connected components must have odd size; let that be S'. Since $S \subseteq C$ and C is a connected component, there is an edge $e \in \delta(S - S')$. Since S' is not internally connected to $S - S'$, it must be that $e \in \delta(S)$ too. Now since the graph is matching-covered with minimum weight perfect matchings, there must be some minimum weight perfect matching $M \ni e$. But because S' is odd, there must also be an edge $f \in M \cap \delta(S')$. But note that $e \neq f$, and both $e, f \in \delta(S)$. This is a contradiction, since S cannot have more than one edge in a perfect matching. This shows that S must be connected and Algorithm 4 only produces minors of its input graph.

Next we prove the statement for Algorithm 3. This algorithm either calls Algorithm 4, or finds triads and contracts them. The former produces minors of the input graph, and the latter also produces minors of the input graph since triads are connected.

Note that the graph returned by Algorithm 3 may not be a proper minor of the input graph; that could happen if the node weight of some v goes above $1/6$ the total node weight. In this scenario, the complement of v might not be connected and yet we contract it. However the algorithm immediately returns and the decision oracle is not called on this returned graph. So this does not contradict the statement of the lemma.

Finally we prove the statement for Algorithm 2. The only graphs produced and passed onto Algorithm 3 are obtained from the input graph by vertex removals and edge removals. So they are all minors of the input graph. The output of Algorithm 3 might not be a proper minor, but this output is only used to decide which edges and vertices to remove from the original graph to get to induced graphs on $S - \{v\}$. ◀

7 Analysis of the algorithm

First we will prove that our oracle-based algorithm returns a correct answer. Next, we will bound the running time and prove that our algorithm runs in NC, modulo the calls to $\mathcal{O}$; this constitutes the most challenging part of the analysis.

7.1 Correctness

We will need the following lemma.

▶ **Lemma 40.** *Suppose that $\mathcal{L}$ is a laminar family of sets in a node-weighted graph $G = (V, E)$, and we replace every $S \in \mathcal{L}$ whose node weight is larger than half of the total node weight by the complement, i.e., $V - S$. Then the resulting family of sets $\mathcal{L}'$ is also laminar.*

Proof. Let S, S' be two sets in $\mathcal{L}$. They are either disjoint or one is contained in the other.

If $S \cap S' = \emptyset$: They cannot both have node weight more than $1/2$. So at most one of them gets replaced by its complement. Then it is easy to see that the resulting sets do not cross.

If $S \subseteq S'$: There are three possibilities. If none of them gets replaced by their complements, or both of them get replaced by their complements, they remain nested and therefore do not cross. If one of them gets replaced by its complement, it has to be the larger set S'. In that case the resulting sets become disjoint, and still do not cross. ◀

Using Lemma 40 and Lemma 20, we deduce that Reduce always returns a matching minor of its input graph. By definition, PartialMatching also returns a matching minor of its graph when it finishes (for the analysis of running time see Section 7.2).

This proves the correctness of the algorithm, since we always find a matching minor that has a unique perfect matching (itself), and by Lemma 20, we can extend it to a perfect matching, independently in the preimage of each node.

7.2 Running time

First we analyze PerfectMatching (Algorithm 2) assuming the calls to PartialMatching (Algorithm 3) are in NC.

▶ **Lemma 41.** *Assuming the calls to* PartialMatching *are in* NC*,* PerfectMatching *is in* NC*.*

Proof. We simply need to bound the number of levels in the recursion. We will prove that when PartialMatching returns a matching minor H, the node weight of every node is at most $5/6$ the total node weight. This proves that in each recursive call to PerfectMatching, the number of vertices gets reduced by a factor of $5/6$.

Note that the first time in Algorithm 3 that a node's weight goes above $1/6$ the total weight, the algorithm stops and returns a two-node minor. So we just need to prove that the weight of the node that just went above $1/6$ is not more than $5/6$. The current minor was obtained from the previous minor by either Reduce, or by shrinking triads. But Reduce never creates nodes with weight more than half the total weight. The weight of each node in a triad is also at most $1/6$ the total weight, so after shrinking the triad, the new weight can be at most $1/6 + 1/6 + 1/6 = 1/2$ the total weight. This finishes the proof. ◀

Finally, we need to prove that PartialMatching finishes in a polylogarithmic number of steps. Using the structural facts, Lemma 42 and Lemmas 32 and 34, we establish the following lemma.

▶ **Lemma 42.** *In each iteration of Algorithm 3, the number of non-isolated edges gets reduced by a factor of* $1 - \Omega(1/\log^2|V|)$*.*

Proof. First assume G is a connected graph. Then we can directly apply Lemmas 32 and 34 for some fixed $\epsilon < 1/9$ to show that we either find $c_1|E|$ triads or there exist $c_2|E|/\log^2|V|$ edge-disjoint even walks. In the former case, after contracting the triads, the number of edges gets reduced by a factor of $1 - c_1$. In the latter case, let $C_1, C_2, \ldots, C_k$ be the edge-disjoint even walks, and let $w \in \mathcal{W}$ be the weight vector such that $\langle w, \text{sign}(C_i)\rangle \neq 0$. Note that w is guaranteed to exist by Lemma 26. In the call to Reduce(G, w), every C_i loses at least edge by Lemmas 24 and 25, either because one of its edges becomes disallowed or it gets shrunk as a result of shrinking top-level tight odd sets. Therefore, one of the candidate graphs in U in Algorithm 3 will have a factor of $1 - c_3/\log^2|V|$ fewer edges, for some constant $c_3 > 0$.

Next assume G is not connected. If so, we apply the above-stated argument to each connected component that is not an isolated edge. We can further assume the same weight vector w works for all connected components. Now if H_1 is the graph obtained from shrinking triads, and H_2 is the result of Reduce(G, w), then we know that the average number of edges in H_1 and H_2 for each connected component is at most $1 - c_3/2\log^2|V|$ times the number of edges in the connected component. So one of H_1, H_2 must have at most $(1 - c_3/2\log^2|V|)$ times as many non-isolated edges as G. ◀

Note that Lemma 42 gives a polylogarithmic upper bound on the number of iterations in Algorithm 3, since if we track the number of non-isolated edges, after every $\Theta(\log^2|V|)$ steps we get a constant factor reduction, and therefore it takes at most $O(\log|E| \cdot \log^2|V|)$ iterations for it to reach 0.

8 Discussion

This paper has identified what appears to be the "core" of the difficult open problem of obtaining an NC matching algorithm, namely the decision problem. We must immediately mention that both decision problems stated in Section 1.2 have been the subject of numerous attacks over the past decades and hence resolution is not likely to be an easy matter. At the same time, we hope that since the "target" has been more precisely identified, the resolution of the open problem will gain added impetus.

An obvious open question is to build on the quasi-NC algorithm of [11] and related results of [12] to obtain the appropriate oracle-based NC algorithms and pseudo-deterministic RNC algorithms for linear matroid intersection and for finding a vertex of a polytope with faces given by totally unimodular constraints. An interesting problem defined by Papadimitriou and Yannakakis [29] , called Exact Matching, is the following: Given a graph G with a subset of the edges marked red and an integer k, find a perfect matching with exactly k red edges. This problem is known to be in RNC [27], even though it is not yet known to be in P. Is there a pseudo-deterministic RNC algorithm for it?

The phenomenon identified in Section 1.3 clearly deserves to be studied in depth. To the best of our knowledge, there are only two algorithmic results for bipartite matching that have not been extended to general graphs. The first is obtaining a fully polynomial randomized approximation scheme for counting the number of perfect matchings [14]; this is also among the outstanding open problems of theoretical computer science today. The second is obtaining an $O(m^{10/7})$ algorithm for maximum matching [23], which beats the earlier algorithms for sparse graphs.

References

1. Nima Anari and Vijay V. Vazirani. Planar Graph Perfect Matching is in NC. In *Proceedings of the 59th IEEE Annual Symposium on Foundations of Computer Science*. IEEE Computer Society, 2018.
2. Alberto Caprara, Alessandro Panconesi, and Romeo Rizzi. Packing cycles in undirected graphs. *Journal of Algorithms*, 48(1):239–256, 2003.
3. Laszlo Csanky. Fast parallel matrix inversion algorithms. *SIAM Journal on Computing*, 5(4):618–623, 1976.
4. Marek Cygan, Harold N Gabow, and Piotr Sankowski. Algorithmic Applications of Baur-Strassen's Theorem: Shortest Cycles, Diameter and Matchings. *arXiv preprint*, 2012. `arXiv:1204.1616`.
5. Samir Datta, Raghav Kulkarni, and Sambuddha Roy. Deterministically isolating a perfect matching in bipartite planar graphs. *Theory of Computing Systems*, 47(3):737–757, 2010.
6. Jack Edmonds. Maximum matching and a polyhedron with 0, 1-vertices. *Journal of research of the National Bureau of Standards B*, 69(125-130):55–56, 1965.
7. David Eppstein and Vijay V. Vazirani. NC Algorithms for Computing a Perfect Matching, Number of Perfect Matchings, and a Maximum Flow in One-Crossing-Minor-Free Graphs. In *Proceedings of the Thirty-First ACM Symposium on Parallelism in Algorithms and Architectures*, 2019.

8 Stephen Fenner, Rohit Gurjar, and Thomas Thierauf. Bipartite perfect matching is in quasi-NC. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 754–763. ACM, 2016.

9 Eran Gat and Shafi Goldwasser. Probabilistic Search Algorithms with Unique Answers and Their Cryptographic Applications. In *Electronic Colloquium on Computational Complexity (ECCC)*, volume 18, page 136, 2011.

10 Shafi Goldwasser and Ofer Grossman. Perfect Bipartite Matching in Pseudo-Deterministic RNC. In *Electronic Colloquium on Computational Complexity (ECCC)*, volume 22, page 208, 2015.

11 Rohit Gurjar and Thomas Thierauf. Linear matroid intersection is in quasi-NC. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 821–830. ACM, 2017.

12 Rohit Gurjar, Thomas Thierauf, and Nisheeth K Vishnoi. Isolating a vertex via lattices: Polytopes with totally unimodular faces. *arXiv preprint*, 2017. `arXiv:1708.02222`.

13 John E Hopcroft and Richard M Karp. An n^5/2 algorithm for maximum matchings in bipartite graphs. *SIAM Journal on computing*, 2(4):225–231, 1973.

14 Mark Jerrum, Alistair Sinclair, and Eric Vigoda. A polynomial-time approximation algorithm for the permanent of a matrix with nonnegative entries. *Journal of the ACM (JACM)*, 51(4):671–697, 2004.

15 Mark R Jerrum, Leslie G Valiant, and Vijay V Vazirani. Random generation of combinatorial structures from a uniform distribution. *Theoretical Computer Science*, 43:169–188, 1986.

16 Donald B Johnson. Parallel algorithms for minimum cuts and maximum flows in planar networks. *Journal of the ACM (JACM)*, 34(4):950–967, 1987.

17 Richard M Karp, Eli Upfal, and Avi Wigderson. Are search and decision programs computationally equivalent? In *Proceedings of the seventeenth annual ACM symposium on Theory of computing*, pages 464–475. ACM, 1985.

18 Richard M Karp, Eli Upfal, and Avi Wigderson. Constructing a perfect matching is in random NC. *Combinatorica*, 6(1):35–48, 1986.

19 Alexander V Karzanov. O nakhozhdenii maksimal'nogo potoka v setyakh spetsial'nogo vida i nekotorykh prilozheniyakh; title translation: On finding maximum flows in networks with special structure and some applications. *Mathematicheskie Voprosy Upravleniya Proizvodstvom*, 1973.

20 László Lovász. On determinants, matchings, and random algorithms. In *FCT*, volume 79, pages 565–574, 1979.

21 László Lovász and Michael D Plummer. *Matching theory*, volume 367. American Mathematical Soc., 2009.

22 Michael Luby. A simple parallel algorithm for the maximal independent set problem. *SIAM journal on computing*, 15(4):1036–1053, 1986.

23 Aleksander Madry. Navigating central path with electrical flows: From flows to matchings, and back. In *Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on*, pages 253–262. IEEE, 2013.

24 Meena Mahajan and Kasturi R Varadarajan. A new NC-algorithm for finding a perfect matching in bipartite planar and small genus graphs. In *Proceedings of the thirty-second annual ACM symposium on Theory of computing*, pages 351–357. ACM, 2000.

25 Silvio Micali and Vijay V Vazirani. An $O(\sqrt{|V|}|E|)$ algorithm for finding maximum matching in general graphs. In *Foundations of Computer Science, 1980., 21st Annual Symposium on*, pages 17–27, 1980.

26 Gary L Miller and Joseph Naor. Flow in planar graphs with multiple sources and sinks. In *Foundations of Computer Science, 1989., 30th Annual Symposium on*, pages 112–117. IEEE, 1989.

27 Ketan Mulmuley, Umesh V Vazirani, and Vijay V Vazirani. Matching is as easy as matrix inversion. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 345–354. ACM, 1987.

28 Manfred W Padberg and M Ram Rao. Odd minimum cut-sets and b-matchings. *Mathematics of Operations Research*, 7(1):67–80, 1982.

29 Christos H Papadimitriou and Mihalis Yannakakis. The complexity of restricted spanning tree problems. *Journal of the ACM (JACM)*, 29(2):285–309, 1982.

30 Michael O Rabin and Vijay V Vazirani. Maximum matchings in general graphs through randomization. *Journal of Algorithms*, 10(4):557–567, 1989.

31 Piotr Sankowski. NC Algorithms for Weighted Planar Perfect Matching and Related Problems. In *45th International Colloquium on Automata, Languages, and Programming (ICALP 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

32 Ola Svensson and Jakub Tarnawski. The matching problem in general graphs is in quasi-NC. In *Foundations of Computer Science (FOCS), 2017 IEEE 58th Annual Symposium on*, pages 696–707, 2017.

33 Leslie G Valiant. The complexity of computing the permanent. *Theoretical computer science*, 8(2):189–201, 1979.

34 Vijay V Vazirani. NC algorithms for computing the number of perfect matchings in K3, 3-free graphs and related problems. *Information and computation*, 80(2):152–164, 1989.

35 Vijay V Vazirani. A theory of alternating paths and blossoms for proving correctness of the $O(\sqrt{|V|}|E|)$ general graph maximum matching algorithm. *Combinatorica*, 14(1):71–109, 1994.

Advancing Subgroup Fairness via Sleeping Experts

Avrim Blum
Toyota Technological Institute at Chicago, IL, United States
avrim@ttic.edu

Thodoris Lykouris
Microsoft Research NYC, United States
thlykour@microsoft.edu

Abstract

We study methods for improving fairness to subgroups in settings with overlapping populations and sequential predictions. Classical notions of fairness focus on the balance of some property across different populations. However, in many applications the goal of the different groups is not to be predicted equally but rather to be predicted well. We demonstrate that the task of satisfying this guarantee for multiple overlapping groups is not straightforward and show that for the simple objective of unweighted average of false negative and false positive rate, satisfying this for overlapping populations can be statistically impossible even when we are provided predictors that perform well separately on each subgroup. On the positive side, we show that when individuals are equally important to the different groups they belong to, this goal is achievable; to do so, we draw a connection to the sleeping experts literature in online learning. Motivated by the one-sided feedback in natural settings of interest, we extend our results to such a feedback model. We also provide a game-theoretic interpretation of our results, examining the incentives of participants to join the system and to provide the system full information about predictors they may possess. We end with several interesting open problems concerning the strength of guarantees that can be achieved in a computationally efficient manner.

2012 ACM Subject Classification Theory of computation → Online learning algorithms

Keywords and phrases Online learning, Fairness, Game theory

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.55

Related Version A full version of the paper is available at `https://arxiv.org/abs/1909.08375`.

Funding *Avrim Blum*: The author was supported in part by NSF grants CCF-1815011 and CCF-1733556.
Thodoris Lykouris: Research initiated during the author's visit to TTI-Chicago while he was a Ph.D. student at Cornell University. The author was supported in part under NSF grant CCF-1563714 and a Google Ph.D. fellowship.

Acknowledgements The authors would like to thank Suriya Gunasekar for various useful discussions during the initial stages of this work and Manish Raghavan for offering comments in a preliminary version of the work.

1 Introduction

Concerns about ethical use of data in algorithmic decision-making have spawned an important conversation regarding machine learning techniques that are fair towards the affected populations. We focus here on binary decision-making: e.g., deciding whether to approve a loan, admit a student to an honors class, display a particular job advertisement, or prescribe a particular drug. While multiple fairness notions have been suggested to inform these decisions, most assume access to labeled data and that this data is drawn from i.i.d. distributions. In practice, data patterns are often dynamically evolving and the feedback received is biased by the decisions of the algorithms – such as only learning whether a student should have been

admitted to an honors class if the student is actually admitted – which induces additional misrepresentation of the actual data patterns. Despite the rich literature, approaching fairness considerations without these strong assumptions is rather underexplored.

Most fairness notions impose a requirement of balance across groups. For example, demographic parity [9] aims to ensure that the percentage of positive predictions is the same across different populations, and equality of opportunity [22] aims to ensure that the percentage of false negative predictions is the same across them. These notions are useful in identifying inequities between subpopulations, especially in settings such as criminal recividism [30] where there is a conflict between the incentives of the participants and the goal of accurate prediction. However, these notions may not be appropriate when the goal of each group is to be predicted as accurately as possible, and explicitly performing worse on one group in order to produce balance would be morally objectionable or absurd. For example, in a health application, a balance notion may lead to penalizing the majority population by willfully providing it worse treatment to make amends for the fact that a minority population is not classified correctly due to insufficient data. This would be clearly inappropriate.

More generally, there are many natural scenarios where the goal of each group is just to be predicted as accurately as possible. A student wishes to be admitted in an honors class only if they are qualified to succeed in it; otherwise their academic record may be jeopardized. A person requesting a microloan can be often significantly harmed by receiving it unless they return it (see [33] for an interesting discussion). A drug prescription is only beneficial if it helps enhance the health of the patient; otherwise it may cause adverse effects. In these settings what groups care about is not being treated *equally* but rather being treated *well*.

In this work, we consider a fairness notion for sequential non-i.i.d. settings where the subpopulations and the designer both strive for accurate predictions. We say that a prediction rule f is unfair to a subgroup g with respect to a family of rules $\mathcal{F}$ if there is some other rule $f_g \in \mathcal{F}$ that performs significantly better than f on g, in which case we say that f_g witnesses this unfairness. More generally, given a collection of rules, some of which may come from the designer, some from third-party entities, and some from the groups themselves, our goal is to achieve performance on each group comparable to that of the best of these rules for that group, even when groups overlap. Moreover, we aim to achieve this goal with as strong bounds as possible in a challenging "apple-tasting" feedback model where we only receive feedback on positive predictions (e.g., when a loan is given or a student is admitted). Interestingly, while our main results are positive, we show such guarantees are *not* possible if we replace the performance measure of accuracy (or error) with any fixed weighted average of false-positive and false-negative rates. Our positive results can also be thought of as a form of *individual rationality* with respect to the groups: no group has any incentive (up to low order terms) to pull out and, say, form its own lending agency just for members of that group. From this perspective, we also consider notions of *incentive compatibility* (could groups have any incentive to hide prediction rules from the system) and present a computationally-inefficient algorithm along with an open problem related to achieving this guarantee in a computationally-efficient manner.

1.1 Our contribution

We consider a decision-maker with some global decision function f_o that she would like to use (say to decide who gets a loan), and a collection of groups $\mathcal{G}$, where each group $g \in \mathcal{G}$ proposes some function f_g that *it* would like to be used instead on members of g.[1] The guarantee

[1] Both decision-maker and groups may have more functions; the guarantees are with respond to the best of them.

we aim to give is that our overall performance will be nearly as good (or better) than f_o on the entire population with respect to the objective function of the decision-maker, and for each group g our performance is nearly as good (or better) than the performance of f_g with respect to group g's objective. We would like to do this even when groups are overlapping and even when feedback on whether or not a decision was correct is only received when the decision made was "yes" (e.g., when the loan was given or the student was admitted).

Surprisingly, we show that when groups are overlapping and when the group objectives are to minimize the *unweighted* average of false-positive rate (FPR) and false-negative rate (FNR), there exist settings where *every* global prediction rule f must be unfair to one of the groups. In particular, we present a simple example with two overlapping groups having predictors f_1 and f_2 respectively, where performing nearly as well as f_1 on group 1 and nearly as well as f_2 on group 2 is fundamentally impossible when performance is measured as (FPR+FNR)/2 (or as max(FPR,FNR) or as any fixed non-degenerate weighting) even when the input does not arrive in an online manner. This shows that just having the fairness notions be the same across groups and aligned with the goals of the designer (who in this case does not have any additional goal other than to eliminate unfairness) is not by itself sufficient to be able to achieve our fairness criteria under objectives based on fixed combinations of FPR and FNR.

▶ **Informal theorem 1** (Theorem 10)**.** If subgroups are overlapping and their objectives are to minimize the unweighted average of false positive and false negative rates, there exist instances where no global function can simultaneously perform nearly as well on each group as the best function for that group. This holds even in the batch setting.

Instead we aim for low absolute error on each subgroup[2] and show a connection of this notion to an adversarial online learning setting, that of *sleeping experts* [6, 17]. In sleeping experts, each predictor (also referred to as an expert) can decide at each round to either make a prediction (fire) or abstain (sleep). Sleeping experts algorithms guarantee that, for any expert, the performance of the algorithm when the expert fires is nearly as good as the performance of this expert. Providing the functions f_o for the decision-maker and f_g for each group into existing sleeping-experts algorithms (viewing f_g as abstaining on any individual outside of group g) yields the following.

▶ **Informal theorem 2** (Theorem 1)**.** For the objective of minimizing absolute error, we can perform nearly as well as f_g for all groups g while performing nearly as well as f_o overall.

One particular complication that arises in many fairness-related settings, however, is that feedback received is one-sided: we only learn about the outcome if the loan is given, the student is admitted in the class, the advertisement is displayed, or the drug is prescribed; we do not learn about what would have happened when the action is not taken. In online learning, this is known as the *apple tasting* model [24]. We therefore initiate the study of sleeping experts in this apple tasting feedback model, aiming to achieve as strong regret guarantees as possible on a per-group basis. Combining apple tasting with sleeping experts poses interesting challenges as the exploration needs to be carefully coordinated among different subpopulations. In Section 3.2, we provide three different black-box reductions with different advantages in their performance guarantee.

[2] This is equivalent to FPR on a group weighted by the fraction of negative examples in that group plus FNR on a group is weighted by the fraction of positive examples in that group.

▶ **Informal theorem 3** (Theorems 3, 4, and 5)**.** Even if we only receive one-sided feedback, we can still perform nearly as well as f_g for all groups g while performing nearly as well as f_o overall.

Each of our guarantees is somewhat suboptimal. Theorem 3 is based on a construction that does not use sleeping experts, but has an exponential dependence on the number of groups which makes it computationally inefficient when there are many groups. Theorem 4 is a natural adaptation of sleeping experts to this setting but has an error bound with a (sublinear) dependence on the size of the total population instead of only depending on size of the subgroup, which makes the result less meaningful for small groups as this term may dominate their regret bound. Last, Theorem 5 has a more involved use of sleeping experts and does not suffer from the two previous issues, but has a suboptimal dependence on the size of the subgroup. Combining the advantages of these approaches without the resulting shortcomings is an intriguing open question.

The final contribution of our work (Section 4) is to provide a game-theoretic investigation of our setting in terms of incentives of the participating groups. In mechanism design, two important properties that a mechanism should satisfy are Individual Rationality (IR) and Incentive Compatibility (IC). The former asks that no player should prefer to opt out and seek service outside of the system instead. The latter refers to the mechanism creating no incentives for players to misreport their private information. Inspired by the kidney exchange literature [37, 3, 2], we consider each group as a player in our system. The IR property is satisfied when group g can get no benefit (asymptotically) from being predicted by their individual predictor f_g, say via their own loan agency. This is exactly what our above guarantees provide and therefore they can be interpreted as asymptotically IR. This observation brings up the question of whether incentive compatibility is also satisfied by sleeping experts algorithms. In this context, IC means that if a group g has a set of predictors $\{f_g\}$ then they get no benefit (asymptotically) from hiding some of those predictors from the decision-maker. Unfortunately, we show that current sleeping experts algorithms are *not* (even approximately) incentive compatible. On the other hand, we provide an algorithm that achieves both IR and IC guarantees, as well as operating in the apple tasting setting, at the expense of being computationally inefficient (enumerating over all exponentially-many group intersections). This leads to an interesting open question of finding a computationally efficient algorithm that satisfies both IR and IC properties.

▶ **Informal theorem 4** (Theorem 8)**.** Classical sleeping experts algorithms such as AdaNormalHedge do not satisfy the IC property.

▶ **Informal theorem 5** (Theorem 9)**.** Separate multiplicative weights algorithms for each intersection of groups satisfy the IC property at the expense of being computationally inefficient.

▶ **Open question 1** (Section 4.3)**.** Does there exist a computationally efficient algorithm satisfying both IR and IC properties?

1.2 Related work

There is a growing literature aiming to identify natural fairness notions and understand the limitations they impose; see [14, 22, 31, 12, 27, 1] for a non-exhaustive list. With respect to fairness among different demographic groups, notions such as disparate impact [9, 15] or equalized odds [22] aim to achieve some balance in performance across different populations. These make sense when there is an intrinsic conflict between the desires of the different groups

and those of the designer and can help identify undesirable inequities in the system that require remedies (that are often non-algorithmic and need policy changes). Unfortunately, aiming to satisfy these notions can also have undesired implications such as intentionally misclassifying some groups to make amends for the inability to classify other groups well enough. This has given rise to an important debate around alternative notions that do not suffer of these issues (see for example [13]). Our work aims to advance this direction.

When populations are overlapping, there are several recent works in the batch setting that tackle considerations similar to the ones we address. Kearns et al. [26] provide a simple example illustrating the issue that one can be non-discriminatory with respect to, say, gender and race in isolation but heavily discriminate against, say, black female participants; they also discuss how to audit whether a classifier exhibits such unfairness with respect to groups. In a similar motivation, Hebert-Johnson et al. [23] et al. suggest a fairness notion they term mutli-calibration that relates to accurate prediction of all populations that are *computationally identifiable*. Both these works show that their settings are equivalent to agnostic learning, which has strong computational hardness results but tends to have good heuristic guarantees.[3] Subgroup fairness is also discussed by Kim et al. [28, 29] for the related notion of multi-accuracy, with the aim of post-processing data to achieve this notion while maintaining overall accuracy [28] as well as metric subgroup fairness notions [29]. Our approach similarly considers overlapping demographic groups, but focuses on the more complex setting of online decision-making under limited feedback and addresses inherent conflicts between the incentives of different subgroups. In fact, even for the batch setting, our impossibility result of Section 5 with respect to the criterion of unweighted average of false positive and false negative rates is of a purely statistical flavor (has nothing to do with computational issues or arrivals being online). It therefore provides insights on complications that exist even when all the incentives seem well aligned. On the other hand, our connection to sleeping experts does not appear to directly have implications to multicalibration and multiaccuracy for similar reasons as the issues encountered regarding Incentive Compatibility; these notions tend to require a similar *no negative regret* property in order to be satisfied in an online context (similar to our results in Sections 4.1-4.2).

Fairness issues in online decision-making have also recently been considered. Joseph et al. [25] focus on a bandit learning approach and impose what they call *meritocratic fairness* against individuals with some stochastic but unknown quality. Celis et al. [10] discuss how to alleviate bias in settings like online advertising where bandit learning can lead to overexposure or underexpsure to particular actions. Blum et al. [7] focus on adversarial online learning and examine for different fairness notions whether non-discriminatory predictors can be combined efficiently in an online fashion while preserving non-discrimination. A recent line of work also focuses on online settings where decisions made today affect what is allowed tomorrow as they need to be connected via some metric-based notion of individual fairness [34, 19, 21]. Related to our work, Raghavan et al. [36] study externalities that arise in the exploration of classical stochastic bandit algorithms when applied across different subpopulations. Finally, Bechavod et al. [5] study a similar notion of one-sided feedback in online learning with fairness in mind. Unlike us, the latter work does not apply to overlapping populations but instead they take a contextual bandit approach, focus on stochastic and not adversarial losses, and aim for balance notions of fairness (where, as we discussed, there is conflict between incentives of designer and the groups).

[3] Their connection implies that for exact auditing, a linear dependence on the number of subpopulations is required but can be overcome if additional structure exists. Our work also has a linear dependence on the number of subgroups as it explicitly lists the subgroups. Understanding if this can be improved in our setting when the subgroups and predictors have additional structure is an interesting open direction.

Our work applies adversarial online learning which was initiated by the seminal works of Littlestone and Warmuth [32], and Freund and Schapire [16]. The classical experts guarantee we use in our first reduction can come from any of these or later developed algorithms. The apple tasting setting we consider to model the one-sided feedback was introduced by Helmbold et al. [24]; a related concept is that of label-efficient prediction that instead has an upper bound on the number of times the learner can explore [11]. Sleeping experts have been introduced by Blum [6] and Freund et al. [17]. Subsequently they were extended to a more general case of confidence-rated experts, and the results were better optimized [8, 18, 35]. The sleeping expert full feedback guarantee we use in our reductions is the one of AdaNormalHedge [35]. To the best of our knowledge, our work is the first to consider sleeping experts in the context of either apple tasting or label efficient prediction. We do so to enable our algorithms to deal with overlapping populations while only using realistic feedback (incorporating the one-sided nature of feedback).

2 Basic model

Online learning with group context

We consider an online learning setting with multiple overlapping demographic groups. The set of groups $\mathcal{G}$ can correspond to divisions based on gender, age, ethnicity, or other attributes. People (also referred to as examples) arrive sequentially, and the example at round $t = 1, 2, \ldots, T$ can be simultaneously a member of multiple groups (e.g., *female* and *hispanic*); the subset $\mathcal{G}_t \subseteq \mathcal{G}$ denotes all groups that person t belongs to.

At each round t, the system designer (or *learning algorithm* or *learner*) denoted by $\mathcal{A}$ aims to classify incoming examples by predicting a label $\hat{y}_t \in \hat{\mathcal{Y}}$. For example, in binary classication with deterministic predictors, the prediction space consists of positive and negative labels, i.e. $\hat{\mathcal{Y}} \in \{+, -\}$ (e.g. whether the corresponding person should be admitted to an honors class). To assist her goal, the designer has access to a set $\mathcal{F}_t$ of rules that suggest particular labels according to the features of the example; these are typically referred to as *experts* although they are not necessarily associated to any particular external knowledge. At every example, each expert $f \in \mathcal{F}_t$ makes a prediction $\hat{y}_{t,f} \in \hat{\mathcal{Y}}$ and the learner selects which expert's advice to follow in a (possibly randomized) manner; we use $p_{t,f}$ to denote the probability with which the learner follows the advice of expert f. Subsequently, the true label $y_t \in \mathcal{Y}$ is realized and each expert f is associated with a loss $\ell_{t,f} \in [0, 1]$. For example, if both the prediction and true label spaces are deterministic, i.e. $\hat{\mathcal{Y}} = \mathcal{Y} = \{+, -\}$, then a reasonable loss is whether the prediction was correct: $\ell_{t,f} = \mathbf{1}\{\hat{y}_{t,f} \neq y_t\}$. In order to not impose any i.i.d. assumption, we allow the losses to be adversarially selected. The learner then suffers expected loss $\hat{\ell}_t(\mathcal{A}) = \sum_{f \in \mathcal{F}_t} p_{t,f} \ell_{t,f}$ and observes some feedback (discussed below).

Feedback observed

In the first portion of this paper, we assume the learner receives *full feedback*, i.e. at the end of the round she observes the losses of all experts (this typically can be achieved by observing the label). In Section 3.2, we turn our attention to the apple tasting setting in which we only receive feedback about the losses when we select the positive outcome. To ease notation and streamline presentation, we present here the performance and fairness notions for full feedback and defer the apple tasting definitions to Section 3.2.

Overall regret guarantee

One natural goal for the designer in this setting is that it performs as well as the best among a class of experts $\mathcal{F}$ that are available every round, i.e. $\forall t : \mathcal{F} \subseteq \mathcal{F}_t$. Intuitively, when there exists a rule that consistently classifies examples more accurately, the learner should eventually realize this fact and trust it more often (or at least perform as well via combining other rules). This is formalized by the following notion of regret.

$$Reg = \sum_{t=1}^{T} \sum_{f \in \mathcal{F}_t} p_{t,f} \ell_{t,f} - \min_{f^\star \in \mathcal{F}} \sum_{t=1}^{T} \ell_{t,f^\star}. \tag{1}$$

In the classical expert setting, $\mathcal{F}_t = \mathcal{F}$ for all rounds t. Algorithms such as the celebrated multiplicative weights [16] incur regret that, on average, vanishes over time at a rate of $\sqrt{\log(|\mathcal{F}|)/T}$. Hence, despite the input not being i.i.d., the penalty we pay for not knowing in advance which expert is the best is very small and goes to 0 at a fast rate. We allow changes in the sets $\mathcal{F}_t$ to incorporate adaptive addition of rules as well as group-specific rules.

Subgroup regret guarantees

In order to not treat any group worse than what is inevitable, we are especially interested in group-based performance guarantees. In particular, for each group $g \in \mathcal{G}$, we want the performance on its members to be nearly as good as the best expert in a class $\mathcal{F}(g)$. This class can consist of all rules in $\mathcal{F}$, in which case this means we care not only about competing with the best rule in the class overall but also having the same guarantee for each group. It also allows each group to have rules specialized to it; for example, a third-party entity may observe a disparity in the performance of the group compared to what is achievable and recommend the use of a particular rule. To ease presentation we assume that the set $\mathcal{F}(g)$ is fixed in advance but most of our results extend to the case where new rules are added adaptively as potential unfairness is observed (see Remark 2). This notion of group-based regret is formalized below:

$$Reg(g) = \sum_{t:g \in \mathcal{G}_t} \sum_{f \in \mathcal{F}_t} p_{t,f} \ell_{t,f} - \min_{f^\star \in \mathcal{F}(g)} \sum_{t:g \in \mathcal{G}_t} \ell_{t,f^\star}. \tag{2}$$

In Section 3, we show that, via connecting to the literature of sleeping experts, we can have the average group-based regret vanish across time for all groups while still retaining the overall regret guarantee described above. In Section 5, we show that this is not in general possible for objectives based on fixed averages of false-positive and false-negative rates.

3 Sleeping experts and one-sided feedback

In this section, we focus on subgroup regret guarantees and conceptually connect the quest for these guarantees to the literature of sleeping experts and the incentives of the groups.

3.1 Subgroup regret guarantees via sleeping experts

Sleeping experts

Sleeping expert algorithms were originally developed to seamlessly combine task-specific rules so that their coexistence does not create negative externalities to other tasks. More formally, there is a set of experts $\mathcal{H}$ and, at each round t, any expert $h \in \mathcal{H}$ may decide

to either become active (fire) or abstain (sleep); the set of experts that fire at round t is denoted by $\mathcal{H}_t$. At any round, the algorithm can only select among active experts, i.e. puts non-zero probability $p_{t,h}$ only on experts $h \in \mathcal{H}_t$. The goal is that, for all experts $h^\star \in \mathcal{H}$, the performance of the algorithm in the rounds where the experts fire is at least as good as the one of $h^\star$. More formally the sleeping regret is:

$$\mathit{SleepReg}(h^\star) = \sum_{t:h^\star \in \mathcal{H}_t} \sum_{h \in \mathcal{H}} p_{t,h}\ell_{t,h} - \sum_{t:h^\star \in \mathcal{H}_t} \ell_{t,h^\star}. \tag{3}$$

The goal is to ensure that the average sleeping regret for any sleeping expert $h \in \mathcal{H}$ is vanishing with the number of times the expert fires, $T(h) = |t : h \in \mathcal{H}(t)|$, Multiple algorithms [6, 17, 8, 18, 35] achieve this goal. Probably the most effective among them is AdaNormalHedge by Luo and Schapire [35] with an average sleeping regret of $O\left(\sqrt{\log(|H|)/T(h)}\right)$. We elaborate on this algorithm in Section 4.1 in order to discuss its game-theoretic properties.

Subgroup regret formulated via sleeping experts

Looking closely at the definition of the desired subgroup regret from Eq. (2) and the one of sleeping regret from Eq. (3), there are clear similarities, which motivates formulating our problem as a sleeping experts problem and applying algorithms such as AdaNormalHedge. This leads to the following theorem.

▶ **Theorem 1.** *Let $\mathcal{A}$ be an algorithm with sleeping regret bounded by $\mathcal{O}\left(\sqrt{T(h) \cdot \log(|\mathcal{H}|)}\right)$ for any expert $h \in \mathcal{H}$ where $\mathcal{H}$ is any class. Let $\mathcal{G}$ be a set of overlapping groups and $N = |\mathcal{F}| + \sum_{g \in \mathcal{G}} |\mathcal{F}(g)|$. Then $\mathcal{A}$ can provide subgroup regret guarantee of $\mathcal{O}\left(\sqrt{T(g) \cdot \log(N)}\right)$ for any $g \in \mathcal{G}$ while ensuring overall regret guarantee of $\mathcal{O}(\sqrt{T \log(N)})$.*

Proof. The idea to bound subgroup regret as a sleeping experts problem is to have each expert $f \in \mathcal{F}(g)$ fire only for members of group g, guaranteeing that they experience performance at least as good as its own. One small issue is that we may want to use the same rule $f \in \mathcal{F}_t$ for multiple subgroups; to deal with that, we create different copies of this expert, each associated to one group.

More formally, we create a set of global sleeping experts $\mathcal{H}$ with one sleeping expert $h \in \mathcal{H}$ for every expert $f \in \mathcal{F}$. These sleeping experts fire every round and ensure the overall regret guarantee. Subsequently we create disjoint sets $\mathcal{H}(g)$ for each group $g \in \mathcal{G}$ where again we create a sleeping expert $h \in \mathcal{H}(g)$ for any expert $f \in \mathcal{F}(g)$. These sleeping experts fire only when the example is a member of group g and hence ensure the subgroup regret guarantee. The eventual sleeping expert set is $\bigcup_{g \in \mathcal{G}} \mathcal{H}(g) \cup \mathcal{H}$. The cardinality of this set is $N = |\mathcal{F}| + \sum_{g \in \mathcal{G}} |\mathcal{F}(g)|$. This formulation enables to automatically apply sleeping experts algorithms achieving subgroup regret of $\sqrt{T(g) \cdot \log(N)}$ for any group $g \in \mathcal{G}$ while also guaranteeing an overall regret of $\sqrt{T \cdot \log(N)}$. ◀

▶ Remark 2. In the above formulation, we assumed that all the experts exist in the beginning of the time-horizon. However, the sleeping experts algorithms allow for experts to be added dynamically over time (treating them as not firing in the initial rounds). Hence we can adaptively add new sleeping experts if a group or some third-party entity suggests it, guaranteeing that we do at least as well in the remainder of the game on the members of the group without affecting with the subgroup regret guarantees of other overlapping groups.

3.2 Subgroup regret with one-sided feedback

We now move our attention to the more realistic setting where we receive the label of the example (and therefore learn the losses of the experts) only if we select the positive outcome (e.g. admit the student to the honors class and observe her performance). This is captured by the so-called *apple tasting* setting which dates back to the work of Helmbold et al. [24].

Pay-for-feedback

Our algorithms opearate in a more challenging model where, at the beginning of each round, the learner needs to select whether to ask for the label. If she asks for it, she receives it at a cost of 1 instead of the loss of this outcome. Any guarantee for this setting is automatically an upper bound for the apple tasting setting. We can transform any such algorithm to an apple tasting one by selecting the positive outcomes at the rounds that we ask the label and ignoring any extra feedback. The loss of the positive outcome is upper bounded by 1; since we assume the losses to be bounded in $[0, 1]$, the loss in the pay-per feedback model is therefore only larger.

There are a few reasons why we want to work on this more stringent setting instead of the classical apple tasting setting. First, this feedback model makes it easy to create an estimator that is unbiased (since it does not condition on the prediction for the example and therefore our estimates do not suffer from selection bias). Second, in some applications, this model actually is more appropriate; for example, one may need to poll the participant to learn about their experience (which may be independent of whether they were classified as positive). Finally, this serves as an upper bound on the apple tasting setting; as we will see, arguing about lower bounds in this setting is significantly easier. Using the feedback in a more fine-grained way is an interesting open direction.

We now offer three guarantees for sleeping experts in the pay-per feedback setting, which are all achieved via black-box reductions to full-feedback algorithms (either sleeping experts or classical experts). Although the reductions become more involved as we proceed in the paper, no guarantee strictly dominates each other. Our algorithms select random points of exploration (when they ask for the labels to receive feedback). We denote by $\mathcal{E}$ the set of rounds that the algorithm ended up exploring (this is a random variable). For ease of presentation, we assume that we know the size of each demographic group and of all the intersections among groups; if these are not known, we can apply the so called doubling trick (similarly to the way described in Section 4.2).

First reduction: Independent classical experts algorithms per intersection

The first reduction we provide comes from treating all disjoint intersections between subgroups separately and running separate apple-tasting versions of classical (non-sleeping) experts algorithms on each intersection. Although this provides optimal dependence on the size of each subpopulation, the guarantee has an exponential dependence on the number of different groups.

For each disjoint intersection I between groups, let $T(I)$ be the size of this intersection and denote by $g \in I$ the case when intersection I includes g. Our algorithm splits the examples that lie in this intersection in $(T(I))^{2/3}$ phases, each of which consists of $(T(I))^{1/3}$ examples. At every phase, we select one random point of exploration. Whenever an example comes that belongs in I, our algorithm follows the advice of a classical experts algorithm (e.g. multiplicative weights) that is associated to I. This experts algorithm is updated at the end of the phase by the sample of the exploration round. This construction is in the spirit of Awerbuch and Mansour [4].

▶ **Theorem 3.** *Let $\mathcal{A}$ be an algorithm with regret bounded by $\mathcal{O}\sqrt{T \cdot \log(|\mathcal{H}|)}$ when compared to an expert class $\mathcal{H}$, run on T examples and split the examples in disjoint intersections, where each intersection corresponds to a distinct profile of subgroup memberships. For each intersection I, randomly selecting an exploration point every $(T(I))^{1/3}$ examples and running separate versions of $\mathcal{A}$ for each I provides subgroup regret on group $g \in \mathcal{G}$ of*

$$\mathcal{O}\left(\left(2^{|\mathcal{G}|}\right)^{1/3} \cdot (T(g))^{2/3} \cdot \sqrt{\log(N)}\right)$$

where $T(g) = |t : g \in \mathcal{G}(t)|$ is the size of the g-population and $N = |\mathcal{F}| + \sum_{g \in \mathcal{G}} |\mathcal{F}(g)|$.

Proof sketch. The guarantee follows from three observations formalized in Appendix A.

1. Among the exploration points, we run a classical experts algorithm so, on these examples, we have a regret guarantee that is square-root of the number of these examples.
2. For each phase, the exploration point is randomly selected and therefore the regret that we incur in the exploration point is an unbiased estimator of the average regret we incur in the whole phase (since the distribution of the algorithm in the phase is the same). As a result, the total regret in a phase is in expectation $(T(I))^{1/3}$ times the regret at the exploration point.
3. A particular group can have examples in at most $2^{|G|}$ intersections (as this is all the possible membership relationships with respect to the demographic groups). ◀

Second reduction: Sleeping experts with fixed exploration phases

Aiming to avoid the exponential dependence on the number of groups, we now apply a sleeping experts algorithm such as AdaNormalHedge as our base algorithm. The algorithm described in this part removes this exponential dependence but introduces a dependence on the time horizon and therefore the regret guarantee can be suboptimal for minority populations whose size is significantly smaller than the total population. On the other hand, when all populations are well represented (and are of the same order as the time-horizon) then the guarantee has the optimal dependence on the size of the population without suffering in the number of groups.

The algorithm splits the examples in $T^{2/3}$ phases and selects one random point in each of the phases. Each phase consists in total of $T^{1/3}$ examples but its examples can be distributed differently across it. At the end of the phase, we update a sleeping experts algorithm (e.g. AdaNormalHedge) based on the observations at the exploration point.

▶ **Theorem 4.** *Let $\mathcal{A}$ be an algorithm with sleeping regret bounded by $\mathcal{O}\left(\sqrt{T(h) \cdot \log(|\mathcal{H}|)}\right)$ for any expert h in class $\mathcal{H}$. Randomly selecting an exploration point every $T^{1/3}$ examples (irrespective of what groups they come from) and running $\mathcal{A}$ on these points provides subgroup regret on group $g \in \mathcal{G}$ of*

$$\mathcal{O}\left(T^{1/6} \cdot (T(g))^{1/2} \cdot \sqrt{\log(N)}\right)$$

where $T(g) = |t : g \in \mathcal{G}(t)|$ is the size of the group-g population and $N = |\mathcal{F}| + \sum_{g \in \mathcal{G}} |\mathcal{F}(g)|$.

Proof sketch. The guarantee follows from two observations formalized in Appendix B.

1. Given that we run a sleeping experts algorithm across the exploration points, if we just focused on those examples, we simultaneously satisfy regret on them that is square-root of their size.

2. Within any phase the exploration point is selected uniformly at random. As a result, it is an unbiased estimator of the average regret we incur in the whole phase. Note that this is now the average across all rounds and not only rounds where we have members of the particular group which results in the dependence on the time-horizon T. ◀

Third reduction: Sleeping experts with adaptive exploration phases

Our final reduction aims to remove the dependence on the time horizon while also avoiding an exponential dependence on the number of groups. Towards this end, we make the size of the phases adaptive based on the sizes of the populations. Our guarantee has both the aforementioned desired properties. On the negative side, the exponent on the group size is suboptimal (see discussion in the end of the section).

We again use a sleeping experts algorithm $\mathcal{A}$ across phases but phases are now designed adaptively. At the beginning of each phase r, we initialize a counter per group to capture the number of examples we have seen from it in phase r. When an example arrives, we increase the corresponding counters for all groups related to the example (possibly the example belongs in multiple groups; then we increase all the corresponding counters simultaneously). The phase ends when one of the groups $g \in \mathcal{G}$ has received $(T(g))^{1/4}$ examples in this phase.

At the beginning of a phase r, we draw for each group $g \in \mathcal{G}$ a uniform random number $X(g,r) \in \{1, 2, \ldots, T(g)\}$. This determines the exploration round for group g at phase r; let $t(r,g)$ be the random variable determining the time that the $X(g,r)$-th example (after time τ_r) from group g will arrive. If the phase ends before this example arrives, i.e. $t(r,g) > \tau_r$ then we associate this phase with 0 estimated losses for group g: $\tilde{\ell}_f(r,g) = 0$ for all $f \in \mathcal{F}(g)$. Otherwise, the estimated loss corresponding to the phase is the loss at the exploration point, i.e. $\tilde{\ell}_f(r,g) = \ell_{t(r,g),f}$. Since the phase ends once any group counter reaches the upper bound on examples for its phase, this means that we may not reach the exploration point for some other groups (this is the reason why we may have $t(r,g) > \tau_{r+1}$). Before proceeding to the next phase, we feed the estimated losses $\tilde{\ell}(r,g)$ to $\mathcal{A}$.

▶ **Theorem 5.** *Applying the above algorithm provides subgroup regret on group $g \in \mathcal{G}$ of*

$$\mathcal{O}\left(|\mathcal{G}| \cdot (T(g))^{3/4} \cdot \sqrt{\log(N)}\right)$$

where $T(g) = |t : g \in \mathcal{G}(t)|$ is the size of the g-population and $N = |\mathcal{F}| + \sum_{g \in \mathcal{G}} |\mathcal{F}(g)|$.

Proof sketch. There are three components to prove this guarantee, formalized in Appendix C.

1. The number for relevant phases of each group (phases where they have at least one example) is at most $T(g)$. This provides an upper bound on the number of phases that we need to consider with respect to group g.
2. Using a similar analysis as before, we can create a guarantee about the regret we are incurring in the exploration points and multiply it by the $(T(g))^{1/4}$ which is the size of the phase. This would have created a completely unbiased estimator if there was no overlap with other groups.
3. A final complication is that exploration may occur due to other groups so we need to understand how much we lose there. For that, we observe that smaller groups are explored with higher probability (the interaction with larger groups does not therefore significantly increase their probability of inspection). On the other hand, larger groups do not often collide with significantly smaller groups due to the latters' size. ◀

On the optimality of the bounds

In the pay-for-feedback model, even if we do not work in a sleeping experts setting, the best that one can hope for is a guarantee of $T^{2/3}$. If we explore T^a examples, we obtain a regret of $T^{a/2}$ on them. If the estimator is unbiased, we need to then multiply this with T^{1-a}, which gives a regret of $T^{1-a/2}$. Since we pay for feedback, we also lose T^a. The maximum of the two is minimized when $a = 2$. As a result, even if the groups were disjoint, we cannot hope for a better subgroup regret than $(T(g))^{2/3}$. Note that our results do not quite achieve this bound: having either a multiplicative term exponential in the number of groups (Theorem 3) or having a portion of the regret bound be in terms of the total time T rather than $T(g)$ (Theorem 4). Achieving a bound of $(T(g))^{2/3}$ without enumerating over all possible disjoint intersections across groups is therefore an interesting open direction.

4 A game-theoretic interpretation

In this section, we provide a game-theoretic interpretation of the above subgroup regret guarantee, connecting it to the notions of Individually Rationality (IR) and Incentive Compatibility (IC).

Individual Rationality

In game theory, a mechanism is considered Individually Rational when the participants prefer to stay and be served in the system rather than to leave and use their best outside option. Consider each subgroup as a player in a game and the cost experienced by this player as the total loss in all its members; e.g., imagine each group has a representative who looks out for their best interests. This representative has access to the rules in $\mathcal{F}(g)$ (private type) and can opt to defect from the global learning system and create its own predictor with only rules in $\mathcal{F}(g)$.

We say that a learning algorithm induces an IR mechanism if no group has significant incentive to opt out. (We cannot require zero incentive since the system needs some time to learn.)

The subgroup regret guarantee can be thought as an asymptotic version of the individual rationality property. The guarantee ensures that the average benefit from being served outside of the system vanishes as the group size grows as formalized below.

▶ **Definition 6.** *A learning algorithm induces an asympotically individual rational (IR) mechanism if no group gains (asymptotically) by getting served outside of the system.*

$$\forall g \in \mathcal{G} : \sum_{t:g\in\mathcal{G}_t} \sum_{f\in\mathcal{F}_t} p_{t,f}\ell_{t,f} - \min_{f^\star\in\mathcal{F}(g)} \sum_{t:g\in\mathcal{G}_t} \ell_{t,f^\star} = o(T(g)).$$

Incentive Compatibility

A second desired game-theoretic notion is that of Incentive Compatibility which states that the player has no incentive to misreport her true type. In our context, we define the type of a group to be the set $\mathcal{F}(g)$ of experts that it knows about, and IC means that group g could not achieve enhanced performance for the group by hiding a subset of $\mathcal{F}(g)$ and removing it from the global learning process. Recall that $\hat{\ell}_t(\mathcal{A}) = \sum_{f\in\mathcal{F}(t)} p_f^t \ell_f^t$ denote the expected loss of the algorithm $\mathcal{A}$ at round t.

We say that a learning algorithm induces an IC mechanism if removing a subset of group-based experts does not improve the performance of the group. More formally, let $\mathcal{A}(\emptyset)$ denote the algorithm running with all the experts and nothing removed and $\mathcal{A}(\{g, H\})$ denote the algorithm running with the subset $H \in \mathcal{F}(g)$ removed from the group-based experts and also from the overall set $\mathcal{F}$. Then:

▶ **Definition 7.** *A learning algorithm $\mathcal{A}$ induces an asympotically incentive compatible (IC) mechanism if no group gains (asymptotically) by hiding a subset of its experts.*

$$\forall g \in \mathcal{G}, \forall H \in \mathcal{F}(g) : \sum_{t:g\in\mathcal{G}(t)} \hat{\ell}_t(\mathcal{A}(\emptyset)) - \sum_{t:g\in\mathcal{G}(t)} \hat{\ell}_t(\mathcal{A}(\{g, H\})) = o(T(g)).$$

As before, we cannot hope to achieve the IC property exactly as extra experts will definitely delay the learning process, but we would like to satisfy an approximate version of this property where the average benefit from removing some experts vanishes as the group size grows. This is a desirable property as, when satisfied, it suggests that the groups should not overly think about potential adverse effects of suggesting particular predictors but instead provide all their proposed rules.

4.1 Roadblocks in applying sleeping experts for Incentive Compatibility

In this section, we show a strong negative result about classical sleeping experts algorithms with respect to the IC property. To make this formal, we present the result for the AdaNormalHedge algorithm of Luo and Schapire [35] but a similar intuition carries over to other sleeping experts algorithms (see Section 4.3).

AdaNormalHedge

AdaNormalHedge [35] is an algorithm with strong adaptive regret guarantees. Its sleeping experts version starts with a set of experts with cardinality N and a prior distribution q that is typically initialized uniformly: $q_i = 1/N$ for all $i \in [N]$. Every expert keeps two quantities $R_{t,i}$ capturing the total regret it has experienced so far in the rounds that it fired and $C_{t,i}$ capturing the desired regret guarantee. These parameters determine the weight of its expert which is expresssed using a potential function $\Phi(R, C) = \exp(\frac{\max(0,R)^2}{3C})$ giving rise to a weight function:

$$w(R, C) = \frac{1}{2}(\Phi(R + 1, C + 1) - \Phi(R - 1, C + 1).$$

More formally, both the expert quantities are initialized to 0, i.e. $R_{0,i} = C_{0,i} = 0$. At round $t = 1 \dots T$, a set $A(t)$ of experts is activated and the learner predicts with probability proportional to the weight of its firing expert: $p_{t,i} \propto q_i \cdot w(R_{t-1,i}, C_{t-1,i}) \cdot \mathbf{1}\{i \in F(t)\}$.[4] The adversary then reveals the loss vector ℓ_t and the learner suffers loss $\hat{\ell}_t = \sum_{i\in[N]} p_{t,i}\ell_{t,i}$. This gives rise to an instantaneous regret for each firing expert: $r_{t,i} = \left(\hat{\ell}_t - \ell_{t,i}\right) \cdot \mathbf{1}\{i \in A(t)\}$ which is used to update the expert parameters: $R_{t,i} = R_{t-1,i} + r_{t,i}$ and $C_{t,i} = C_{t-1,i} + |r_{t,i}|$, before proceeding to the next round.

The regret of this algorithm with respect to an expert $i \in [N]$ is roughly of order $\sqrt{C_{T,i}}$ which in the sleeping experts version gives a sleeping regret $\sqrt{|t : i \in A(t)| \log(N)}$. This is why using such an algorithm for subgroup regret guarantees provides a guarantee of $\sqrt{T(g) \log(N)}$, ignoring constants.

[4] Luo and Schapire [35] predict arbitrarily if all weights are 0; we commit on selecting uniformly at random then.

The IC lower bound

Although the subgroup regret guarantee that sleeping experts provide makes them satisfy an asymptotic version of the IR property, we now show that this is not the case for the IC property; we illustrate this for AdaNormalHedge and further discuss it in Section 4.3.

▶ **Theorem 8.** *AdaNormalHedge does not induce an asymptotically incentive compatible mechanism, i.e. there exists an instance where a group can asymptotically benefit from hiding one of its experts.*

Proof. Consider a setting with two groups where the bigger group B consists of the whole population whereas the smaller group S corresponds to half of the population. Every odd round an example from S arrives and every even round an example from $B \setminus S$ arrives. The algorithm has access to one global expert $\mathcal{F} = \{f\}$ as well as one group-specific expert per group: $\mathcal{F}(g) = \{f(g)\}$ for $g \in \{B, S\}$.

- Both group-specific experts have always loss $\ell_{t,f(g)} = 0.2$ if $g \in \mathcal{G}(t)$.
- The global expert is really bad at predicting group S: $\ell_{t,f} = 1$ if $g \in S$ but makes no mistakes on the remaining population: $\ell_{t,f} = 0$ if $g \in B \setminus S$.

The high-level idea on why IC does not hold is the following. Group B prefers to use expert $f(S)$ on members of group S and expert f on $B \setminus S$; this is achieved if it hides expert $f(B)$. However, if it does not, AdaNormalHedge ends up using often expert $f(B)$ on $B \setminus S$ which leads to a much higher loss. Intuitively, since $f(B)$ and f make predictions on exactly the same set of examples and $f(B)$ has lower total loss, then $f(B)$ gets much higher weight than f — the algorithm therefore uses $f(B)$ instead of f on $B \setminus S$.

More formally, suppose first that group B hides expert $f(B)$. The sleeping regret guarantee for expert $f(S)$ guarantees that expert $f(S)$ is selected in all but a vanishingly small number of rounds. As a result, asymptotically, the loss accumulated from group S is $0.2 \cdot (T/2)$. Given that $f(B)$ is not an option (as it is hidden), in members $B \setminus S$ the algorithm can only select expert f incurring 0 loss.

We now show that, if $f(B)$ is not hidden, it incurs more loss in members of $B \setminus S$ without gaining on S. In this case, we show below that expert f is selected with probability $p_{t,f} \leq 1/2$ after a few initial rounds. This leads to an additional loss of at least $0.2 \cdot (T/4)$ on examples of $B \setminus S$ since we select expert $f(B)$ which has higher loss than f on these examples in at least half of the even rounds in expectation. As a result, by not hiding expert $f(B)$, the algorithm selects it often which leads in an additional loss that is linear in T – this directly implies that group B is better off by hiding this expert and enhancing the overall performance.

What is left is to show why the probability of selecting expert f is indeed $p_{f,t} \leq 1/2$ after a few inital steps. The sleeping regret guarantee for expert $f(S)$ implies that the cumulative (across rounds) probability of selecting expert f on examples in S until round t is at most $\sqrt{t}$ up to constants and log factors. As a result, the expected loss of the algorithm until round t is at most $\hat{\ell}_t \leq 0.2 \cdot t/2 + \sqrt{t}$ and the total instantaneous regret of experts f and $f(B)$ on odd rounds (i.e., members of group S) is

$$\sum_{\substack{\tau \leq t \\ \tau \text{ is odd}}} r_{\tau,f} \geq 0.8 \cdot t/2 - \sqrt{t} \qquad \text{and} \qquad \sum_{\substack{\tau \leq t \\ \tau \text{ is odd}}} r_{\tau,f(B)} \leq \sqrt{t} \qquad \text{respectively.}$$

On even rounds (i.e., members of $B \setminus S$), the algorithm selects either f or $f(B)$; therefore its loss is at most 0.2. This means that the instantaneous regret on these rounds is:

$$\sum_{\substack{\tau \leq t \\ \tau \text{ is even}}} r_{\tau,f} \geq -0.2 \cdot t/2 \qquad \text{and} \qquad \sum_{\substack{\tau \leq t \\ \tau \text{ is even}}} r_{\tau,f(B)} \leq 0.2 \cdot t/2 \qquad \text{respectively.}$$

As a result, after a few initial rounds, the cumulative instantaneous regret of f is consistently negative and also smaller than the one of $f(B)$. This means that, by the construction of the potential function, the weight of f is 0 as $\Phi(R_{f,t+1}+1, C+1) = \Phi(R_{f,t+1}-1, C+1) = 1$. Since Φ is an increasing function on its first argument and the probability is proportional on the weight, this means that f has the smallest probability across all other experts who fire and therefore has probability of being selected at most $1/2$ which is what we wanted to show. Note that, when $f(B)$ is hidden, the weight still becomes 0 but now f is the sole alternative and is therefore always selected in members of $B \setminus S$. ◀

The reason why this negative result arises is that, by hiding a subset of the experts, the group can potentially lead the algorithm to use different experts in different disjoint parts of the population. Sleeping experts algorithms penalize each expert for its overall performance at times when it fires and does not distinguish disjoint subpopulations where it performs much better.

4.2 Incentive Compatibility in a computationally inefficient way

Multiplicative weights for each disjoint subgroup

We now turn our attention to the use of separate algorithms for each disjoint intersection of groups. This suffers from an exponential dependence on the number of groups but satisfies both IR and IC. In some sense, it therefore serves as an existential proof that satisfying these quantities in an online manner is feasible and creates an intriguing open question of whether this can be achieved in a computationally efficient manner.

We run separate multiplicative weights algorithms for each disjoint group intersection. More formally, for every $S \in 2^{|\mathcal{G}|}$, we run sub-algorithm $\mathcal{A}(S)$ on examples t where $g \in S$ if and only if $g \in \mathcal{G}_t$. We denote by $\mathcal{A}_t$ this sub-algorithm. We assume that experts are not added adaptively for this part (if a new expert appears then we reinitialize the algorithm). The sub-algorithm $\mathcal{A}(S)$ has as experts all experts that fire at examples associated with S so all members of $\mathcal{F}$ or $\mathcal{F}(g)$ for some $g \in S$. We let $N(\mathcal{A})$ denote the number of those experts for ub-algorithm $\mathcal{A}$.

For sub-algorithm $\mathcal{A}$, each expert i is initialized with weight $w_{t,i}(\mathcal{A}) = 1$. The algorithm selects experts proportionally to the weights in the corresponding sub-algorithm, i.e. $p_{t,i} = \frac{w_{t,i}(\mathcal{A}_t)}{\sum_{j \in \mathcal{F}(S)} w_{t,j}(\mathcal{A}_t)}$. We then multiplicatively update weights with learning rate η only for $\mathcal{A}_t$: $w_{t+1,i}(\mathcal{A}) = w_{t,i} \cdot (1-\eta)^{\ell_{t,i} \cdot \mathbf{1}\{\mathcal{A}=\mathcal{A}_t\}}$. Denoting by $T(\mathcal{A})$ the size of the disjoint subgroup, we should let $\eta = \sqrt{\log(N(\mathcal{A}))/T(\mathcal{A})}$ for a guarantee sublinear to this size. To deal with the fact that we do not know the size of each disjoint subgroup in advance we apply the so called doubling trick, assuming that it is 2^r (initial $r = 2$) and reinitializing the algorithm by increasing r – this can happen at most $\log(T)$ times.

This algorithm satisfies the vanishing subgroup regret property (thus also the asymptotic IR property) as any group consists of multiple disjoint subgroups and the group has vanishing regret within each of them; other than the regret term, it cannot hope to do better if it is served outside of the system with its own functions. We now show that these separate multiplicative weights algorithms also achieve the asymptotic IC property. For that, we use a nice property of multiplicative weights establishing that multiplicative weights not only does not do worse than the best expert, but it also does not do better. This was first formalized by Gofer and Mansour [20] and was used in a fairness context by Blum et al. [7] to establish that equality of average losses across different groups is preserved when combining experts that have equally good performance across these groups.

▶ **Theorem 9.** *Running separate multiplicative weights algorithms for each disjoint intersection among groups induces an asymptotically IC mechanism, i.e. no group can asymptotically benefit by hiding any of its experts.*

Proof. One essential step of the proof is the property that multiplicative weights with a fixed learning rate has performance almost equal to the one of the best expert. This is exactly shown in the proof of Theorem 3 in [7] which establishes that for any sub-algorithm and any fixed learning rate, the performance of the sub-algorithm is equal to the performance of the best expert $L^\star$ in these rounds plus, up to constants, $\eta \cdot L^\star + \log(N(\mathcal{A}))/\eta$. Since we run each sub-algorithm with a fixed learning rate for at most 2^r rounds, this is, up to constants less than $\sqrt{2^r \log(N(\mathcal{A}))}$. Summing across all $r = 2 \dots T(\mathcal{A})$, this is at most $\sqrt{T(\mathcal{A}) \log(N(\mathcal{A}))}$.

To now establish the asymptotic IC property, note that the intervals that have fixed learning rates are not affected by any groups' decision to hide a subset of their predictors. As a result, if some predictor is removed, then multiplicative weights will then have performance (asymptotically) close to the one of the best expert without the hidden one; this performance can only be worse (up to the regret term). This holds for any disjoint subgroup; it therefore still holds summing across all the (possibly exponential) disjoint subgroups. ◀

4.3 Open problem: Computationally efficient Incentive Compatibility

In Section 4.1, we showed that AdaNormalHedge does not induce an IC mechanism as it used a single weight that deteriorated significantly in some disjoing subgroups and made some experts not usable in places that were essential and where hiding some experts would help with that. Using a single weight for each sleeping expert and updating it based on its instantaneous regret is not unique to AdaNormalHedge but is present to other algorithms such as the one of Blum and Mansour [8]. Since the algorithms do not distinguish between instantaneous regret obtained by different disjoint subgroups, the same construction can extend to those as well. On the other hand, in Section 4.2, we also showed that separate multiplicative weights for each subgroup provide the asymptotic IC property as they have the nice property that the performance of the algorithm is exactly as good as the one of the best expert in this group. On the negative side, having one sub-algorithm per disjoint subgroup is computationally costly as it means that we have exponential number of sub-algorithms.

These two results lead to the following open question: can we design a computationally efficient algorithm (not enumerating over all disjoint subgroups) that satisfies the subgroup regret guarantee? Besides its application to subgroup fairness, this question has independent technical interest as it seems to require a fundamentally new approach for the sleeping experts problem.

5 Impossibility results for fixed averages of FNR and FPR

In this section we demonstrate that if rather than minimizing the *fraction of errors*, each group wishes to minimize the unweighted average of its False Negative Rate (FNR) and False Positive Rate (FPR), or any fixed nontrivial weighted average of these two quantities, then guarantees of the form given in earlier sections are intrinsically not possible when there is no zero-error predictor.[5]

[5] Minimizing just FNR (resp. just FPR) is trivial by predicting positive (resp. negative) on every example.

False negative and false positive rates

Many fairness notions explicitly consider false negative and false positive rates. For example, the notion of Equalized Odds [22] imposes that both these rates should be the same for all groups of interest, and the notion of Equality of Opportunity [22] imposes equality for just the FNR. Given the attention paid to false negative and false positive rates, it is natural to consider the objective of minimizing their average (or minimizing their maximum, which our negative results will also apply to). More formally, under this objective, the loss of the algorithm corresponds to $\frac{1}{|t:y_t=+|}\sum_{t:y_t=+}\hat{\ell}_t + \frac{1}{|t:y_t=-|}\sum_{t:y_t=-}\hat{\ell}_t$. If we have just one group and aim to achieve performance compared to the one of the best expert $f^\star \in \mathcal{F}(g)$ for this objective, i.e. $\frac{1}{|t:y_t=+|}\sum_{t:y_t=+}\ell_{t,f^\star} + \frac{1}{|t:y_t=-|}\sum_{t:y_t=-}\ell_{t,f^\star}$, this is feasible even in an online setting by running a no-regret algorithm and weighting each example based on the number of examples with the same label that exist. Also note that, unlike balance notions, *if groups are disjoint* then simultaneously optimizing this objective for both groups does not create some inherent conflict between groups, e.g. necessitating to have performance on some group that is worse than necessary.

Unfortunately, we show that while minimizing the unweighted average of FNR and FPR seems like a benign objective, it can be impossible to produce a global prediction strategy that, on each group, performs nearly as well as the best predictor given for that group under this objective. That is, even though all groups appear to have objectives that are "aligned" they may not be simultaneously satisfiable. This impossibility result holds due to purely statistical reasons even in the batch setting and has nothing to do with the online or non-i.i.d. nature of examples. Note that if the populations are not overlapping, the batch setting for this goal is straightforward: among the available classifiers, for each group select the one with the best performance according to the given objective. The following theorem shows that, with overlapping populations, this is no longer possible.

▶ **Theorem 10.** *Consider overlapping groups wishing to achieve performance on their members comparable to their group-specific predictors with respect to the objective of unweighted average of false positive and false negative rates. Even in the batch setting, there exist settings where it is impossible to simultaneously achieve this goal for both groups.*

Proof. Assume two groups $\mathcal{G} = \{A, B\}$ that have 80% of their examples disjoint from the other group and the remainder 20% in common, as illustrated in Figure 1. All the non-overlapping examples of group A have positive label and the non-overlapping examples of group B have negative label. With respect to the shared portion, the examples have labels that are uniformly distributed: conditioned on being in the intersection, the probability of the label being positive is 0.5.

Regarding the predictors, we have two predictors $\mathcal{F} = \{f_A, f_B\}$. Predictor f_A correctly predicts positive in the non-overlapping examples $(A \setminus B)$, and predicts negative in the overlapping part $A \cap B$. The false positive rate of this predictor is 0 and the false negative rate is 1/9. Analogously, predictor f_B predicts negative on the examples in $B \setminus A$ and predicts positive in $A \cap B$, resulting in a false positive rate on examples in B of 1/9 and a false negative rate of 0. Therefore, for both $g \in \{A, B\}$, f_g has a generalization error of 1/18 with respect to the unweighted average of false positive and false negative rates.

We now show that this performance cannot (even approximately) be simultaneously achieved for both groups by combining these two predictors even in the batch setting. In particular, since examples in $A \cap B$ all have uniformly random labels, we can only select some probability p to predict a positive label for points in the intersection. Even if we perfectly classify all the examples not in the intersection, if $p > 1/2$ then the false positive

rate of A is higher than $1/2$ (as it misclassifies half of the negative examples); otherwise the false negative rate of B is no less than $1/2$. This means that one of the two populations will necessarily have unweighted average of false positive and false negative rates higher than $1/4$ and therefore will not achieve performance even approximately as good as its best predictor. ◀

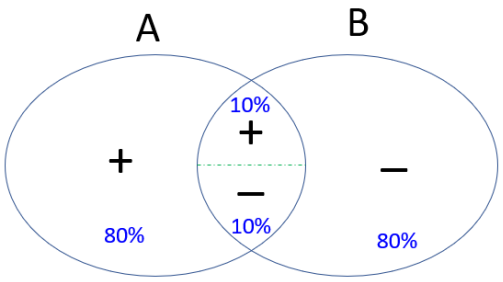


Figure 1 Construction illustrating that unweighted average of false negative and false positive rate cannot be simultaneously optimized with respect to overlapping populations (Theorem 10).

Discussion

The above result illustrates the additional complications caused by overlapping groups and highlights a subtlety in the positive results we provided in the previous sections. In particular, the key distinction is that for the objective of average loss (or error rate), the average contribution of any given example is the same across all groups; a misclassification is equally damaging for all groups an example belongs to. This is not the case in the setting considered here of minimizing the average (or maximum or any fixed nontrivial combination) of FPR and FNR where the harm of each misclassified example needs to be weighted differently across groups depending on their proportion of positive and negative examples.

6 Conclusions

In this paper, we consider settings where different overlapping populations coexist and we wish to design algorithms that do not treat any population unfairly. We consider a notion of fairness that corresponds to predicting as well as humanly (or algorithmically) possible on each given group, rather than based on requirements for equality. This framework can directly incorporate a designer's goal of good overall prediction by creating one extra group (for the designer) that includes all the examples. Our results extend to the more realistic one-sided feedback (apple tasting) setting and have a nice game-theoretic interpretation. Our work makes a step towards identifying fairness notions that work well in online settings and are compatible with the existence of multiple parties, each with their own interests in mind. Our impossibility results with respect to the average (or maximum) of false negative and false positive rates demonstrate that satisfying the interests of different overlapping populations is quite subtle, further highlighting the positive results.

Regarding incentives, we show how to efficiently achieve Individual Rationality and how to inefficiently achieve both Individual Rationality and Incentive Compatibility. Achieving IR and IC together in a computationally efficient way (without enumerating across all disjoint

intersections of subgroups) is a very interesting question that seems to require novel learning-theoretic ideas as we discuss in Section 4.3. On the apple tasting front, we provide three distinct guarantees; the resulting algorithms are near-optimal but suffer from orthogonal shortcomings as we discuss in Section 3.2. Avoiding these and achieving an optimal guarantee for sleeping experts with apple tasting is an interesting open question.

References

1. Alekh Agarwal, Alina Beygelzimer, Miroslav Dudík, John Langford, and Hanna Wallach. A reductions approach to fair classification. *arXiv preprint*, 2018. `arXiv:1803.02453`.
2. Itai Ashlagi, Felix Fischer, Ian A. Kash, and Ariel D. Procaccia. Mix and match: A strategyproof mechanism for multi-hospital kidney exchange. *Games and Economic Behavior*, 91, May 2013. `doi:10.1016/j.geb.2013.05.008`.
3. Itai Ashlagi and Alvin E Roth. Free riding and participation in large scale, multi-hospital kidney exchange. *Theoretical Economics*, 9(3):817–863, 2014.
4. Baruch Awerbuch and Yishay Mansour. Adapting to a reliable network path. In *Proceedings of the twenty-second annual symposium on Principles of distributed computing (PODC)*, 2003.
5. Yahav Bechavod, Katrina Ligett, Aaron Roth, Bo Waggoner, and Zhiwei Steven Wu. Equal Opportunity in Online Classification with Partial Feedback. In *33rd Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2019.
6. Avrim Blum. Empirical support for winnow and weighted-majority algorithms: Results on a calendar scheduling domain. *Machine Learning*, 26(1):5–23, 1997.
7. Avrim Blum, Suriya Gunasekar, Thodoris Lykouris, and Nathan Srebro. On preserving non-discrimination when combining expert advice. In *32nd Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2018. URL: `http://papers.nips.cc/paper/8058-on-preserving-non-discrimination-when-combining-expert-advice`.
8. Avrim Blum and Yishay Mansour. From external to internal regret. *Journal of Machine Learning Research (JMLR)*, 2007.
9. Toon Calders, Faisal Kamiran, and Mykola Pechenizkiy. Building Classifiers with Independency Constraints. In *IEEE International Conference on Data Mining (ICDM)*, 2009.
10. L Elisa Celis, Sayash Kapoor, Farnood Salehi, and Nisheeth K Vishnoi. An Algorithmic Framework to Control Bias in Bandit-based Personalization. *arXiv preprint*, 2018. `arXiv:1802.08674`.
11. Nicolo Cesa-Bianchi, Gábor Lugosi, and Gilles Stoltz. Minimizing regret with label efficient prediction. *IEEE Transactions on Information Theory*, 51(6):2152–2162, 2005.
12. Alexandra Chouldechova. Fair prediction with disparate impact: A study of bias in recidivism prediction instruments. *Big data*, 5(2):153–163, 2017.
13. Sam Corbett-Davies and Sharad Goel. The measure and mismeasure of fairness: A critical review of fair machine learning. *arXiv preprint*, 2018. `arXiv:1808.00023`.
14. Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. Fairness Through Awareness. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference (ITCS)*, 2012.
15. Michael Feldman, Sorelle A. Friedler, John Moeller, Carlos Scheidegger, and Suresh Venkatasubramanian. Certifying and Removing Disparate Impact. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2015.
16. Yoav Freund and Robert E Schapire. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *J. Comput. Syst. Sci.*, 1997.
17. Yoav Freund, Robert E Schapire, Yoram Singer, and Manfred K Warmuth. Using and combining predictors that specialize. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing (STOC)*, pages 334–343. ACM, 1997.
18. Pierre Gaillard, Gilles Stoltz, and Tim Van Erven. A second-order bound with excess losses. In *Conference on Learning Theory (COLT)*, 2014.

19 Stephen Gillen, Christopher Jung, Michael Kearns, and Aaron Roth. Online Learning with an Unknown Fairness Metric. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

20 Eyal Gofer and Yishay Mansour. Lower bounds on individual sequence regret. *Machine Learning*, 103(1):1–26, 2016.

21 Swati Gupta and Vijay Kamble. Individual Fairness in Hindsight. In *Proceedings of the 2019 ACM Conference on Economics and Computation (EC)*, 2019.

22 Moritz Hardt, Eric Price, and Nathan Srebro. Equality of opportunity in supervised learning. In *Advances in neural information processing systems (NIPS)*, 2016.

23 Ursula Hebert-Johnson, Michael Kim, Omer Reingold, and Guy Rothblum. Multicalibration: Calibration for the (Computationally-Identifiable) Masses. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, 2018.

24 David P. Helmbold, Nick Littlestone, and Philip M. Long. Apple Tasting and Nearly One-Sided Learning. In *33rd Annual Symposium on Foundations of Computer Science (FOCS)*, 1992.

25 Matthew Joseph, Michael Kearns, Jamie H Morgenstern, and Aaron Roth. Fairness in learning: Classic and contextual bandits. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

26 Michael Kearns, Seth Neel, Aaron Roth, and Zhiwei Steven Wu. Preventing Fairness Gerrymandering: Auditing and Learning for Subgroup Fairness. In *In Proceedings of the 35th International Conference on Machine Learning (ICML)*, 2018.

27 Niki Kilbertus, Mateo Rojas Carulla, Giambattista Parascandolo, Moritz Hardt, Dominik Janzing, and Bernhard Schölkopf. Avoiding discrimination through causal reasoning. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

28 Michael P. Kim, Amirata Ghorbani, and James Zou. Multiaccuracy: Black-Box Post-Processing for Fairness in Classification. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, AIES '19, 2019. doi:10.1145/3306618.3314287.

29 Michael P. Kim, Omer Reingold, and Guy N. Rothblum. Fairness Through Computationally-bounded Awareness. In *Proceedings of the 32Nd International Conference on Neural Information Processing Systems (NeurIPS)*, 2018. URL: http://dl.acm.org/citation.cfm?id=3327345.3327393.

30 Jon Kleinberg, Himabindu Lakkaraju, Jure Leskovec, Jens Ludwig, and Sendhil Mullainathan. Human decisions and machine predictions. *The quarterly journal of economics*, 133(1):237–293, 2017.

31 Jon M. Kleinberg, Sendhil Mullainathan, and Manish Raghavan. Inherent Trade-Offs in the Fair Determination of Risk Scores. In *Innovations of Theoretical Computer Science (ITCS)*, 2017.

32 Nick Littlestone and Manfred K. Warmuth. The Weighted Majority Algorithm. *Inf. Comput.*, 108(2):212–261, February 1994. doi:10.1006/inco.1994.1009.

33 Lydia T. Liu, Sarah Dean, Esther Rolf, Max Simchowitz, and Moritz Hardt. Delayed Impact of Fair Machine Learning. *35th International Conference on Machine Learning (ICML)*, 2018.

34 Yang Liu, Goran Radanovic, Christos Dimitrakakis, Debmalya Mandal, and David C Parkes. Calibrated Fairness in Bandits. In *Workshop on Fairness, Accountability, and Transparency in Machine Learning (FAT-ML)*, 2017.

35 Haipeng Luo and Robert E. Schapire. Achieving All with No Parameters: AdaNormalHedge. In *Proceedings of The 28th Conference on Learning Theory (COLT)*, 2015. URL: http://jmlr.org/proceedings/papers/v40/Luo15.html.

36 Manish Raghavan, Aleksandrs Slivkins, Jennifer Vaughan Wortman, and Zhiwei Steven Wu. The Externalities of Exploration and How Data Diversity Helps Exploitation. In *Proceedings of the 31st Conference On Learning Theory (COLT)*, 2018.

37 Alvin E. Roth, Tayfun Sönmez, and M. Utku Ünver. Efficient Kidney Exchange: Coincidence of Wants in Markets with Compatibility-Based Preferences. *American Economic Review*, 97(3):828–851, June 2007. doi:10.1257/aer.97.3.828.

A Proof of Theorem 3

Theorem 3 restated

Let $\mathcal{A}$ be an algorithm with regret bounded by $\mathcal{O}\sqrt{T \cdot \log(|\mathcal{H}|)}$ when compared to an expert class $\mathcal{H}$, run on T examples and split the examples in disjoint intersections, where each intersection corresponds to a distinct profile of subgroup memberships. For each intersection I, randomly selecting an exploration point every $(T(I))^{1/3}$ examples and running separate versions of $\mathcal{A}$ for each I provides subgroup regret on group $g \in \mathcal{G}$ of

$$\mathcal{O}\left(\left(2^{|\mathcal{G}|}\right)^{1/3} \cdot (T(g))^{2/3} \cdot \sqrt{\log(N)}\right)$$

where $T(g) = |t : g \in \mathcal{G}(t)|$ is the size of the g-population and $N = |\mathcal{F}| + \sum_{g \in \mathcal{G}} |\mathcal{F}(g)|$.

Proof. The guarantee follows from three observations:

1. Among the exploration points, we run a classical experts algorithm so, on these examples, we have a regret guarantee that is square-root of the number of these examples.
2. For each phase, the exploration point is randomly selected and therefore the regret that we incur in the exploration point is an unbiased estimator of the average regret we incur in the whole phase (since the distribution of the algorithm in the phase is the same). As a result, the total regret in a phase is in expectation $(T(I))^{1/3}$ times the regret at the exploration point.
3. A particular group can have examples in at $2^{|G|}$ intersections (as this is all the possible membership relationships with respect to the demographic groups).

We now formalize these three ideas, to obtain the guarantee. Let $\mathcal{F}(I)$ be the set of experts that are either global or belong to some $g \in I$, and update $\mathcal{F}(g)$ to include both the global experts and the group-specific experts. Initially we split the performance of our algorithm across all the intersections I such that $g \in I$. Let $f^\star$ be the comparator with the minimum cumulative loss on g.

$$Reg(g) = \sum_{t:g \in \mathcal{G}(t)} \sum_{f \in \mathcal{F}(g)} p_{t,f} \ell_{t,f} - \sum_{t:g \in \mathcal{G}(t)} \ell_{t,f^\star} = \sum_{I:g \in I} \sum_{t:g \in \mathcal{G}(t) \cap I} \left(\sum_{f \in \mathcal{F}(g)} p_{t,f} \ell_{t,f} - \ell_{t,f^\star} \right)$$

Focusing on a particular intersection I we connect its regret to the performance of the exploration points. Denote by $t(r, I)$ the exploration point for phase r in intersection I. Also denote by $\tau(r)$ the beginning of the r-th phase. We use the fact that the exploration point is an unbiased representation on the regret at a phase (as it is selected uniformly and the algorithm is only updated at the end of the phase). Applying the guarantee for $\mathcal{A}$ on the exploration times of all phases, we obtain:

$$\begin{aligned}
\sum_{t:g \in \mathcal{G}(t) \cap I} \left(\sum_{f \in \mathcal{F}(g)} p_{t,f} \ell_{t,f} - \ell_{t,f^\star} \right) &= \sum_{r=1}^{(T(I))^{2/3}} \sum_{t=\tau_r}^{\tau_{r+1}-1} \left(\sum_{f \in \mathcal{F}(g)} p_{t,f} \ell_{t,f} - \ell_{t,f^\star} \right) \cdot \mathbf{1}\{g \in \mathcal{G}(t) \cap I\} \\
&= (T(I))^{1/3} \cdot \sum_{r=1}^{(T(I))^{2/3}} \mathbb{E}\left[\sum_{f \in \mathcal{F}(g)} p_{t(r,I),f} \ell_{t(r,I),f} - \ell_{t(r,I),f^\star} \right] \\
&\leq (T(I))^{1/3} \sqrt{(T(I))^{2/3} \log(|\mathcal{F}|)} = (T(I))^{2/3} \cdot \sqrt{\log(|\mathcal{F}|)}
\end{aligned}$$

where the expectations are taken over the random selections of $t(r, I)$.

Combining the above and applying Holder inequality, we obtain:

$$\mathcal{R}(g) = \sum_{I:g\in I}\sum_{t:g\in\mathcal{G}(t)\cap I}\left(\sum_{f\in\mathcal{F}(g)} p_{t,f}\ell_{t,f} - \ell_{t,f^\star}\right) \leq \sum_{I:g\in I}(T(I))^{2/3}\cdot\sqrt{\log(|\mathcal{F}|)}$$
$$\leq \left(\sum_{I:g\in I} 1\right)^{1/3}\cdot\left(\sum_{I:g\in I}(T(I))\right)^{2/3}\sqrt{\log(|\mathcal{F}|} \leq \left(2^{|\mathcal{G}|}\right)^{1/3}\cdot(T(g))^{2/3}\cdot\sqrt{(\log(|\mathcal{F}|)}.$$

Finally, we need to control how much we are losing via the exploration rounds. Since we apply pay-per-feedback we lose 1 every time we inspect (an upper bound on the apple tasting loss). For any intersection, we have exactly $(T(I))^{2/3}$ such exploration points so we are not losing something more compared to what we previously discussed which completes the proof. ◀

B Proof of Theorem 4

Theorem 4 restated

Let $\mathcal{A}$ be an algorithm with sleeping regret bounded by $\mathcal{O}\left(\sqrt{T(h)\cdot\log(|\mathcal{H}|)}\right)$ for any expert $h\in\mathcal{H}$ where $\mathcal{H}$ is any class. Randomly selecting an exploration point every $T^{1/3}$ examples (irrespective of what groups they come from) and running $\mathcal{A}$ on these points provides subgroup regret on group $g\in\mathcal{G}$ of

$$\mathcal{O}\left(T^{1/6}\cdot(T(g))^{-1/2}\cdot\sqrt{\log(N)}\right)$$

where $T(g) = |t : g\in\mathcal{G}(t)|$ is the size of the g-population and $N = |\mathcal{F}| + \sum_{g\in\mathcal{G}}|\mathcal{F}(g)|$.

Proof. The guarantee follows from two observations:

1. Given that we run a sleeping experts algorithm across the exploration points, if we just focused on those examples, we simultaneously satisfy regret on them that is square-root of their size.
2. Within any phase the exploration point is uniformly at random selected. As a result, it is an unbiased estimator of the average regret we incur in the whole phase. Note that this is now the average across all rounds and not only rounds where we have members of the particular group which results to the dependence on the time-horizon T.

We now formalize these ideas. As before, we connect the regret on a group to the one of the exploration points. Denote by $t(r)$ the random variable that corresponds to the exploration point. Now the size of the phases is fixed in advance so r-th phase starts at $\tau_r = r\cdot T^{1/3}$. Similarly as before, we denote by $f^\star$ the comparator with the minimum cumulative loss on g. Applying linearity of expectation and Jensen's inequality, we obtain:

$$Reg(g) = \sum_{t:g\in\mathcal{G}(t)}\sum_{f\in\mathcal{F}(g)} p_{t,f}\ell_{t,f} - \sum_{t:g\in\mathcal{G}(t)}\ell_{t,f^\star}$$
$$= \sum_{r=1}^{T^{2/3}}\sum_{t=\tau_r}^{\tau_{r+1}-1}\left(\sum_{f\in\mathcal{F}(g)} p_{t,f}\ell_{t,f} - \ell_{t,f^\star}\right)\cdot\mathbf{1}\{g\in\mathcal{G}(t)\}$$

$$\ldots = T^{1/3} \cdot \sum_{r=1}^{T^{2/3}} \mathbb{E}\left[\left(\sum_{f\in\mathcal{F}(g)} p_{t(r),f}\ell_{t(r),f} - \ell_{t(r),f^\star}\right)\cdot \mathbf{1}\{g \in \mathcal{G}(t(r))\}\right]$$
$$\leq T^{1/3} \cdot \mathbb{E}\left[\sqrt{\sum_{r=1}^{T^{2/3}} \mathbf{1}\{g \in \mathcal{G}(t(r))\}\cdot \log(N)}\right]$$
$$\leq T^{1/3} \cdot \sqrt{\mathbb{E}\left[\sum_{r=1}^{T^{2/3}} \mathbf{1}\{g \in \mathcal{G}(t(r))\}\cdot \log(N)\right]}$$
$$= T^{1/3} \cdot \sqrt{T(g)\cdot T^{-1/3}\cdot \log(N)} = T^{1/6}\cdot\sqrt{T(g)\cdot\log(N)}.$$

The expectation is taken over the random selections of the exploration times $t(r)$. The second-to-last equality holds as each member of group g has probability $T^{-1/3}$ to be an exploration point and therefore the expected number of exploration points in the group is $T(g)\cdot T^{-1/3}$.

Finally, for the pay-for-feedback model, we need to also consider the effect of inspected rounds in loss (we are losing 1 every time that we inspect). The number of these rounds on examples in g is at most $T(g)\cdot T^{-1/3} \leq (T(g))^{2/3}$ so lower order than the term we already have. ◀

C Proof of Theorem 5

Theorem 5 restated

Applying the algorithm described in the third reduction (Section3.2 provides subgroup regret on group $g \in \mathcal{G}$ of

$$\mathcal{O}\left(|\mathcal{G}|\cdot(T(g))^{3/4}\cdot\sqrt{\log(N)}\right)$$

where $T(g) = |t : g \in \mathcal{G}(t)|$ is the size of the g-population and $N = |\mathcal{F}| + \sum_{g\in\mathcal{G}}|\mathcal{F}(g)|$.

Proof. There are three important components to prove this guarantee.

1. The number for relevant phases of each group (phases where they have at least one example) is at most $T(g)$. This provides an upper bound on the number of phases that we need to consider with respect to group g.
2. Using a similar analysis as before, we can create a guarantee about the regret we are incurring in the exploration points and multiply it by the $(T(g))^{1/4}$ which is the size of the phase. This would have created a completely unbiased estimator if there was no overlap with other groups.
3. A final complication is that exploration may occur due to other groups so we need to understand how much we lose there. For that, we observe that smaller groups are explored with higher probability (the interaction with larger groups does not therefore significantly increase their probability of inspection). On the other hand, larger groups do not often collide with significantly smaller groups due to the latters' size.

More formally, to analyze the subgroup regret for $g \in \mathcal{G}$, let's consider a fictitious setting where all phases have equal size for group g. This can be done by padding 0s in the end of the phase. This fictitious setting has the same loss as the original (as it only differs in that it has some more examples with zero loss). For this fictitious setting, the exploration point is

an unbiased estimator of the average regret. We first analyze the subgroup regret assuming that the points of inspection in other groups do not overlap with g. Applying similar ideas as in the previous theorem:

$$\begin{aligned}
Reg(g) &= \sum_{t:g\in\mathcal{G}(t)} \sum_{f\in\mathcal{F}(g)} p_{t,f}\ell_{t,f} - \sum_{t:g\in\mathcal{G}(t)} \ell_{t,f^\star} \\
&= \sum_{r=1}^{T(g)} \sum_{t=\tau_r}^{\tau_{r+1}-1} \left(\sum_{f\in\mathcal{F}(g)} p_{t,f}\ell_{t,f} - \ell_{t,f^\star} \right) \cdot \mathbf{1}\{g\in\mathcal{G}(t)\} \\
&= (T(g))^{1/4} \cdot \sum_{r=1}^{T(g)} \mathbb{E}\left[\left(\sum_{f\in\mathcal{F}(g)} p_{t(r),f}\ell_{t(r,g),f} - \ell_{t(r,g),f^\star} \right) \cdot \mathbf{1}\{g\in\mathcal{G}(t(r,g)), t(r,g)\le t_{r+1}\} \right] \\
&= (T(g))^{1/4} \cdot \sum_{r=1}^{T(g)} \mathbb{E}\left[\left(\sum_{f\in\mathcal{F}(g)} p_{t(r),f}\tilde{\ell}_f(r,g) - \tilde{\ell}_{f^\star(r,g)} \right) \right] \\
&\le (T(g))^{1/4} \cdot \sqrt{T(g)\log(N)} = (T(g))^{3/4}\sqrt{\log(N)}
\end{aligned}$$

The expectation is taken over the random selections of the exploration times $t(r,g)$. The last inequality holds as the number of non-zero entries in the previous sum is at most the number of phases and we run $\mathcal{A}$ on the estimated losses.

Let's understand now how much group g is harmed by the fact that some of its examples may be inspected by overlapping groups. As a result, we need to count in the regret the contribution of these examples as well. Note that each group has at most one exploration point within a phase – this is the reason why we get the dependence on $|\mathcal{G}|$ in our bound.

Group g is only affected by others' exploration if this happens on examples that are also members of group g. We first consider the effect of smaller groups than g. For any group g' with $T(g') \le T(g)$ the exploration points at group g' are at most the size of the group times the probability that each example is an exploration point for g', i.e. $T(g')\cdot(T(g'))^{-1/4} = (T(g'))^{3/4}$. Since the size of g' is smaller than the size of g, this contributes at most an extra term of $(T(g))^{3/4}$ in the regret.

Let's now consider groups g' with $T(g') > T(g)$. Then, even if all examples of g are also examples of g', the probability that each of them is an exploration point due to g' is $(T(g'))^{-1/4}$. Hence the expected number of exploration points of g' on examples in g is at most $T(g)\cdot(T(g'))^{1/4} \le (T(g))^{3/4}$. ◀

Instance Complexity and Unlabeled Certificates in the Decision Tree Model

Tomer Grossman
Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot 76100, Israel
tomer.grossman@weizmann.ac.il

Ilan Komargodski
NTT Research, Palo Alto, CA, USA
ilan.komargodski@ntt-research.com

Moni Naor[1]
Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot 76100, Israel
moni.naor@weizmann.ac.il

Abstract

Instance complexity is a measure of goodness of an algorithm in which the performance of one algorithm is compared to others per input. This is in sharp contrast to worst-case and average-case complexity measures, where the performance is compared either on the worst input or on an average one, respectively.

We initiate the systematic study of instance complexity and optimality in the query model (a.k.a. the decision tree model). In this model, instance optimality of an algorithm for computing a function is the requirement that the complexity of an algorithm on any input is at most a constant factor larger than the complexity of the best correct algorithm. That is we compare the decision tree to one that receives a certificate and its complexity is measured only if the certificate is correct (but correctness should hold on any input). We study both deterministic and randomized decision trees and provide various characterizations and barriers for more general results.

We introduce a new measure of complexity called *unlabeled*-certificate complexity, appropriate for graph properties and other functions with symmetries, where only information about the structure of the graph is known to the competing algorithm. More precisely, the certificate is some permutation of the input (rather than the input itself) and the correctness should be maintained even if the certificate is wrong. First we show that such an unlabeled certificate is sometimes very helpful in the worst-case. We then study instance optimality with respect to this measure of complexity, where an algorithm is said to be instance optimal if for every input it performs roughly as well as the best algorithm that is given an unlabeled certificate (but is correct on every input). We show that instance optimality depends on the group of permutations in consideration. Our proofs rely on techniques from hypothesis testing and analysis of random graphs.

2012 ACM Subject Classification Theory of computation → Complexity theory and logic

Keywords and phrases decision tree complexity, instance complexity, instance optimality, query complexity, unlabeled certificates

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.56

Funding *Tomer Grossman*: Israel Science Foundation (no. 950/16).
Ilan Komargodski: AFOSR grant FA9550-15-1-0262, Israel Science Foundation (no. 950/16), and a Levzion Fellowship.
Moni Naor: Israel Science Foundation (no. 950/16).

[1] Incumbent of the Judith Kleeman Professorial Chair.

11th Innovations in Theoretical Computer Science Conference (ITCS 2020).
Editor: Thomas Vidick; Article No. 56; pp. 56:1–56:38

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Acknowledgements We thank Scott Aaronson for discussions on quantum certificates, Ron Fagin for useful comments, and Ofer Grossman for helpful discussions and Yoram Moses and Benny Applebaum for suggesting instance optimality in other settings. We thank some of the referees of the paper for helpful comments.

1 Introduction

Worst-case analysis is the hallmark of theoretical computer science. An algorithm is evaluated based on its performance on the worst input and we want to find the algorithm whose performance is the best according to this criterion. Nevertheless, in some cases one might not be interested only in worst-case complexity of a function f. For instance, when the worst case is inherently "very bad" for all algorithms (and then measuring the algorithms by its worst case input gives meaningless comparisons), when the worst case is very rare, or when there is some prior information about the distribution of the inputs.

In this work we consider a new measure of complexity for problems we call *the instance complexity*. Here, we compare a given algorithm to one that has additional information about the input. That is, we are interested in the best algorithm for a given problem that performs as well as possible on every input compared to the best algorithm that *knows* (something about) the input. The instance complexity (denoted IC) of a problem is the overhead of solving the problem on every input vs. solving the problem on a given instance, provided that algorithms the specific instance is a correct algorithm (on all inputs). Here is one way to define this notion. Suppose that we have a collection of algorithms $\mathcal{A}$ that are correct on all on inputs.

▶ **Definition 1.1** (Instance complexity). *Let f be a function, and A an algorithm that evaluates f. Let $\mathrm{cost}(A, x)$ be the cost of algorithm A on input x, where cost is some non-negative value that we want to optimize (e.g. runtime, space, etc.). We say that f is* instance optimizable *and that A is* instance optimal *if for every input x, and for every algorithm $A' \in \mathcal{A}$ (that evaluates f correctly on every input) the following holds:*

$$\mathrm{cost}(A, x) \in O\left(\mathrm{cost}(A', x)\right).$$

In the definition above, we compare the cost of A on every input with the cost of an algorithm that has the input hardwired. One can relax the hint given to the competing algorithm A' so that it only knows *something* about the input (e.g., an unknown permutation of the input, a subset of the coordinates, etc). Finally, note that both algorithms A and A' should be correct on all inputs, but the competing algorithm is evaluated in terms of cost *only when the hint it is given is correct w.r.t. the input.*

The term "Instance optimality" was coined by Fagin, Lotem and Naor [19] in the context of finding items with the top k aggregate scores in a database of sorted lists. It has appeared in the theoretical computer science literature in several other contexts and forms. For previous works on instance complexity in other settings see Appendix B.

We initiate the systematic study of instance complexity in the *query* model (a.k.a. the *decision tree* model). In this model the input is given through an oracle, and the goal is to minimize the number of queries made to the input oracle. The query complexity model helps us understand the power of various resources (such as randomness and non-determinism) and inherent difficulties in computing a function. Not only is it a simple and clean model, but sometimes results in this model carry over to different models (i.e., "lifting" [43, 28]). See Section 2 for more details about the decision tree model, formal definitions, and known results.

We study two different notions of instance complexity. In the first notion the hint is simply the full input. The instance complexity, IC, of an algorithm is the maximal ratio between its complexity on a given input and the complexity of any other algorithm that *knows* the specific input (but is correct on all inputs). In the second notion the hint is some *unlabeled version of the input*; namely, the input is given in an (adversarially) permuted order. This is relevant when the function we wish to compute treats the input as symmetric, e.g., as in graph properties.[2] We call this notion *unlabeled instance complexity*, uIC,[3] and for a given algorithm define it to be the maximal ratio between its complexity on a given input and the complexity of any other algorithm that *knows* an *unlabeled version of the input* (but is correct on all inputs).

Models for instance complexity. Instance optimality captures the fact that an algorithm is not only optimal for a worst-case input or a random one, but that it is optimal for *every* input (up to constant factors). In the context of decision tree complexity, we formalize this in the following way: An algorithm is instance optimal if for every input x its complexity is at most a constant factor larger than the complexity of any decision tree whose goal is to be as efficient as possible on x (yet correct on all inputs). This naturally leads to comparing the efficiency of the decision tree on an input to the complexity of a decision tree that gets that input as a certificate. Thus, a function is instance optimizable if there is an algorithm such that for **every** input (not just the worst case input) its complexity on that input is at most a constant factor larger than the certificate complexity of the function on the same input. To show that an algorithm is *not* instance optimal it is possible to argue about inputs where the running time is not the worst case.

For example, the parity function is instance optimal, since any algorithm, on any input must read the entire input (see Example 3.1 for detail). The OR function on the other hand is not instance optimal. This is because for any algorithm, A, there exists an input, x, and another algorithm A^*, such that A does n queries on input x while A^* does 1 query (see Example 3.3 for detail).

We wish to emphasize that instance optimality is a fundamentally different notion than worst case complexity. Notice that for the OR function, in the worst case the certificate complexity is $\Theta(n)$ and any algorithm without a certificate also requires $\Theta(n)$ queries yet the OR function is not instance optimal.

We define a similar notion for randomized algorithms. Namely, an algorithm is said to be randomized instance optimal if for every input x its expected complexity is at most a constant factor larger than the randomized certificate complexity of the function on x. We will consider randomized algorithms with two-sided error.

We introduce the notion of **unlabeled certificate complexity** in which the certificate is not the input, but a permutation thereof. This is relevant when the function is invariant under a *family of permutations* (e.g., a graph property or a symmetric function, such as threshold). We show that this is sometimes very helpful, *even in the worst-case*, in the sense that there are functions where the unlabeled certificate complexity is much smaller than the randomized complexity. Having this notion in hand, it is natural to define (randomized-) unlabeled instance optimality, where the standard (randomized) decision tree algorithm is

[2] A graph property is a set of graphs closed under graph isomorphism; see Section 2.3.

[3] The closest notion considered in previous literature is in geometric algorithms by Afshani et al. [2], see Appendix B.

required to have complexity (input-by-input) at most a constant factor larger than the best algorithm that gets a permutation (from the family) of the input as a certificate and computes the value of the function correctly. The algorithm that gets the unlabeled certificate is required to be correct on all inputs, but its complexity is measured only in case the certificate is correct.

1.1 Our Results

We lay the foundation for the study of instance optimality and unlabeled certificate complexity in the decision tree model by providing a host of results including various examples, relations, separations, and more. Our results fall short of a (complete) characterization of which functions are instance optimal and which are not and when unlabeled certificates help and when they do not. However, our results demonstrate that the theory of instance optimality and unlabeled complexity is rich, and contains non-trivial and sometimes surprising phenomena. Thus we believe that this theory deserves further research, and hope that this work will inspire such research.

Below, our results are stated in the order corresponding to the above models: deterministic and randomized instance optimality (Section 1.1.1), and then unlabeled certificate complexity (Section 1.1.2).

1.1.1 Deterministic and Randomized Instance Optimality

We start off by observing that even in the simplest setting of deterministic decision trees, instance optimality is a non-trivial property: there are functions that are instance optimizable and functions that are not. Also, monotonicity does not directly affect instance optimizability. Furthermore, we construct a function that is *strictly* instance optimal (*strict* means that the decision tree has no additional overhead over the one that has a certificate), but once composed with itself, results in a function that is not instance optimal.

▶ **Theorem 1.2** (Deterministic instance optimality). *The following holds:*

1. *There is a monotone function that is* not *deterministic instance optimal while the Majority function is deterministic instance optimizable (and monotone).*
2. *The Addressing and Parity functions are examples of non-monotone functions that are strictly instance optimizable.*
3. *Among the monotone functions, the only ones that can be* strictly *instance optimizable are functions that depend on 0 or 1 variables.*
4. *The composition of two (strictly) instance optimal functions is not necessarily instance optimal.*

In the randomized case, the situation is somewhat different. While the parity and addressing functions behave similarly, the majority function becomes *not* instance optimizable! But, there *are* other instance optimizable monotone functions. Also, there are (non-monotone) graph properties that are instance optimizable. We further provide two results that capture a wide range of functions: (1) random functions are instance optimal and (2) a relation between instance optimizability and properties related to *sensitivity* (see Sections 3.2.1 and 3.2.2). As in the deterministic setting, the composition of instance optimal functions is not necessarily instance optimal.

One notable question left open is whether there is a *monotone* graph property that is randomized instance optimizable. We conjecture that such functions do not exist.

▶ **Theorem 1.3** (Randomized instance optimality). *The following holds:*

1. *The Addressing and Parity functions are randomized instance optimizable functions.*
2. *The Majority function is* not *randomized instance optimizable. Thus, deterministic instance optimality does not imply randomized instance optimality. The other direction is true as well: There exists a function that is randomized instance optimizable but not deterministic instance optimizable (see Section 3.2.3).*
3. *There exists a* monotone *function that is randomized instance optimizable.*
4. *There exists a graph property that is randomized instance optimizable.*
5. *A random function is randomized instance optimizable with high probability.*
6. *The composition of two randomized instance optimal functions is not necessarily randomized instance optimal.*

We consider the computational complexity of figuring out whether a function is instance optimizable and whether a given algorithm is instance optimal. A good time complexity in this setting is polynomial in the size of the representation of the function, i.e. exponential in n for a Boolean function with input $\{0,1\}^n$. We do not have an algorithm for the former problem (deciding whether a function is instance optimizable), but for the latter we show that an algorithm can be *verified* to be instance optimal in time that is polynomial in the domain size of the problem. Naively, one has to work in time *exponential* in the domain size and go over all possible decision trees and test against each one (see Section 3.5).

▶ **Theorem 1.4** (Complexity of checking if an algorithm is instance optimal). *Given a function f on n input variables (given by a truth-table of size 2^n), and a randomized algorithm T, there exists an algorithm U that decides in time $2^{O(n)}$ whether T is an instance optimal algorithm.*

We also consider *proximity property testing*[4] where the goal is to decide whether an object has some property or whether it is *far* from having it [26, 25]. Here it is possible to characterize the properties that are randomized instance optimizable: it is exactly those that can be tested in $O(1)$ queries (see Section 3.6).

1.1.2 Unlabeled Certificate Complexity

We introduce the study of the **unlabeled certificate complexity**. We show that an unlabeled certificate can help significantly in the worst case for graph properties. That is, we construct a function for which an algorithm that has an isomorphic copy of the graph as a certificate outperforms an algorithm that isn't given any certificate.

▶ **Theorem 1.5** (Unlabeled certificate complexity). *In both the randomized and deterministic setting, there exists a graph property for which the unlabeled certificate complexity is $O(n \log n)$ while any algorithm with no certificate requires $\Omega(n^2)$ queries.*

Having the notion of unlabeled certificates we define **unlabeled randomized instance complexity and optimality**. Here the situation changes again. Whereas the majority function is not randomized instance optimizable, it *is* (almost) unlabeled randomized instance

[4] We use the term "proximity property testing" so as not to confuse the reader with the case where we are testing whether the given graph satisfies a property, rather than being close to a graph that satisfies the property.

optimizable.[5] However, when viewed as a *graph property*, the majority function becomes *not* unlabeled randomized instance optimizable. The group of permutations we consider in the latter is the group of all isomorphic copies of the graph. In addition, we show that being a graph property is not directly related to unlabeled instance optimality. We say a function is **labelling instance optimal** if the unlabeled certificate performs as well as the labeled certificate for every input up to a constant. All instance optimal functions are also labelling instance optimal. We show that (up to log factors) the converse is not true: there exists a function which is within $O(\log n)$ of being labelling instance optimal, but not instance optimal in the normal sense.

▶ **Theorem 1.6** (Unlabeled instance optimality)**.** *The following holds:*

1. *The majority function is unlabeled randomized instance optimizable within* $O(\log \log n)$.
2. *The graph property of having more edges than non-edges is far from being unlabeled randomized-instance optimizable – it is at best within* $\tilde{\Omega}(n)$ *from being unlabeled randomized-instance optimizable, where* n *is the number of vertices in the graph.*
3. *The graph property of having at least one edge is unlabeled randomized instance optimizable and not labelling instance optimal.*
4. *There exists a function that is* $O(\log n)$ *away from being labelling instance optimal, but is not instance optimal in the normal sense.*

An impatient reader can skip ahead to Section 4, which is about unlabeled complexity, and contains the proofs of the items in Theorem 1.5 and Theorem 1.6.

The saga of the majority. As we shall see, the *majority* function behaves differently under different notions of instance optimality. While it is deterministic instance optimizable, it is *not* randomized instance optimizable (Lemma 3.5). Having only a permutation of the input as a certificate does not provide additional power to the one receiving it, i.e., it is (almost) unlabeled randomized instance optimizable (Lemma 4.4). On the other hand, viewing majority as a graph property, getting a permutation of the graph may help and majority becomes a function that is *not* unlabeled randomized instance optimizable (Lemma 4.5).

2 The Query Model and Instance Optimality

The query model is one of the simplest computational models and has been studied widely. See, for example, Buhrman and de Wolf [12], Jukna [34, Chapter 14] and O'Donnell [41, Chapter 8.6] for a survey. We give some standard definitions of *decision trees* and several related complexity measures such as *deterministic*, *randomized* and *certificate* complexity, and then briefly recall the known connections between them.

A decision tree is an algorithm for computing the value of a function on an a-priori unknown input. Let $f\colon \{0,1\}^n \to \{0,1\}$ be a Boolean function over a set of n input variables $x_1, \ldots, x_n$. A deterministic decision tree T over a set of input variables $x_1, \ldots, x_n$ is a rooted binary tree. Every internal vertex is labeled with an input variable. The leaves of the tree are labeled by 0 or 1, and every internal vertex has one outgoing edge labeled by 0 and the other by 1. The computation of the tree is done in the natural way from the root to the

[5] In this result, the optimality ratio is a super-constant function ($\log \log$) of the input size. In such cases, where the optimality ratio is a super constant function $g(n)$ of the input size, we will say that the function is *within* $g(n)$ *of being (unlabeled /randomized-) instance optimizable.*

leaves according to the assignment of the input variables. We denote by $T(x)$ the output of the decision tree T on input x. We say that the decision tree T computes f if for any input x it holds that $f(x) = T(x)$.

Denote by $\mathcal{T} = \mathcal{T}_n$ the set of all decision trees on functions of n inputs. The complexity (i.e., cost) of a decision tree $T \in \mathcal{T}$ on input x, denoted by $\mathrm{D}(T, x)$, is the length of the path from the root to the leaf corresponding to x, or in other words, the number of queries made by the algorithm to the input x. For a function f, denote by $\mathcal{T}_f$ the set of all decision trees in $\mathcal{T}$ that compute f:

$$T \in \mathcal{T}_f \iff \forall x\colon T(x) = f(x).$$

The following two definitions deal with the worst-case deterministic and certificate decision tree complexity measures. Roughly speaking, the deterministic complexity measures the smallest possible cost over all decision trees that compute f.

▶ **Definition 2.1** (Worst-case deterministic complexity)**.** *The* ***worst-case deterministic*** *decision tree complexity of* $f\colon \{0,1\}^n \to \{0,1\}$*, denoted by* $\mathrm{D}(f)$*, is*

$$\mathrm{D}(f) = \min_{T \in \mathcal{T}_f} \max_{x \in \{0,1\}^n} \mathrm{D}(T, x).$$

The certificate complexity measures the minimal number of bits a decision tree has to query in order to *verify* the output of the function on a specific input (rather than to compute it). That is, we still want the decision tree to compute f correctly on all inputs, but we measure its complexity only with respect to a fixed input.

▶ **Definition 2.2** (Worst-case certificate complexity)**.** *The* ***certificate*** *complexity of* f *on input* x *is*

$$\mathrm{C}(f, x) = \min_{T \in \mathcal{T}_f} \mathrm{D}(T, x),$$

The ***worst-case certificate*** *complexity of* f *is* $\mathrm{C}(f) = \max_x \mathrm{C}(f, x)$*.*

Throughout this paper we use the term "decision tree" and "algorithm" synonymously. Since in the decision tree model we are only interested in the number of queries made by a given algorithm, the two terms are equivalent. We also use the terms "certificate complexity", "queries made by an algorithm with a certificate" and "queries made by any algorithm" synonymously. "Certificate complexity" and "an algorithm with a certificate" are synonyms since when arguing that the certificate complexity of a given function is some value, q one has to prove that some algorithm with access to a certificate only has to make q queries. This is the same as "queries made by any algorithm" since the certificate complexity is the decision tree with the lowest complexity, that is the algorithm that makes fewest queries.

We proceed with the randomized variants of the above two definitions. In randomized complexity it is common to talk about zero-error (Las Vegas), one-sided and two-sided (BPP style) algorithms. We will discuss only the latter for simplicity. There are two (equivalent) ways to view a *randomized* decision tree: one as a tree that has vertices that are coin flips. The other, which we will mostly use, as a probability distribution Δ over the set of all deterministic decision trees $\mathcal{T}$. Given an input x, the algorithm first samples a decision tree from the distribution and then executes it. The randomized decision tree complexity of a distribution (i.e., a randomized decision tree) Δ on input x, denoted by $\mathrm{R}(\Delta, x)$, is the expected number of queries made by the chosen tree on input x.

A distribution Δ is *correct* for a function f if for any input x, a randomly chosen decision tree $T \leftarrow \Delta$ outputs the value $f(x)$ with probability $2/3$ over the choice of T.[6] We denote by $\boldsymbol{\Delta}_f$ the set of all distributions Δ that are correct for f:

$$\Delta \in \boldsymbol{\Delta}_f \iff \forall x\colon \Pr_{T\leftarrow\Delta}[T(x) = f(x)] \geq 2/3.$$

Analogously to the deterministic setting, the worst-case randomized complexity measures the smallest possible (expected) cost over all randomized decision trees that compute f.

▶ **Definition 2.3** (Worst-case randomized complexity)**.** *The* ***worst-case randomized*** *decision tree complexity of f, denoted by $\mathrm{R}(f)$, is*

$$\mathrm{R}(f) = \min_{\Delta\in\boldsymbol{\Delta}_f} \max_x \underbrace{\mathop{\mathbb{E}}_{T\leftarrow\Delta}[\mathrm{D}(T,x)]}_{\mathrm{R}(\Delta,x)}.$$

For randomized certificate complexity, we still want the decision tree to be correct on every input with high probability, but the complexity is measured only on a fixed input.

▶ **Definition 2.4** (Worst-case randomized certificate complexity)**.** *The* ***randomized certificate*** *complexity of a function f on input x is*

$$\mathrm{RC}(f,x) = \min_{\Delta\in\boldsymbol{\Delta}_f} \mathop{\mathbb{E}}_{T\leftarrow\Delta}[\mathrm{D}(T,x)],$$

The worst-case randomized certificate complexity of f is $\mathrm{RC}(f) = \max_x \mathrm{RC}(f,x)$.

2.1 Instance Complexity and Optimality

Adapting the general notion of instance optimality (see Definition 1.1) to the setting of query complexity is done by fixing a class of query algorithms (say, deterministic or randomized ones) and then asking whether there exists an algorithm in this class that is "the best" on every input. Such an algorithm must have complexity at most a constant factor larger than the complexity of every other algorithm on *every* input. In particular, it has to be at most a constant factor worse than an algorithm that is designed for any specific x^*, and if it finds any inconsistencies between the given input x and x^*, reads the whole input. This leads us to the observation that a sufficient and necessary condition for instance optimality of a function in the query model is the existence of a deterministic (resp. randomized) decision tree whose complexity is at most a constant factor larger than the deterministic (resp. randomized) *certificate* decision complexity of the function per input.

The strongest notion one could hope for is what we call *strict* instance optimality, where the certificate does not help, even in low order constant factors (i.e., the optimality ratio is 1 and there is no additive term). More precisely, there exists a deterministic decision tree such that on every input x the decision tree complexity on x is upper bounded by the certificate decision tree complexity of f on input x.

▶ **Definition 2.5** (Strict optimality)**.** *A function f is* strict *instance optimizable if there exists a deterministic decision tree T such that for every input x:*

$$\mathrm{D}(T,x) = \mathrm{C}(f,x).$$

[6] The value of $2/3$ is arbitrary, since we can get any constant by amplification (i.e., by sampling several decision trees from Δ, executing them and taking the majority result).

Being optimal on every input without any overhead is a very strong requirement (and indeed not many non-trivial functions satisfy it; see below), so we relax it by allowing multiplicative and additive slack in the overhead for the decision tree without the certificate. For this we need to talk about a sequence of functions.

▶ **Definition 2.6** (D-Instance optimality). *A sequence of functions $f = \{f_n\}_{n\in\mathbb{N}}$ is* D-instance optimizable *if there exists a sequence of deterministic decision trees $T = \{T_n\}_{n\in\mathbb{N}}$, where each $T_n \in \mathcal{T}_{f_n}$, and (universal) constants $c_1, c_2 \geq 0$ such that on every input $x \in \{0,1\}^n$, it holds that*

$$\mathrm{D}(T_n, x) \leq c_1 \cdot \mathrm{C}(f_n, x) + c_2.$$

▶ **Definition 2.7** (R-Instance optimality). *A sequence of functions $f = \{f_n\}_{n\in\mathbb{N}}$ is* R-instance optimizable *if there exists a sequence of distributions over decision trees $\Delta = \{\Delta_n\}_{n\in\mathbb{N}}$, where each $\Delta_n \in \boldsymbol{\Delta}_{f_n}$, and (universal) constants $c_1, c_2 \geq 0$ such that on every input $x \in \{0,1\}^n$, it holds that*

$$\mathrm{R}(\Delta_n, x) \leq c_1 \cdot \mathrm{RC}(f_n, x) + c_2.$$

We refer to the *constant factor* c_1 above as the *optimality ratio*. The above deterministic (resp. randomized) algorithm is called *instance optimal*.

While we mostly talk about being within a constant of the best on any instance we will also consider being within some function $g(n)$. If we have an algorithm T such that for $g(n)$ for every input $x \in \{0,1\}^n$ it hold that

$$\mathrm{D}(T, x) \leq g(n) \cdot \mathrm{C}(f_n, x),$$

then we say that f is within g of being instance optimizable.

2.2 Unlabeled Complexity and Optimality

We introduce a new complexity measure for decision trees that we call *unlabeled certificate complexity*. Roughly speaking, the unlabeled complexity is the amount of queries an algorithm that is given some *permutation* of the certificate needs to perform.

This notion only makes sense for functions that are invariant under some permutation group Γ, such as symmetric properties or graph properties. In the case of symmetric functions, we consider the group of all permutations over the inputs, and in the case of graph properties, we focus on the group of permutations over the vertices (i.e., all isomorphic copies of the graph). We denote by Γ the group of permutations.

The idea behind the notion is to model the situation where the algorithm has a lot of knowledge about the input, such as the graph structure (e.g. the degree sequence), but not the actual labels of the graph. We refer to a decision tree that gets such a certificate as an *unlabeled certificate* decision tree.

As before, we expect an unlabeled decision tree T to correctly compute f on every input, but we will measure its complexity over all inputs that are consistent with the certificate under the given group of permutations.

▶ **Definition 2.8** (Worst-case unlabeled certificate complexity). *The* ***unlabeled certificate*** *complexity of a function f that is invariant under Γ on input x is*

$$\mathrm{AC}(f, \Gamma, x) = \min_{T\in\mathcal{T}_f} \max_{\pi\in\Gamma} \mathrm{D}(T, \pi(x)).$$

The ***worst-case certificate complexity*** *of f under Γ is $\mathrm{AC}(f, \Gamma) = \max_x \mathrm{AC}(f, \Gamma, x)$.*

For randomized unlabeled certificate complexity, the definition is analogous but we allow the decision tree to be randomized and to err.

▶ **Definition 2.9** (Worst-case randomized unlabeled certificate complexity)**.** *The* ***randomized unlabeled certificate*** *complexity of a function f invariant under Γ on input x is*

$$\mathrm{RAC}(f,\Gamma,x) = \min_{\Delta\in\boldsymbol{\Delta}_f} \max_{\pi\in\Gamma} \mathop{\mathbb{E}}_{T\leftarrow\Delta}[\mathrm{D}(T,\pi(x))],$$

The ***worst-case randomized unlabeled certificate complexity*** *of f under Γ is* $\mathrm{RAC}(f,\Gamma) = \max_x \mathrm{RAC}(f,\Gamma,x)$.

It follows immediately that the unlabeled certificate complexity of a function is lower bounded by its certificate complexity and upper bounded by its deterministic complexity. That is, for any Γ such that f is invariant under Γ we have

$$\mathrm{C}(f) \le \mathrm{AC}(f,\Gamma) \le \mathrm{D}(f).$$

The same holds in the randomized case:

$$\mathrm{RC}(f) \le \mathrm{RAC}(f,\Gamma) \le \mathrm{R}(f).$$

Throughout the paper we let the default Γ be the set of all graph isomorphisms. That is, if our function is a graph property, and Γ is not specified, then Γ is the set of all graph isomorphisms.

Given this new notion of unlabeled certificates one can define instance optimality with respect to it. Here, we are trying to compete with the best algorithm that is given some permutation of the certificate, rather than the exact certificate as in the definitions of Section 2.1. If an algorithm is instance optimal in this sense, this means that even the knowledge about the graph structure does not help improve the performance.

Namely, we can ask whether having access to a permutation of a certificate reduces the complexity by more than a constant factor compared to a decision tree that gets no certificate at all.

▶ **Definition 2.10** (Unlabeled instance optimality)**.** *A sequence of functions $f = \{f_n\}_{n\in\mathbb{N}}$ invariant under a sequence of family of permutations $\{\Gamma_n\}_{n\in\mathbb{N}}$, is* unlabeled D-instance optimizable *(resp.* unlabeled R-instance optimizable*) if there exists a sequence of deterministic (resp. randomized) decision trees $T = \{T_n\}_{n\in\mathbb{N}}$, where each $T_n \in \mathcal{T}_{f_n}$, and constants $c_1, c_2 \ge 0$ such that on every input x, it holds that*

$$\mathrm{D}(T,x) \le c_1 \cdot \mathrm{AC}(f_n,\Gamma_n,x) + c_2 \qquad (\textit{resp. } \mathrm{R}(T,x) \le c_1 \cdot \mathrm{RAC}(f_n,\Gamma_n,x) + c_2)$$

We can also compare the labeled certificate to the unlabeled certificate. A function is labelling instance optimal if the number of queries required to evaluate the function given an unlabeled certificate is at most a constant times the number queries required to evaluate the function with a labeled certificate.

▶ **Definition 2.11** (labelling instance optimality)**.** *A sequence of functions $f = \{f_n\}_{n\in\mathbb{N}}$ invariant under a sequence of family of permutations $\{\Gamma_n\}_{n\in\mathbb{N}}$, is* labelling D-instance optimizable *(resp.* labelling R-instance optimizable*) if there exists a sequence of deterministic (resp. randomized) decision trees $T = \{T_n\}_{n\in\mathbb{N}}$, where each $T_n \in \mathcal{T}_{f_n}$, and constants $c_1, c_2 \ge 0$ such that on every input x, it holds that*

$$\mathrm{AC}(T,x) \le c_1 \cdot \mathrm{C}(f_n,\Gamma_n,x) + c_2 \qquad (\textit{resp. } \mathrm{RAC}(T,x) \le c_1 \cdot \mathrm{RC}(f_n,\Gamma_n,x) + c_2)$$

2.3 Additional Definitions & Known Relations

Graph properties. In some of our result we will be interested in graph properties. A graph property is a set of graphs closed under graph isomorphism. That is Ψ is a graph property if for every graph $G = (V, E)$ and every permutation π over V, it holds that $G \in \Psi$ if and only if $\pi(G) \in \Psi$, where $\pi(G) = (V, E')$ and $E' = \{(\pi(u), \pi(v)) \mid (u, v) \in E\}$).

Sensitivity. Two central notions in the analysis of Boolean functions are the *sensitivity* and *block sensitivity*, introduced by Cook, Dwork and Reischuk [14] and Nisan [40], respectively. They both measure how sensitive a function f is to changes in its input. Let $f\colon \{0,1\}^n \to \{0,1\}$ be a function and $x \in \{0,1\}^n$ be an input. The *sensitivity* of f on x, denoted $\mathsf{s}(f, x)$, is the number of bit positions i such that $f(x) \neq f(x \oplus e_i)$, where $e_i \in \{0,1\}^n$ is a vector that is 1 at coordinate i and 0 everywhere else. The sensitivity of f, denoted by $\mathsf{s}(f)$, is $\max_x \mathsf{s}(f, x)$.

Block sensitivity is a useful generalization of sensitivity. The *block sensitivity* of f on x, denoted $\mathsf{bs}(f, x)$, is the maximum number t such that there is collection $B = \{B_1, \ldots, B_t\}$ of *disjoint* subsets of $[n]$ with $f(x) \neq f(x \oplus B_i)$ for all $i \in [t]$. The overall block sensitivity $\mathsf{bs}(f)$ of f is the maximum of $\mathsf{bs}(f, x)$ over all x. Note that sensitivity is just block sensitivity where the blocks are restricted to be of size 1.

It is not hard to see that $\mathsf{s}(f) \leq \mathsf{bs}(f) \leq \mathrm{C}(f)$. The biggest known gap between $\mathsf{s}(f)$ and $\mathsf{bs}(f)$ is quadratic (see Rubinstein [45] and Ambainis and Sun [6]). Showing that this gap is at most polynomial has been a major open problem for many years, until it had been resolved recently by [32], who showed that for all f, $\mathsf{bs}(f)$ is bounded by $\mathsf{s}(f)^4$.

More relations. In general, all worst-case complexity measures discussed above are known to have a polynomial relationship and figuring out the precise relationships is a major research program.

▶ **Proposition 2.12.** *For every Boolean function f we have:*

1. $\mathrm{RC}(f) \leq \mathrm{C}(f) \leq \mathrm{D}(f) \leq n$ *and* $\mathrm{R}(f) \leq \mathrm{D}(f)$.
2. $\mathrm{C}(f) \leq \mathsf{s}(f) \cdot \mathsf{bs}(f)$ *(see [40] and [12, Theorem 2]).*
3. $\mathrm{D}(f) \leq \mathsf{s}(f) \cdot \mathsf{bs}(f)^2 \leq \mathsf{bs}(f)^3$ *(see [12, Corollary 1]).*
4. $\mathsf{bs}(f) \leq \mathsf{s}(f)^4$ *(see [32]).*
5. $\mathrm{D}(f) \leq \mathrm{C}(f)^2$ *(see [10, 49] and, for example, [34, Theorem 14.3]).*
6. $\mathsf{bs}(f) \leq 3\mathrm{RC}(f) \leq 3\mathrm{R}(f)$ *(see [40] and [1]) and* $\mathsf{bs}(f) \leq \mathrm{D}(f)$.

▶ **Proposition 2.13.** *For every* monotone *Boolean function f it holds that* $\mathrm{C}(f) = \mathsf{s}(f) = \mathsf{bs}(f)$ *(see [40] [12, Proposition 3]) and hence* $\mathrm{D}(f) \leq \mathsf{s}(f)^2$.

Yao's Minimax Principle. When proving lower bounds on the randomized query complexity of a function f it is often easier to rephrase the problem by applying Yao's [52] Minimax Principle (see [39] Chapter 2). In words, it says that in order to prove a lower bound on the randomized decision tree complexity, it is enough to come up with one particularly bad distribution over *inputs* which fools all deterministic algorithms. Often Yao's Minimax principle is used by saying that if t expected queries are required to distinguish between two input distributions with high probability (where the function outputs 1 where the input is chosen from one distribution but when the input is chosen from the second distribution the function outputs 0), then any deterministic algorithm must perform at least t queries.

3 Properties Of Instance Optimal Functions

This section is devoted to showing several results regarding properties of instance optimal function in both the deterministic and randomized setting. We prove Theorem 1.2 and Theorem 1.3, as well as showing, given a truth table representing a function, and an algorithm, how to test if the algorithm is instance optimal in time polynomial in the size of the truth table. Lastly we give a characterization of which functions are R-instance optimal in the proximity property testing model.

3.1 Deterministic Instance Optimality

We show that there are functions which are D-instance optimizable and there are functions which are not. We also prove that there are no non-trivial monotone functions whose optimality ratio is 1 (i.e., *strict* D-instance optimizable). The arguments in this section are simple and provide good examples for the notions discussed.

Let $\mathrm{Parity}\colon \{0,1\}^n \to \{0,1\}$ be the parity function on n variables, defined as $\mathrm{Parity}(x) = \bigoplus_{i=1}^{n} x_i$. Assume that n is a power of 2 and let $\mathrm{Addr}\colon [n] \times \{0,1\}^n \to \{0,1\}$ be the addressing function defined as $\mathrm{Addr}(i, x_1, \ldots, x_n) = x_i$. Both of these functions are D-instance optimizable. In both cases the argument is that on all instances the certificate complexity has the same value (n and $1 + \log n$, respectively) and it is possible to achieve it with a straightforward deterministic algorithm.

▶ **Example 3.1** (Item 2 of Theorem 1.2)**.** The Parity function and the Addr function are strict D-instance optimizable.

Proof. The proof for both Parity and Addr being strict D-instance optimizable rely on a common statement that follows from the definitions of deterministic and certificate complexity: given a function $f\colon \{0,1\}^n \to \{0,1\}$, for any decision tree $T \in \mathcal{T}_f$ and any input $x \in \{0,1\}^n$, it holds that $\mathrm{D}(T, x) \geq \mathrm{C}(f, x)$. For the Parity and Addr functions the straightforward trees $T_{\mathrm{Parity}} \in \mathcal{T}_{\mathrm{Parity}}$ (that reads the whole input) and $T_{\mathrm{Addr}} \in \mathcal{T}_{\mathrm{Addr}}$ (that reads the address part plus the resulting address) satisfy $D(T_{\mathrm{Parity}}, x) = n$ and $D(T_{\mathrm{Addr}}, x) = \log n + 1$ for every $x \in \{0,1\}^n$. Thus, it remains to prove that $\mathrm{C}(\mathrm{Parity}, x) \geq n$ and $\mathrm{C}(\mathrm{Addr}, x) \geq \log n + 1$. We do this by showing that any decision tree (that is always correct) must make at least n queries for Parity and at least $\log n + 1$ queries for addressing. We show that for any algorithm that makes less queries, we can change bits that the algorithm does not query, and this will change the value of the function, thus making the algorithm err.

For Parity, consider an algorithm that on some input x queries less than n input locations. Since there is a location that the algorithm does not query and the function's output depends on all locations, by flipping the value of an unqueried bit, the output of the function must change, but the algorithm cannot notice this and hence it is wrong either on x or on x with the flipped bit.

For Addr, consider an algorithm that on some input x reads $k \leq \log n$ values. These k bits can either fully fix the index i, but then there are two possible values for the bit in the i-th position (which determine the value of the function) that the algorithm cannot distinguish. The other case is that the algorithm did not query the value of the index i in full. Denote by $\ell > 0$ the number of bits of i that the algorithm did not query. There are 2^ℓ possibilities for the value of i. The algorithm can query only $k - (\log n - \ell)$ locations out of the possible n. Since $k - (\log n - \ell) < 2^\ell$, there must be a way to assign values to the unread location so as to fool the algorithm to output the wrong answer. ◀

We show that there is a *monotone* function that is D-instance optimizable (unlike the parity and addressing functions that are not monotone). However, the function is not *strict* D-instance optimizable. The function is the majority function $\mathrm{Maj}\colon \{0,1\}^n \to \{0,1\}$ defined as

$$\mathrm{Maj}(x_1,\ldots,x_n) = 1 \iff \sum_{i=1}^{n} x_i \geq \frac{n}{2}.$$

▶ **Example 3.2** (Item 1(i) of Theorem 1.2)**.** The majority function is D-instance optimizable.

Proof. In order to determine the output of Maj on an input $x \in \{0,1\}^n$, every decision tree has to find (at least) $n/2+1$ locations (in x) that have the same value, 0 or 1. Since the algorithm is deterministic, this can take up to n queries in the worst case. On the other hand, an algorithm with a certificate for x can query exactly $n/2+1$ locations that have the same value. Any algorithm that has a certificate for x that queries at most $n/2$ queries can be fooled to output the wrong answer. We get that for every input x it holds that $C(\mathrm{Maj}, x) = n/2+1$ and $D(\mathrm{Maj}, x) \leq n$, as needed. ◀

To complete the picture, we give a (monotone) function that is not D-instance optimizable.

▶ **Example 3.3** (Item 1(ii) of Theorem 1.2)**.** The OR function (that outputs '1' iff the Hamming weight of the input is at least 1) is not D-instance optimizable.

Proof. An algorithm with a certificate can make just one query on every input with Hamming weight 1 (to a location where there is supposedly a '1'). However, every decision tree (not getting a certificate) that is correct on every input must query all the coordinates on one of these inputs, as otherwise we can fool it to output the wrong answer. This is done by planting '0's at the points it queries and a '1' at a point that it does not query. ◀

Strict instance optimality. Both the addressing and parity functions were shown to be *strict* D-instance optimizable (the algorithm with the certificate could not outperform the one without the certificate even by a multiplicative small constant term). On the other hand, in our analysis of the instance optimality of the majority function, we showed that an algorithm with a certificate may perform twice as fast than the one that does not have the certificate. Namely, the optimality ratio of the majority function is 2.

These examples are no coincidence: No non-trivial (that is not constant or dictatorship) *monotone* function can be strictly D-instance optimizable.

▶ **Lemma 3.4** (Item 3 of Theorem 1.2)**.** *There are no* strict *D-instance optimizable* monotone *functions except those that depend on* 0 *or* 1 *variables.*

Proof. Consider the DNF of the function i.e. the OR of all the minimal terms that make the function '1'. Suppose that the function has only one term. It cannot be of length 1, since the function does not depend on one variable. Therefore, if all the variables in the term but one are set to '1', the certificate size is 1, but for any decision tree all the variables in the term must be read and if the one set to '0' is the last one, then the complexity on that input is greater than 1.

So suppose that there are at least two terms in the conjunction. Due to the monotonicity of the function they are not on exactly the same sets of variables. Consider the two instances, one satisfying the first term and the other satisfying the second one (and nothing else). A certificate verifier just needs to access the variables of one term, but a deterministic (or even probabilistic) will have to ask on a variable that appears in one term but not the other and hence will not have the same complexity and the instance satisfying the other term. ◀

3.2 Randomized Instance Optimality

The question we consider is whether randomness changes anything w.r.t. instance optimality. It is not hard to see that the parity function and the addressing function are not only D-instance optimizable but they are also R-instance optimal (the proof of Example 3.1 actually shows it). However, the situation changes for the majority function: we show that, in contrast to Example 3.2, the majority function is *not* R-instance optimizable. We complement the picture in this section with an example of a *monotone* function that is also R-instance optimal.

▶ **Lemma 3.5** (Item 2 of Theorem 1.3, first part)**.** *The majority function is* not *R-instance optimizable.*

Proof. Given that the deterministic certificate size of majority is always of size $\Theta(n)$, we must use the fact that the algorithm may err. We will give a distribution $\mathcal{D}$ on the inputs such that any algorithm that does not get a certificate (but may be tailored to inputs that come from the distribution $\mathcal{D}$) must query $\Omega(n)$ bits in order to be correct with probability at least $2/3$. On the other hand, for any input x from this distribution $\mathcal{D}$, an algorithm with a certificate for x only needs to query $O(\sqrt{n})$ of the bits to be correct with high probability.

For $x \in \{0,1\}^n$ let $wt(x)$ stands for the Hamming weight of x. The hard distribution on inputs $x \in \{0,1\}^n$ is to choose uniformly at random from the set $wt(x) \geq n/2 + \sqrt{n}$ and the set $wt(x) \leq n/2 - \sqrt{n}$ (in the former Maj is 1 and in the latter it is 0). Consider a randomized decision tree for Maj that is correct on this distribution with probability at least $2/3$, i.e. distinguishes between the case that $wt(x) \geq n/2 + \sqrt{n}$ and the case that $wt(x) \leq n/2 - \sqrt{n}$ with probability at least $2/3$.

We show that this task requires $\Omega(n)$ queries on a uniform input of Hamming weight $n/2 + \sqrt{n}$. First, by symmetry, notice that the best strategy for the algorithm is to sample random locations in x and query them. Second, the lower bound follows from the fact that $\Omega\left(1/\varepsilon^2\right)$ samples are needed to distinguish (with constant probability) a biased random coin that outputs heads with probability $1/2 + \varepsilon$ from one that outputs heads with probability $1/2 - \varepsilon$.[7]

On the other hand, given a certificate for an x of Hamming weight at least $n/2 + \sqrt{n}$, there is a randomized algorithm that makes $q = 10\sqrt{n}$ queries and outputs the right answer with probability at least $2/3$. The algorithm, given a certificate $x^* \in \{0,1\}^n$ and an input $x \in \{0,1\}^n$, does:

1. Query x at q random locations i where x^*_i is 1.
2. If all queries return 1, output 1.
3. Otherwise, read the whole input and output the majority value.

The algorithm makes a mistake only if at Step 2 all queries returned '1', but the Hamming weight of x was less than $n/2$. This can happen only if there is a set $I \subseteq [n]$ of coordinates of size at least $|I| \geq \sqrt{n}$ such that $x^*_i = 1$ but $x_i = 0$ for every $i \in I$, *and* the algorithm did not query at any of them. The probability that this happens is at most

$$\left(1 - \frac{\sqrt{n}}{n}\right)^q = \left(1 - \frac{1}{\sqrt{n}}\right)^{10\sqrt{n}} \leq 1/3.$$

Thus, this algorithm outputs the correct result with probability at least $2/3$, as needed. ◀

[7] See, for example, Claim 5.6 here: `http://www.tau.ac.il/ mansour/advanced-agt+ml/scribe5-lower-bound-MAB.pdf` (Accessed: June 2018).

We give an explicit monotone function that is R-instance optimizable. We call this function the *lexicographic threshold* function.

▶ **Lemma 3.6** (Item 3 of Theorem 1.3)**.** *There exists an explicit monotone function that is R-instance optimizable.*

Proof. Consider the lexicographic order of strings in $\{0,1\}^n$ under $\geq_{\text{lex}}$. Let $b = 1010\ldots1010 \in \{0,1\}^{2n}$ and define $\text{LTF}_n\colon \{0,1\}^{2n} \to \{0,1\}$ as $\text{LTF}_n(x) = 1$ if and only if $x \geq_{\text{lex}} b$. Denote by $b_{[i]}$ the substring $b_1 \ldots b_i$ where b_i is the i-th bit of b from left to right.

We show that the randomized query complexity of LTF is roughly equal to the randomized query complexity of LTF given access to a certificate for the input. That is, we show that any randomized query algorithm for LTF that has access to a certificate for the input must work (roughly) as hard as an algorithm that does not have access to a certificate.

Our algorithm, that we denote by P_n, is to read bit by bit from left to right. Denote by $x_{[i]}$ the substring read by the algorithm (ordered left to right) until iteration i. Initially, $x_{[0]} = \perp$ and $i = 0$. At iteration $1 \leq i \leq 2n-1$, the algorithm returns 1 if $x_{[i]} >_{\text{lex}} b_{[i]}$, returns 0 if $x_{[i]} <_{\text{lex}} b_{[i]}$ and continues to the next iteration otherwise (i.e., if $x_{[i]} =_{\text{lex}} b_{[i]}$). If the algorithm reaches $i = 2n$, it returns 1 (since the last bit of b is 0). Note that the algorithm is deterministic. The number of queries that this algorithm does depends on the input and is $2n-1$ in the worst case.

We prove that for every input x it holds that $\mathbb{E}[P_n(x)] \leq 2\text{RC}(\text{LTF}, x)$.

For any x let $\mathsf{pref}(x)$ be the length of the prefix of x that algorithm P_n reads before it stops. In other words, it is the length of the prefix of x that agrees with the string b. Consider a pair of locations $2i-1$ and $2i$ that are smaller than $\mathsf{pref}(x)$. If the probability that A reads at least one of them is smaller than $2/3$, then algorithm A may err with probability at least $1/3$. If instead of input x it is given input x' that is flipped in locations $2i-1$ or $2i$, then the value of $\text{LTF}_n(x') \neq \text{LTF}_n(x)$, but A distinguishes the two inputs with probability less than $2/3$. ◀

▶ Remark 3.7 (Quantum instance optimality). In the quantum case, the Parity function is *not* instance optimal. Based on Grover's algorithm [29], it is known that *quantum* certificate complexity is exactly the square root of randomized certificate complexity (up to a constant factor). An explicit statement appears in Aaronson [1], where this claim is shown to be true on an input-by-input basis. On the other hand, computing Parity requires $n/2$ queries (quantumly) as shown by Farhi et al. [20] and Beals et al. [8].[8]

3.2.1 Sensitivity and Instance Optimality

We show that a function for which the decision tree complexity for every input is (roughly) equal to its sensitivity, then the function is instance optimizable. This is not true in the opposite direction.

▶ **Lemma 3.8.** *Let f be an n-input Boolean function. If there is a (randomized) decision tree for f s.t. for any x the complexity on input x is $\Theta(\mathsf{s}(f,x))$, then this algorithm is instance optimal.*

Proof. Let f be an n-input Boolean function. Assume that there is a decision tree algorithm (randomized or deterministic) whose complexity for every input x is $O(\mathsf{s}(f,x))$. For the deterministic case we show that any algorithm A that does not err must query $\mathsf{s}(f,x))$

[8] We thank Scott Aaronson for bringing these results to our attention.

location on x: suppose that there is an index i such that $f(x) \neq f(x \oplus e_i)$ that A does not query x_i. Then, on input $x \oplus e_i$ the algorithm A makes the same decision as on input x and thus must err.

For the randomized case, we show that any algorithm (that does not err with probability larger than $1/3$ on any input) must query at least as many points in expectation as $2\mathsf{s}(f,x)/3$ on input x before deciding the output of the function.

Consider two input distributions: one with the point x and the other one is on $\mathsf{s}(f,x)$ points of the form $x \oplus e_i$ for a random $i \in [n]$ such that $f(x) \neq f(x \oplus e_i)$. If an algorithm makes at most j queries, then the probability of distinguishing the two distributions is at most $j/\mathsf{s}(f,x)$. That is, the output distribution of such an algorithm when given one or the other distribution can differ by at most $j/\mathsf{s}(f,x)$. Therefore, if the expected number of queries the randomized algorithms makes on x is less than $2\mathsf{s}(f,x)/3$, then the expected difference is less than $2/3$ which means that the algorithm will err with probability at least $1/3$. ◀

From this lemma we can obtain many results concerning the instance optimality of many algorithms. For instance, we can recover our results for the parity, addressing and lexicographic threshold functions.

The lemma actually holds for block-sensitivity as well (with essentially the same proof). But this still does not give a tight characterization, as there are functions where the block-sensitivity is lower than the randomized certificate complexity (by some polynomial factor) [1, 24].

3.2.2 Random Functions are Instance Optimizable

Consider the uniform distribution of Boolean functions, i.e. selecting at random one from the set of size 2^{2^n} functions $\{0,1\}^n \to \{0,1\}$. A random Boolean function $f\colon \{0,1\}^n \to \{0,1\}$ is instance optimizable with probability $1 - 2^{-cn}$ for some constant c and the naive algorithm that reads the *entire input* is an instance optimal algorithm. A proof follows.

What we need to show is that for such a random function on *all* inputs any algorithm must read a large fraction of the bits.

For input $x \in \{0,1\}^n$ consider all $\binom{n}{2}$ elements at (Hamming) distance 2. We have that except with probability $\ll 2^{-n}$ we will have at least $\binom{n}{2}/4$ values x' of distance 2 from x such that $f(x) \neq f(x')$.

Consider first a deterministic algorithm for x that queries k locations. If the algorithm is not to err, then it must cover all those x''s in the sense that it queries at least one of the two bits where x and x' differ. Now if it queries only $k \in o(n)$ locations it can cover at most $\binom{k}{2} + k(n-k)$ such x', but this is much smaller than $\binom{n}{2}/4$.

To get a lower bound for a randomized algorithm, consider the set of locations that have probability at least $1/8$ to be queried. If this set does not cover an x' at distance 2 from x where $f(x') \neq f(x)$, then we have that the algorithms errs with probability at least $3/4$ when the input is x'. So again we get that this set should be large and the expected number of location queried is $\Theta(n)$. So, altogether, we get that on all inputs x an algorithm tailored for x must ask $\Omega(n)$ queries and therefore the naive algorithm is instance optimal. This proves Item 5 of Theorem 1.3.

▶ Remark 3.9. It would have been nice if we could show the above claim (that random functions are instance optimizable) by showing that such functions have high block sensitivity *on any input*. But note that to use the union bound we need to get probability less than 2^{-n} and that just won't work. Instead, we gave a direct proof.

3.2.3 A Separation From Deterministic Instance Optimality

As we have seen, deterministic instance optimality does not imply randomized instance optimality (e.g., the Majority function, Lemma 3.2 and 3.5). There are several examples for the other direction, utilizing cases where there is a difference between the worst case deterministic and randomized complexities. In particular the following example:

▶ **Lemma 3.10** (Item 2 of Theorem 1.3, second part)**.** *There exists a function that is R-instance optimal but not D-instance optimal.*

Proof. Let $g(x_1)$ (where $|x_1| = n$) be a function for which $\mathrm{D}(g) \in \Omega(n)$, $\mathrm{R}(g) \in o(n)$ and $\mathrm{C}(g) \in o(n)$ in the worst case. Such functions are known, for example Saks and Wigderson [46] or Ambainis et al. [5]. Let Parity_ℓ be the parity function of all the bits of the input string of length ℓ. Let x be the concatenation of x_1 and x_2, and let $f(x) = g(x_1) \oplus \mathrm{Parity}_\ell(x_2)$ where we set $\ell = |x_2| = \mathrm{R}(g)$. For this length we have $\mathrm{C}(\mathrm{Parity}, x_2) = \mathrm{R}(g)$ on all inputs x_2.

Regarding the deterministic complexity of f, we have that $\mathrm{D}(f) = \mathrm{D}(g) + \mathrm{D}(\mathrm{Parity}_\ell) \geq \mathrm{D}(g) \in \Omega(n)$. Similarly $\mathrm{C}(f, x) = \mathrm{C}(g, x_1) + \mathrm{C}(\mathrm{Parity}_\ell, x_2) \in o(n)$ and thus on the worst case input, the certificate outperforms the deterministic algorithm, and thus f is not D instance optimal.

On the other hand, in the randomized case: let Δ_g be the best randomized algorithm to evaluate g in the worst case. Let $\Delta_{\mathrm{Parity}_\ell}$ be the algorithm that evaluates Parity_ℓ by simply querying the entire input. Let Δ_f be the algorithm that evaluates f by doing both Δ_g and$\Delta_{\mathrm{Parity}_\ell}$. We have, for any x, $\mathrm{R}(\Delta_f, x) = \mathrm{R}(\Delta_g, x_1) + \mathrm{R}(\Delta_{\mathrm{Parity}_\ell}, x_2) \in O(\mathrm{R}(g))$. Similarly $\mathrm{RC}(f, x) = \mathrm{RC}(g, x_1) + \mathrm{RC}(\mathrm{Parity}_\ell, x_2) > \mathrm{RC}(\mathrm{Parity}_\ell, x_2) \in \Omega(R(g))$ and thus f is R-instance optimal. ◀

3.3 Instance Optimality and Graph Properties

In this section we study the instance optimizability of a specific set of functions, ones that test graph properties.[9] Our motivation comes from our example in Lemma 3.6 of a monotone function which is R-instance optimizable: the function is very "far" from being a graph property as the location of each bit greatly influences its effect on the outcome of the function. This raises the question of whether non-symmetry is necessary for R-instance optimizability of monotone functions.

As a first step, we ask the following question: if we restrict our attention to graph properties, then is it possible that having a certificate *always* "helps"? In Lemma 3.11 we show that it is not the case. Specifically, we give an explicit graph property that is instance optimal. The graph property that we use is the *scorpion* property. An n vertex graph $G = (V, E)$ is a *scorpion* graph if it contains 3 special vertices: a vertex b (body) of degree $n - 2$, a vertex t (tail) of degree 2 and a vertex s (sting) of degree 1. The tail is adjacent to both b and s. Edges in between the remaining $n - 3$ vertices in $V \setminus \{b, s, t\}$ may be present or not.

▶ **Lemma 3.11** (Item 4 of Theorem 1.3)**.** *The scorpion graph property is R-instance optimal.*

Proof. We first show that a deterministic algorithm can test whether a given graph is a scorpion graph with $O(n)$ queries. This is a result of Best et al. [9]. We provide a proof of this in Appendix A for completeness.

[9] We consider finite and undirected graphs with neither self loops nor parallel edges. A graph property is an invariant that depends only on the abstract structure of the graph (and not on specific graph representations). See Section 2.3.

▷ Claim 3.12 ([9]). Testing whether a given n vertex graph is a scorpion graph takes at most $O(n)$ queries.

To complete the proof we show that on any instance G the complexity is $\Omega(n)$:

▷ Claim 3.13. For any graph G the best randomized algorithm on G that is correct on all inputs with probability at least $2/3$ must make $\Omega(n)$ queries in expectation.

Proof. Consider first a graphs that is a scorpion. The sensitivity of such a graph is at least $n-1$: there is a unique vertex that is the body (of degree $n-2$). If any of its $n-2$ adjacent edges is missing, then the graph ceases to be a scorpion. So as in the proof of Lemma 3.8 any correct algorithm must make $\Omega(n)$ queries in expectation. Consider now a graph that is not a scorpion. We partition its vertices into $n/3$ groups of triples and know that if for any one of the triples we completely change the neighborhood the triple can become the b, t and s of the scorpion graph (and the corresponding graph will satisfy the property). So consider the following two distributions on graphs: one is simply G the other one starts with G, picks a triple at random and changes the neighborhoods so they become the body, tail and sting. Differentiating between these two distributions implies that the number of triples touched (i.e. at least one edge in their neighborhood is queried) is close to $n/3$, but since each edge belongs to at most two triples we get that $\Omega(n)$ edges must be queried in expectation. ◁

◀

3.3.1 Conjecture: monotone graph properties are not instance optimizable

One of the most famous conjectures in the literature of query complexity is the *evasiveness conjecture* (a.k.a. the Aanderaa-Rosenberg conjecture). Roughly speaking, the conjecture says that for any non-trivial graph property any deterministic decision tree algorithm must query at least a constant fraction of all the edges (in the worst case). This conjecture was resolved by Rivest and Vuillemin [44]. In another version, called the *Aanderaa-Karp-Rosenberg conjecture* (AKR conjecture), the specific constant is also conjectured to be 1. This was resolved for graphs of prime power order (number of vertices) by Kahn, Saks and Sturtevant [35].

Yao [52, §4, Question (2)] asked whether an analogue of the Rivest-Vuillemin result holds for monotone graph properties and *no-error* randomized algorithms. Namely, whether a constant fraction of edges must be queried in expectation for any non-trivial monotone graph property by any randomized decision tree algorithm. The first result related to this question was of Yao [53] who showed a lower bound of $\Omega(n \cdot \log^{1/12} n)$. This was improved by King [36] and then by Hajnal [30] to $\Omega(n^{5/4})$ and $\Omega(n^{4/3})$, respectively. The currently best lower bound is $\Omega(n^{4/3} \cdot \log^{1/3} n)$ of Chakrabarti and Khot [13].

If the algorithm is allowed to make two-sided errors the best lower bound we are aware of is that of Jain and Zhang [33]. One could also pose a conjecture analogous to Yao's for such algorithms:

▶ **Conjecture 3.14.** *Any randomized decision tree algorithm has to query a constant fraction of edges for any non-trivial monotone graph property P, even if it is allowed to make an error with probability $1/3$. That is, $R(P)$ is $\Theta(n^2)$ for any non-trivial monotone graph property P on n-vertex graphs.*

We make a conjecture regarding instance optimality of graph properties, namely that every *monotone* graph property is not R-instance optimizable. That is, a randomized algorithm that gets a certificate will always be better than any randomized algorithm (that has no

certificate) on *some* input. The examples we gave in previous sections do not rule out this conjecture: the scorpion property is not monotone, the lexicographic threshold function is not a graph property, and the majority function (which can be phrased as a graph property) is not R-instance optimizable. Recall that the majority function is D-instance optimal, so the conjecture is false for deterministic computation.

▶ **Conjecture 3.15.** *Every non-trivial monotone graph property is not R-instance optimizable.*

We do not know what is the relationship between Conjectures 3.14 and 3.15.

3.4 Composition of Instance Optimal Functions

Given boolean function $f\colon \{0,1\}^n \to \{0,1\}$ and $g\colon \{0,1\}^m \to \{0,1\}$. The composition of f and g, denoted $f \circ g\colon \{0,1\}^{nm} \to \{0,1\}$, is defined as the value of $f(g(\vec{x_1}), ..., g(\vec{x_n}))$ where each $\vec{x_i}$ is an independent vector of m bits. Most classical notions in query complexity (such as deterministic and certificate complexities, degree, sensitivity and more) are known to behave well under composition (see for example Tal [48, Lemma 3.1]). In this section we show that instance optimality, on the other hand, does not compose. That is, the composition of two instance optimal functions is not necessarily instance optimal.

▶ **Lemma 3.16** (Item 4 of Theorem 1.2 and Item 6 of Theorem 1.3)**.** *There exist two functions that are both strictly D-instance optimal and R-instance optimal, but their composition results in a function that is neither R nor D instance optimal.*

Proof. Let $f\colon \{0,1\}^n \to \{0,1\}$ have input of the form: $(a_1, b_1), ..., (a_{n/2}, b_{n/2})$. The function f outputs b_i where i is the first index for which $a_i = 1$. We also define $a_{n/2} = 1$, regardless of the true input. The lemma follows immediately from the claims below.

▷ Claim 3.17. The function f is strictly D-instance optimal.

Proof. Consider the naive algorithm that queries each a_i in order until it finds an i for which $a_i = 1$ and then queries b_i. WLOG suppose the naive algorithm outputs 0. Suppose towards a contradiction that there exists an algorithm A that makes at most k queries for some input X, while the naive algorithm makes $k+1$ queries. We will show that by changing the values of bits that A did not query we can make A output the wrong value. If A makes at most k queries one of two things must be true:

1. For some $i < k$, A does not query any one of a_i and b_i.
2. A doesn't query a_k or A doesn't query b_k. If Item 1 does not occur, then this must occur (since if for every $i < k$, A queries either a_i or b_i, then querying both a_k and b_k would require $k+1$ total queries).

For the first case, if $a_i = b_i = 1$ then A makes an error. For the second case, we have two options: either A queries a_k or A queries b_k. Suppose A queries a_k. Note that $a_k = 1$ since the naive algorithm makes $k+1$ queries. If $b_k = 1$ then A has made an error. For the second option, suppose A queries b_k. If $a_k = 0$ but $a_{k+1} = b_{k+1} = 1$ then A has made an error. Note that the maximum number of queries the naive algorithm makes is $k + 1 = \frac{n}{2}$ since we know automatically that $a_{n/2} = 1$. ◁

▷ Claim 3.18. The function f is R-instance optimal.

Proof sketch. This is very similar to the proof of Lemma 3.6, that the lexicographic function is R-instance optimal. Essentially, any algorithm that is correct on 2/3 of the input must query the a_i's in order, since otherwise the input can be set to fool the algorithm. ◁

▷ Claim 3.19. For any algorithm $\Delta \in \mathbf{\Delta}_{f\circ f}$ (i.e., that computes $f \circ f$ on all inputs correctly with probability 2/3) there exists an input, x, for which $\mathrm{C}(f \circ f, x) \in O(n)$ while $\mathrm{R}(\Delta, x) \in \Omega(n^2)$.

The above claim implies that instance optimality does not compose in both the randomized and the deterministic case. This is true since our upper bound uses a deterministic decision tree while the lower bound is for any randomized decision tree.

Proof. Consider $f' \circ f''$. where both f' and f'' are f as defined above. Denote the input of f' by $[a_1^{\mathsf{out}}(\vec{x}_{1,1}), b_1^{\mathsf{out}}(\vec{x}_{1,2})], ..., [a_{n/2}^{\mathsf{out}}(\vec{x}_{n/2,2}), b_{n/2}^{\mathsf{out}}(\vec{x}_{n/2,2})]$, and similarly the input of each f'' by $\vec{x}_{i,j} = (a_1^{\mathsf{in}}, b_1^{\mathsf{in}}), ..., (a_{n/2}^{\mathsf{in}}, b_{n/2}^{\mathsf{in}})$. Note that a_i^{in} and b_i^{in} consists of a single bit. On the other hand $a_i^{\mathsf{out}}(\vec{x}_{i,1})$ and $b_i^{\mathsf{out}}(\vec{x}_{i,2})$ are functions on n bits.

Consider an input of the following form: for every i, $\vec{x}_{i,2}$ has $a_1^{\mathsf{in}} = 1$ (and thus for all i, $b_i^{\mathsf{out}}(\vec{x}_{i,2})$ can be computed in $O(1)$). Suppose that for all $i \neq \frac{n}{2}$, $\vec{x}_{i,2}$ has $b_1^{\mathsf{in}} = 0$, (and thus for all $i \neq \frac{n}{2}$, $b_i^{\mathsf{out}}(\vec{x}_{i,2}) = 0$), and $\vec{x}_{n/2,2}$ has $b_1^{\mathsf{in}} = 1$ (and thus $b_{n/2}^{\mathsf{out}}(\vec{x}_{n/2,2}) = 1$). Suppose that for every i, $\vec{x}_{i,1}$ has $a_k^{\mathsf{in}} = 0$ for every $k \neq \frac{n}{2}$, and $a_{n/2}^{\mathsf{in}} = 1$ (and thus for all i, computing $a_i^{\mathsf{out}}(\vec{x}_{i,1})$ requires $\Omega(n)$ queries). Suppose there exists a single i^*, which is chosen uniformly at random from $[1, \frac{n}{2} - 1]$, such that $\vec{x}_{i^*,1}$ has $b_{n/2}^{\mathsf{in}} = 1$ (and thus $a_{i^*}^{\mathsf{out}}(\vec{x}_{i^*,1}) = 1$). Similarly, suppose $\vec{x}_{n/2,1}$ has $b_{n/2}^{\mathsf{in}} = 1$ (and thus $a_{n/2}^{\mathsf{out}}(\vec{x}_{n/2,1}) = 1$). For all $i \neq i^*$, $\frac{n}{2}$ suppose $\vec{x}_{i,1}$ has $b_{n/2}^{\mathsf{in}} = 0$ (and thus for $i \neq i^*$, $\frac{n}{2}$, we have $a_i^{\mathsf{out}}(\vec{x}_{i,1}) = 0$).

Consider the following certificate algorithm: First the algorithm determines all of $b_i^{\mathsf{out}}(\vec{x}_{i,2})$. To do this, for every $\vec{x}_{i,2}$ the algorithm queries a_1^{in}, b_1^{in}. If for any $i \neq \frac{n}{2}$, either $a_1^{\mathsf{in}} \neq 1$ or $b_1^{\mathsf{in}} \neq 0$ then the input doesn't match the certificate (and in this case the algorithm can query everything, and we don't care about the number of queries). This requires $O(n)$ queries, since for every $b_i^{\mathsf{out}}(\vec{x}_{i,2})$ the algorithm makes two queries and there are $O(n)$ such b_i^{out}. Since for all $i < \frac{n}{2}$, $b_i^{\mathsf{out}}(\vec{x}_{i,2}) = 0$, if the certificate algorithm finds a single $i < \frac{n}{2}$ for which $a_i^{\mathsf{out}}(\vec{x}_{i,1}) = 1$ then the algorithm with a certificate knows that the output is 0. Thus the algorithm with a certificate can query all of the bits of $\vec{x}_{i^*,1}$ in $O(n)$ queries. If the certificate is truthful all of the $a_i^{\mathsf{in}} = 0$, except for $i = \frac{n}{2}$ for which $a_{n/2}^{\mathsf{in}} = 1$ and $b_{n/2}^{\mathsf{in}} = 1$, and thus $a_{i^*}^{\mathsf{out}}(\vec{x}_{i^*,1}) = 1$ (once again, if the queries do not match the certificate, then the certificate can query everything, and the number of queries is irrelevant). Thus the certificate algorithm outputs 0 in $O(n)$ queries.

On the other hand any randomized algorithm without a certificate will take $\Omega(n^2)$ queries. Suppose that for every i the randomized algorithm is given the value of each $b_i^{\mathsf{out}}(\vec{x}_{i,2})$, without any queries. The output of the function is '1' iff there exists an $i < \frac{n}{2}$ for which $a_i^{\mathsf{out}}(\vec{x}_{i,1}) = 1$. That is, the algorithm needs to output the OR function of $\frac{n}{2} - 1$ many $a_i^{\mathsf{out}}(\vec{x}_{i,1})$. In order to compute a single $a_i^{\mathsf{out}}(\vec{x}_{i,1})$ correctly with probability $\frac{2}{3}$ any randomized algorithm must make at least $k \in \Omega(n)$ queries due to Claim 3.18. That is $k = cn$ for some constant c. Consider any randomized algorithm that makes at most $\frac{cn^2}{10}$ queries. This algorithm, can determine at most $\frac{cn^2/10}{cn} = \frac{n}{10}$ many $a_i^{\mathsf{out}}(\vec{x}_{i,1})$ correctly with probability $\frac{2}{3}$ (since the input $\vec{x}_{i,1}$ and $\vec{x}_{i',1}$ are independent for each i and i'). Suppose the algorithm has determined $\frac{n}{10}$ many $a_i^{\mathsf{out}}(\vec{x}_{i,1})$ correctly with probability 1 (which is more than what any randomized algorithm can do). Since the single i^* that satisfies $a_{i^*}^{\mathsf{out}}(\vec{x}_{i^*,1}) = 1$ is distributed uniformly at random among all the possible $\frac{n}{2} - 1$ many i's, and since any randomized algorithm knows the value of at most $\frac{n}{10}$ many $a_i^{\mathsf{out}}(\vec{x}_{i,1})$, and since $\frac{n}{10} < \frac{2}{3}(\frac{n}{2} - 1)$, any randomized algorithm that makes at most $\frac{cn^2}{10}$ will succeed with probability less than $\frac{2}{3}$. ◁

◀

3.5 Testing Instance Optimality

In this section, we deal with the computational complexity of the problem of constructing and verifying instance optimal algorithms. The input is a function, i.e., the truthtable and the output is a decision tree.

We give an algorithm[10] U (for *universal*) that for a function f and an instance x, outputs an *optimal* randomized certificate tree for f on x. Thus, if one wants to test whether a given decision tree T is instance optimal for a problem f, then it is enough to compare its complexity on input x to the complexity of the output of U on f and input x for every $x \in \{0,1\}^n$. A different interpretation of U is that if one wants to prove that f is instance optimizable, then one has to give a decision tree that performs just as well as the decision tree generated by U for every input.

The running time of U for a problem f and input x is polynomial in the domain size of f and thus the total running time of testing whether a decision tree is instance optimal is a polynomial in the domain size of f

▶ **Theorem 3.20** (Theorem 1.4 rephrased). *There exists an algorithm U that gets as an input a function $f\colon \{0,1\}^n \to \{0,1\}$, and a certificate y for the input to the function, and generates in time $2^{O(n)}$ a correct randomized decision tree T_y for f such that $\mathrm{R}(\mathrm{T}_y, y) \in O(\mathrm{RC}(f,y))$.*

Proof. Let $f\colon \{0,1\}^n \to \{0,1\}$ be a Boolean function and $y \in \{0,1\}^n$ be a certificate for the input. For a subset $S \subseteq [n]$ let $y_S \in \{0,1\}^n$ be the incidence vector of S, i.e., a string whose i-th bit is:

$$(y_S)_i = \begin{cases} y_i & \text{if } i \notin S \\ \bar{y}_i & \text{if } i \in S \end{cases}$$

The algorithm U is given as Algorithm 1. On a high level, it uses the truth table of f to construct a linear program whose variables are all the possible inputs to f and the output is a set of n numbers $p_1, \ldots, p_n \in [0,1]$ that correspond to the probability that the certificate verification tree should query the ith input bit.

Each such bit should be queried independently and the result is a non-adaptive verifier. The algorithm takes exponential time (its input is a function f whose description may be exponential and it solves an exponentially large linear program) but the representation of the decision tree is small – just a set of probabilities $\{p_i\}_i$.

[10] Throughout the paper the term "algorithm" was used synonymously with the term "decision tree". In this section the term "algorithm" has a different meaning. Specifically, in this section the model the algorithm works in is RAM rather than the query model.

Algorithm 1 Randomized Decision Tree Generation Algorithm.

procedure $U(f, y)$
  Solve the linear program

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{n} p_i \\ \text{subject to} \quad & 0 \leq p_i \leq 1 \ , \forall 1 \leq i \leq n \\ & \sum_{i \in S} p_i \geq 1 \ , \forall S \subseteq [n] \text{ such that } f(y_S) \neq f(y) \end{aligned}$$

end procedure

The certificate verification algorithm is:

Repeat c times:
for $1 \leq i \leq n$ **do**
  query x at location i with probability p_i.
  if the result is equal to y_i **then** continue.
  else query x at all locations and output $f(x)$.
  end if
end for
output $f(y)$.

First note that there is always a solution for the linear program by setting all the p_i's to 1.

We begin by proving that Algorithm 1 satisfies correctness. Namely, that the resulting decision tree computes f correctly on every input (with sufficiently high probability). If $y = x$, then clearly the decision tree generated by Algorithm 1 outputs the correct value with probability 1 (since $f(x) = f(y)$). Otherwise, assume that $y \neq x$. Either the decision tree finds the differing point or not. In the former, it outputs $f(x)$ as needed. In the latter, there are two cases again: either $f(x) \neq f(y)$ or $f(x) = f(y)$. In the latter, we are done again by construction.

We are left with calculating the probability that the differing point is not recognized in case $f(x) \neq f(y)$. Let $S = \{i_1, \ldots, i_k\} \subseteq [n]$ be a subset of size $k \leq n$ of indices such that $x_i \neq y_i$ for $i \in S$. Algorithm 1 is wrong only if it does not query on any x_i such that $i \in S$. But since $f(y) \neq f(y_S)$ (since $y_S = x$) Algorithm 1 ensures that $\sum_{i \in S} p_i \geq 1$. Let $q_i = 1 - p_i$ (i.e., q_i is the probability that the decision tree does not query on x_i) and the probability of the algorithm being wrong is $\prod_{i \in S} q_i$. By the inequality of arithmetic and geometric means, we get that

$$\prod_{i \in S} q_i \leq \left(\frac{\sum_{i \in S} q_i}{k} \right)^k \leq \left(\frac{k-1}{k} \right)^k \leq e^{-1} < 0.4.$$

Hence, Algorithm 1 succeeds with probability at least 0.6. By standard amplification (running the algorithm repeatedly c times and outputting $f(y)$ only if there was never a disagreement), we get an algorithm whose success probability is at least 2/3.

To prove that Algorithm 1 is optimal in terms of the number of queries it makes, let T' be a randomized algorithm that computes f. Assume towards contradiction that $\mathrm{R}(\mathrm{T}', x) \in o(\mathrm{R}(\mathrm{T}, x))$, where T is the randomized decision tree generated by Algorithm 1. We will reach a contradiction by giving an input x' to T' for which it must error with probability

at least 1/3. By definition, $\mathrm{R}(\mathrm{T}, x) = \sum_{i=1}^{n} p_i$ and denote by p'_i the probability that T' queries x_i (on input x). Since $\mathrm{R}(\mathrm{T}', x) \in o(\mathrm{R}(\mathrm{T}, x))$, there exists some $S \subseteq [n]$ for which $\sum_{i \in S} p'_i \in o(1)$ and $f(x) \neq f(x')$ where $x' = x_S$. This is true since otherwise the p'_i's (times some constant) would be a solution to the linear program. Clearly, T' computes f correctly on x' with probability at most $o(1)$, contradicting the assumption. ◀

The conclusion is therefore that when given a function and a decision tree, in order to check whether the decision tree is instance optimal (with a given constant and some slackness) one may go over all inputs, for each one compute the certificate complexity as in Algorithm 1 and compare it to the complexity of the decision tree on that input. So only time proportional to polynomial in the truth table size is required.

▶ Remark 3.21. Aaronson [1, Lemma 5] showed that there is a non-adaptive verifier whose query complexity is similar to the best adaptive verifier. While Aaronson starts with the adaptive verifier and defines from it a non-adaptive one, we construct the non-adaptive verifier directly from the function.

We leave open the question of the complexity of testing whether a given function is instance optimizable.

▶ **Question 3.22.** *What is the complexity of testing whether a given function f is instance optimizable (within some c).*

3.6 Instance Optimality of Proximity Property Testing

Consider instance optimality in *proximity-to-property testing*: the task in proximity property testing, as defined by Goldreich, Goldwasser and Ron [26] (see [25] for a current survey), is to decide whether an object has some property or whether it is *far* from having it (according to some measure of distance). This is actually a function with "don't cares": if the object is close to satisfying the property but does not satisfy it, then any answer is acceptable; in this sense it is different than the other functions considered in the paper which are full domain.

There are several examples and characterizations of problems that can be tested within query complexity that only depends on the proximity parameter (see, for example, [26, 3, 27]). In this case, when the proximity parameter is constant, the number of queries is also constant which means that every such property is "trivially" instance optimizable: both the tester that has a certificate and the tester that does not have it can decide the property within a constant number of queries.

We observe that these properties, namely the ones that are testable within a constant number of queries, are *the only properties that are R-instance optimizable*: the tester that gets a certificate can always efficiently check (using $O(1/\delta)$ queries sampled at random) that its certificate is $\delta/2$-close to the object. If the certificate satisfies the property or if it $\delta/2$ close to satisfying the property, then it outputs "accept". Otherwise it outputs "reject". Here, we are using the fact that we are allowing two-sided errors.

4 Unlabeled Certificates

In this section we switch gears and study the notion of *unlabeled certificates* (see Section 2.2 for formal definitions). In the previous section the competing algorithm received as a certificate the entire input (this can be though of as an "untrusted hint" where the hint is the entire input), whereas in this section the certificate is a *permutation of the input* (the "untrusted hint" is a permutation of the input, where the permutation is chosen adversarially).

The unlabeled certificate makes sense especially for functions that are symmetric under some class of permutations. For instance, if the input is a graph property, the unlabeled certificate will be an isomorphic copy of the graph. That means that the algorithm should always be correct (even if the unlabeled certificate is not an isomorphic copy of graph), but the runtime of the algorithm is only measured when the unlabeled certificate is indeed isomorphic to the graph.

We study both the worst case complexity and the instance complexity of unlabeled certificates. For the worst case, roughly speaking, suppose we have a graph property, and suppose we are given an isomorphic copy of the graph as a certificate. It turns out that there exists a function for which such a certificate helps almost as much as an unlabeled certificate can possibly help. See Section 4.1 for more details.

In the case of instance complexity of unlabeled certificates, we say a function is unlabeled instance optimal if there exists an algorithm that performs as well as the unlabeled complexity (up to a multiplicative constant) for every input. We show that the group of permutations considered is important. Specifically, the majority property is randomized unlabeled instance optimal (where the unlabeled certificate is simply the hamming weight, since the function is symmetric under any permutation of the input), whereas the graph property of having more edges than non edges (now the group of allowed permutations are all isomorphisms of the graph) is far from being unlabeled instance optimal. See Section 4.2 for more details.

4.1 Unlabeled Certificates In the Worst Case

4.1.1 The Power of Unlabeled Certificates

We construct an example of a function (graph property) for which an algorithm that receives an isomorphic copy of the graph as a certificate outperforms any algorithm that has no certificate, thus showing a separation between algorithms receiving unlabeled certificates and algorithms with no certificate in the worst case. The separation is almost the largest possible.

▶ **Theorem 4.1** (Theorem 1.5 rephrased)**.** *There exists a graph property f for which* $\mathrm{AC}(f) \in O(n \log n)$ *while* $\mathrm{R}(f) \in \Omega(n^2)$.

That is, we show a function f for which $\mathrm{R}(f) \in \widetilde{\Omega}(\mathrm{AC}(f)^2)$. Since $\mathrm{D}(f) \geq \mathrm{R}(f)$, and $\mathrm{AC}(f) \geq \mathrm{RAC}(f)$, we also know that there exists a function where $D(f) \in \widetilde{\Omega}(\mathrm{AC}(f)^2)$ and $\mathrm{R}(f) \in \widetilde{\Omega}(\mathrm{RAC}(f)^2)$. Up to log factors, this example is tight in the deterministic setting since $\mathrm{D}(f) < \mathrm{C}(f)^2$ (see for example [34, Theorem 14.3]). In the randomized setting, the best known separation between $\mathrm{R}(f)$ and $\mathrm{RC}(f)$ is $\mathrm{R}(f) \in O(\mathrm{RC}(f)^2)$, and improving this separation is a major open research problem. In fact, if $\mathrm{D}(f) \in O(\mathsf{bs}(f)^2)$ then for all total functions f, $\mathrm{R}(f) \in O(\mathrm{RC}(f)^2)$, in which case our example is also tight in the randomized setting, up to log factors.

Our starting point in the construction is the property that every vertex has degree at least $\log n$. If one has a labeled certificate of the graph, then for each vertex it is easy to check $\log n$ neighbors and see that they indeed exist. But the problem with an unlabeled certificate is that there is no clear way to figure out who the vertices in the graph correspond to in the certificate and hence finding the $\log n$ neighbors might be lengthy and require $\Omega(n)$ queries per vertex. To overcome this we add some gadgets in order to uniquely define the vertices in a manner that allows to find their "identities" with relatively few queries.

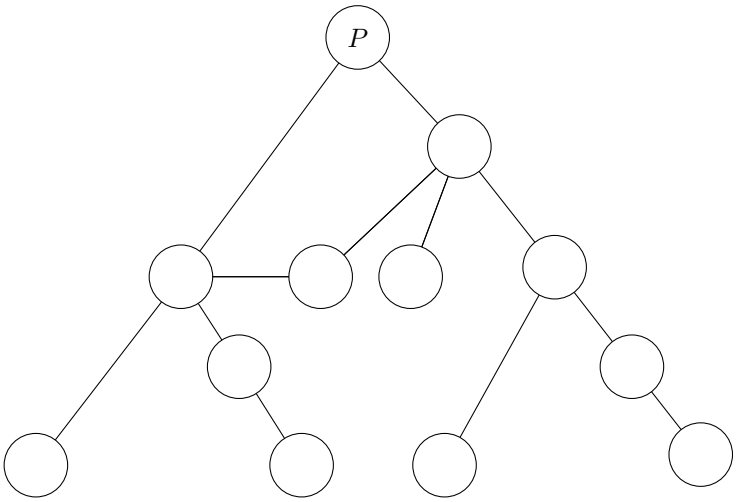


Figure 1 Example of binary tree illustrating Conditions 3 and 4.

▶ **Construction 4.2.** *The function outputs '1' iff all of the following conditions holds:*

1. *There are* exactly 2 *vertices of degree* 1, T_1 *and* T_2*, both of which have an edge to a special vertex, denote this vertex by* P*.*
2. *There is a vertex of degree* $n-3$ *that is adjacent to all vertices except* T_1 *and* T_2*. Call this vertex* B*.*
3. *Vertex* P *is the root of a "binary tree" of size* $10\log n$*. Every "right" child consists of an intermediate vertex. See Figure 1 where such a tree is shown. All vertices in the tree are also adjacent to* B*, this is not shown in the figure. Condition 4 is also shown in the figure.*
4. *There are* 2 *extra vertices, both adjacent to* P*'s right child, and one of which is also adjacent to* P*'s left child. This makes both of* P*'s immediate children also have degree* 5*. See Figure 1.*
5. *All the* $10\log n$ *leaves have degree different than* 5*. This makes* P *and its immediate children the only vertices of degree* 5*.*
6. *We call the set of remaining vertices* ($n - O(\log n)$ *in number) the "crowd" (denoted by* C*). The neighbor of every vertex in the crowd either also belongs to the crowd, or is a leaf of the tree described in Condition 3 or is the vertex* B *from Condition 2. Furthermore, all vertices in the crowd must have degree at least* $\log n$ *on the subgraph induced by* C *(that is, every vertex in* C *must have at least* $\log n$ *neighbors also in* C*).*[11]
7. *No two vertices in the crowd have the same neighborhood when restricted to the leaves of the tree. Note that this is possible since we have more than* $\log n$ *leaves.*

See Figure 2 for an illustration.

Proof. Note that if the function outputs 1, then T_1 and T_2 are the only vertices of degree 1. Thus, once we find T_1 or T_2 we can also find P in $O(n)$ queries by finding T_1's (or T_2's) only neighbor. If we know which vertex is B, then in $O(n)$ queries we can find T_1 and T_2, as they are the only vertices not adjacent to B. Once we find P we can check if Condition 1 is

[11] This condition can be replaced with any condition that can be solved in $O(n\log n)$ queries with a labeled certificate, but requires $\Omega(n^2)$ queries without a certificate.

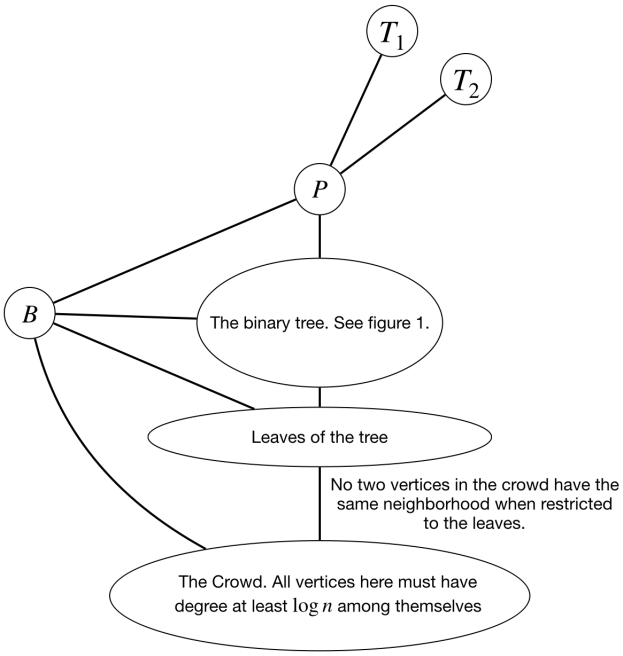


Figure 2 An illustration of Construction 4.2.

met by querying all of P's neighbors, and for each one making $O(n)$ queries to see if their only neighbor is P. If only 2 of P's 5 neighbors have this property then Condition 1 is met. Similarly once we find P, T_1 and T_2 we can check in $O(n)$ queries that Condition 2 is met by checking that one of P's neighbor has degree $n-3$ and is not adjacent to T_1 and T_2 as a neighbor. We can similarly check that Condition 4 is met in $O(n)$ queries given P. If we find a vertex, v, of degree 5 we can find P in $O(n)$ queries by checking if v has exactly two neighbors of degree 5, if it does then $v = P$, otherwise $v \neq P$ since P has 2 neighbors of degree 5, and these two neighbors don't have an edge between them.

The first phase of the algorithm is to find out either T_1, T_2 or B, and afterwards in $O(n)$ queries we also find P. We do this without using the unlabeled certificate similar to the way we check if a graph is a scorpion [9], see Appendix A. Start by picking an arbitrary vertex, v, and querying all of its neighbors. If $\deg(v) \in \{1, 5, n-3\}$ we are done, as v must be T, P (or one of P's immediate children) or B respectively. If $\deg(v) > n-3$ then output 0 since then we can't have two vertices of degree 1. So assume $\deg(v) \notin \{1, 5, n-3, n-2, n-1\}$. Let N_0 denote the set of all of v's neighbors, and let $N_1 = \overline{N_0} \setminus \{v\}$. Notice that if the function outputs 1 then $B \in N_0$ and $T_1, T_2 \in N_1$ since T_1 and T_2 are only adjacent to P, and $v \neq P$. Furthermore $P \in N_1$ because all of P's neighbors have degree $1, 5$ or $n-3$. Iteratively pick $x \in N_0$ and $y \in N_1$, and make a query to see if x is adjacent to y. If x and y are adjacent then y can't be T_1 or T_2 (since T_1 and T_2 only have an edge to P which is also in N_1), so remove y from N_0. If x and y are not adjacent then x can't be B unless y is T_1 or T_2, so remove x from N_0. Eventually N_0 will be empty, in which case y must be either T_1 or T_2. Thus in $O(n)$ queries we find P, and using $O(n)$ further queries we can check if Conditions 1, 2 and 4 are met.

Once we find P we proceed to find the leaves of the tree in $O(n \log n)$ queries. If at any point the structure does not match the tree (each vertex having one child of degree $2+1$ and one of degree $3+1$, where the $+1$ if for the edge with B) output 0 (Condition 3). Once

we find the leaves of the tree, for each leaf we make $n-1$ queries to check if each leaf has degree different than 5 (Condition 5) and to find the neighborhood of all the leaves. Having the neighborhood of each leaf allows us to check if Condition 7 is met, that no two vertices in the crowd have the same neighborhood when restricted to the leaves. If it is met then we can determine the isomorphism (for the vertices in C [12]), between the unlabeled certificate and the true (labeled) certificate. Note that the leaves of the tree are distinguishable since we know exactly which is a right child and which is a left child due to the intermediate right children, and this allows us to determine the isomorphism. With the (labeled) certificate we can check if Condition 6 is met in $O(n \log n)$.

Lower bound for randomized algorithms. It remains to show that the randomized complexity of this function is $\Omega(n^2)$. Suppose the algorithm is given (without any queries) all of the vertices of degree less than $\log n$, their neighbors, the vertex of degree $n-3$, as well as being told which vertices are the leaves of the tree. Suppose that all conditions except 6 are met. It remains to check if Condition 6 is met – that all vertices in C have at least $\log n$ neighbors also in C. Consider the following two input distributions:

1. The vertices in C are split into 3 sets, C_1, C_2 and C_3, where $|C_1| = \log(n) - 1$ and $|C_2| = |C_3| \in \Omega(n)$. The subgraphs induced by C_1 and C_3 are cliques, and every vertex in C_1 is connected to every vertex in C_2. For every vertex in C_2 we pick a vertex at random from C_3 and add an edge. Pick an edge uniformly at random from all edges that have one vertex in C_2 and one in C_3. Call the pair of endpoints of the edge the "marked pair".
2. Same as distribution 1, except the edge which is the marked pair is removed.

For the first distribution Condition 6 is met, since every vertex in C_2 has degree $\log n$ (on the subgraph induced by C) because it has $\log n - 1$ neighbors from C_1 and a single neighbor from C_3. In the second distribution Condition 6 is not met since there exists a single vertex in C_2 with degree $\log n - 1$.

Differentiating between the first distribution and the second distribution requires $\Omega(n^2)$ queries: Suppose that the algorithm A that attempts to distinguish between the two cases is given the partition of the vertices in C into C_1, C_2 and C_3 "for free", as well as being given for free that C_1 and C_3 are cliques. Furthermore, suppose that each query not only tells us if (u, v) is an edge, but also tells us if (u, v) is the marked pair. Note that the two distributions are identical, except that the second distribution does not contain the marked pair as an edge, and thus it is possible to tell apart the two distributions with any advantage (even if algorithm is allowed to err) if and only if the marked pair is queried.

Consider an arbitrary algorithm A. Let $N = |C_2| = |C_3| \in \Omega(n)$. At any point during the execution of A, call a vertex $u \in C_2$ "unmatched" if its adjacent edge has not been discovered yet, call it "thin" if thus far A has made at most $\frac{N}{2}$ queries that involve u, and call it "thick" if A has made at least $\frac{N}{2}$ queries that involve u. We can also think of the choice of the edges as well as the marked one as happening when the queries are made (principle of deferred decision): given an unmatched vertex u with q_u queries made so far, when a new slot is queried, it is matched with probability $1/(n - q_u - 1)$ and if it matched it is chosen as the marked one with probability proportional to the number of unmatched vertices.

[12] We do not have the full isomorphism, but we do know the mapping of the isomorphism for each of the vertices in the crowd – the vertices for which we want to check if they all have degree at least $\log n$ on the subgraph induced on C.

We do not charge the algorithm for queries made to thick vertices: once a vertex u becomes thick the next time A decides to query u it queries all the slots adjacent to it and gets the edge (or the marked pair); this modification does not result in a big change in the complexity, since it at most halves the total number of queries we charge A for making in any given execution. Thus, when considering an execution of A, we assume it can only choose among thin vertices.

We will first show that if A has made at most $q = \frac{1}{200}N^2$ queries so far, then the number of unmatched vertices is at least $N/2$ whp. Consider q independent trials (corresponding to the queries), each successful with probability at most $2/N$ (a vertex becoming matched). By the Chernoff Bound, the probability that q independent (and identical) trials, each successful with probability $\frac{2}{N}$ have more than $\frac{N}{2}$ successes is[13]

$$\Pr[\text{More than } N/2 \text{ vertices become matched}] \leq 2^{-N/2}.$$

Thus any algorithm that thus far has made less than $\frac{1}{200}N^2$ queries, has at least $\frac{N}{2}$ unmatched vertices with high probability.

We conclude that under the assumption that the number of unmatched vertices is at least $N/2$ the probability that A queries the marked pair in the next round is bounded by $4/N^2$: The probability that a given thin vertex u is part of the matched pair is $\frac{2}{N}$ (since there are at least $N/2$ unmatched vertices), and given that we query a vertex that is part of the marked pair, the probability that we query the marked pair is at most $2/N$ (since u is thin). So we get that the probability that A finds the marked pair in q rounds is bounded by $4q/N^2$ (union bound) plus the probability of not having $N/2$ unmatched vertices (which is negligible). So altogether, an algorithm operating with $q = \frac{1}{200}N^2$ queries has a limited probability of success and we get that the worst-case randomized complexity of f (Definition 2.3) is $\Omega(N^2) = \Omega(n^2)$. ◀

▶ Remark 4.3. The technique or gadget used in the above construction in order to make each vertex unique has additional applications. For instance, it can be used to turn an arbitrary function into a graph property while the complexity increases by an additive $O(n \cdot \log n)$ factor, resulting in the first known graph property with separation between the deterministic and randomized complexity. We explore this in future work.

4.2 Unlabeled Instance Optimality

4.2.1 Unlabeled Certificates of Majority

Our first example regarding unlabeled instance complexity is that while the majority function is not R-instance optimizable (Lemma 3.5), it is (almost) *unlabeled* R-instance optimizable. Namely, knowing the Hamming weight of the input does not help in significantly reducing the number of queries made to the input.[14]

Here, the optimality ratio is super-constant, that is, the unlabeled decision tree has an asymptotic advantage over the standard decision tree but it is only doubly-logarithmic in the input size (and this is tight).

[13] We use the following form of Chernoff: $\forall t \geq 2e\,\mathbb{E}[X]$, $\Pr[X \geq t] \leq 2^{-t}$. Indeed in this case $\mathbb{E}[X] = q\frac{2}{N} = \frac{N}{100}$. And $\frac{N}{2} > 2e\frac{N}{100}$.

[14] The majority is a symmetric function, and so in this case Γ contains all permutations of the n inputs, thus the full information is given by the hamming weight of the input.

▶ **Lemma 4.4** (Item 1 of Theorem 1.6). *The majority function is within $O(\log\log n)$ of being unlabeled R-instance optimizable.*

Proof. Consider an algorithm with an unlabeled certificate, which in the case of majority can be thought of as the number of 1's in the input. Suppose that the Hamming weight is $n/2+\varepsilon n$. Then the algorithm must make at least $\Omega(1/\varepsilon^2)$ queries to the input. This follows by fixing the distribution that with probability $1/2$ outputs a random input of Hamming weight $n/2+\varepsilon n$ and with probability $1/2$ outputs a random input of Hamming weight $n/2-\varepsilon n$. By the same fact we used in Lemma 3.5, $\Omega(1/\varepsilon^2)$ queries are required to distinguish the two cases.

We show that without a certificate, we can work with almost the same number of samples: $\Theta(1/\varepsilon^2\cdot\log\log(1/\varepsilon))$. The decision is obviously based on where the majority of the coordinates we read vote, but the issue is when to stop. This is a case of *sequential hypothesis testing* and the danger is that we are trying too many hypotheses and may fail in one of the trials, even though there is decent a probability of not failing in any particular one.

We use a recent result of Daskalakis and Kawase [15] who studied the following problem: Given sample access to an unknown distribution p over $\{0,1\}$ and an explicit distribution q over the same domain. How many samples are needed to reject the hypothesis "$p=q$" when $p\neq q$, while never rejecting when $p=q$. Daskalakis and Kawase showed that $\Theta(1/\mathsf{d}_{p,q}^2\cdot\log\log(1/\mathsf{d}_{p,q}))$ queries are needed and sufficient, where $\mathsf{d}_{p,q}$ is the total variation distance between p and q. We sketch the upper bound directly for our case next.

The algorithm samples a coordinate $i\leftarrow[n]$ uniformly at random and queries its input at i. Let t be the number of queries made so far and let t_0 be the number of 0's and t_1 the number of 1's (where $t_0+t_1=t$). Whenever t is a power of 2, the algorithm decides whether to stop or go on:

1. if $t\geq n$, read the whole input and output the majority value.
2. if $t_b > t/2+2\sqrt{t\cdot\log\log n}$, output the bit b.

We need to show two properties of the algorithm. First, that after enough iterations, it stops with the right answer. Second, that the probability that it stops too early, before making enough queries, is small. For the latter, since our test is done only when t is a power of two, the test is performed at most $\log n$ times. Thus, if we wish that no error will be made by stopping too early we need to reduce the probability of error per test to be at most $1/(3\log n)$. Assume that the input has $n/2+\varepsilon n$ 1's (the case with 0 majority is analogous). We start by showing that the probability that the algorithm halts and outputs the wrong answer is small. Denote by $T_b^{(t)}$ the random variable corresponding to the number of b's seen until query t, for $b\in\{0,1\}$. ($T_b^{(t)}$ is the sum of t independent random variables.) By a union bound, the probability that the algorithm makes a mistake it at most

$$\sum_{t=2^i}\Pr\left[T_0^{(t)}\geq\frac{t}{2}+2\sqrt{t\cdot\log\log n}\right],$$

where the sum is over all t's that are powers of 2. We bound each of these terms separately using Chernoff's bound.[15] Since the input contains $n/2+\varepsilon n$ 1's, it holds that $\mathbb{E}[T_0^{(t)}]=t/2-\varepsilon t$.

[15] We use the additive version of Chernoff's bound [4, Appendix A.1]. Let $X=\sum_{i=1}^n X_i$ be a sum of identically distributed independent random variables $X_1,\ldots,X_n\in\{0,1\}$. Let $\mu=\mathbb{E}[X]=\sum_{i=1}^n\mathbb{E}[X_i]$. It holds that for $a>0$, $\Pr[X>\mu+a]\leq\exp(-2a^2/n)$.

Hence, for every $t \in [n]$:

$$\begin{aligned}\Pr\left[T_0^{(t)} \geq \frac{t}{2} + 2\sqrt{t \cdot \log\log n}\right] &= \Pr\left[T_0^{(t)} \geq \left(\frac{t}{2} - \varepsilon t\right) + \left(\varepsilon\sqrt{t} + 2\sqrt{\log\log n}\right) \cdot \sqrt{t}\right] \\ &\leq \exp\left(-2\left(\varepsilon\sqrt{t} + 2\sqrt{\log\log n}\right)^2\right) \\ &\leq \exp\left(-8\log\log n\right) \leq \log^{-8} n.\end{aligned}$$

Thus, the overall probability that the algorithm makes a mistake is less than $\log n \cdot \log^{-8} n < 1/3$.

We show that with probability at least $2/3$ our algorithm halts (with the correct output) after at most $t' = \min\{2n, 25(\log\log n)/\varepsilon^2\}$ queries. This will finish the proof of the claim. Again, consider an input with $n/2 + \varepsilon n$ 1's. If ε is such that $t' \geq n$, then we are done. Otherwise, by the same Chernoff bound:

$$\begin{aligned}\Pr\left[T_1^{(t')} < \frac{t'}{2} + 2\sqrt{t' \cdot \log\log n}\right] &= \Pr\left[T_1^{(t')} \leq \left(\frac{t}{2} + \varepsilon t\right) - \left(\varepsilon\sqrt{t} - 2\sqrt{\log\log n}\right) \cdot \sqrt{t}\right] \\ &\leq \exp\left(-2\left(\varepsilon\sqrt{t} - 2\sqrt{\log\log n}\right)^2\right) < 1/3.\end{aligned}$$

◀

4.2.2 Unlabeled Certificates of Graph Properties

Our next result is that there exists a graph property which is *not* unlabeled R-instance optimizable. It is our "old friend" the majority property, but this time the group of permutations Γ considered is that of the vertices (rather than the edges), which demonstrates the significance of picking the most appropriate Γ.

▶ **Lemma 4.5** (Item 2 of Theorem 1.6). *There exists a graph property – majority of edges – which is not unlabeled R-instance optimizable. There is a distribution where any correct algorithm takes $\Omega(n^2)$ queries but an algorithm receiving an unlabeled certificate requires only $O(n \log n)$ queries.*

Proof. The graph property is that the graph $G = (V, E)$ with n vertices contains at least $\binom{n}{2}/2$ edges. The hard distribution for (certificateless) randomized decision trees will be a random graph that has either $\binom{n}{2}/2 + n$ edges or $\binom{n}{2}/2 - n$ edges.

On the one hand, a randomized algorithm that has no certificate cannot distinguish random graphs that have $\binom{n}{2}/2 + n$ edges from random graphs that have $\binom{n}{2}/2 - n$ edges unless it makes $\Omega(n^2)$ queries to the input and recovers a constant fraction of the graph. This is the fact we used in the proof of Lemma 3.5. The rest of the proof is devoted to showing that using an unlabeled certificate can significantly help.

We design a decision tree algorithm that has access to an unlabeled certificate and makes only $O(n \cdot \log n)$ queries to the input in expectation on the above distribution and decides the property correctly on all inputs with probability $2/3$. The algorithm works in two steps after which the algorithms recovers the *exact* graph (and not a permutation thereof) or figures out that it was given the wrong certificate. Then, we are back in the case of Lemma 3.5 where the (exact) certificate helps compared to the algorithm without the certificate.

Denote by $G = (V, E)$ the real graph we are querying and denote by $G' = (V, E')$ the permuted graph we have as a certificate. That is, $E' = \{(\pi(u), \pi(v)) \mid (u, v) \in E\})$ for some unknown permutation π. The first step of our algorithm is to sample a random set of $k = c \cdot \log n$ vertices $A = \{v_1, \ldots, v_k\}$ (for some large enough constant $c > 0$) and query all edges (v_i, v_j) for all $i, j \in [k]$. We call the set A the *anchor*. This costs at most $k^2 \in O(\log^2 n)$

queries. Denote by G_A the induced subgraph that the algorithm just queried. The algorithm then tests how many isomorphic subgraphs there are to G_A in G'. If there are no copies, we query all the edges in the graph, as the certificate is not a permutation of the input graph. If there is more than one copy, then we are unlucky (see below) and again we query all edges. If there is exactly one isomorphic subgraph, we have just identified the vertices in A and we can use them as an *anchor* to identify the other vertices.

The next step of the algorithm is to query the input on all possible neighbors for each vertex in the anchor A. This costs $n \cdot k \in O(n \cdot \log n)$ additional queries. For a vertex $v \in V \setminus A$, let $v(A) = (E(v, v_1), \ldots, E(v, v_k))$ be the incidence vector w.r.t. neighborhood in A. For each such $v \in V \setminus A$, look for a vertex in the certificate with the same "fingerprint", i.e., same list of vertices in A (again, if we find no match, we output $\perp$).

At this point, the algorithm completely recovers the permutation between the input and the certificate and thus we are almost done: as in Lemma 3.5, the algorithm checks using $O(n)$ random queries that the input graph corresponds to the certificate; if it does the algorithm answers accordingly (i.e., by the majority of the edges of the ceritificate) and if not the algorithm queries all the edges and answers according to the result. By the correctness of Lemma 3.5 we get that the algorithm is correct with high probability.

The fact that the algorithm is efficient, that is, if it is given a permutation of the true graph then the expected number of queries it makes is $O(n \log n)$, follows from the two claims below. They show that we will indeed recognize the vertices in A uniquely and then each other vertex in $V \setminus A$ (with very good probability). The overall complexity of the algorithm is $O(n \cdot \log n)$ queries with probability $1 - 1/n$ (the probability is over the distribution of the graphs and the random choices of the algorithm).

▷ Claim 4.6. With all but $1/n$ probability, there is only one induced subgraph of G that is isomorphic to G_A.

Proof. In Alon-Spencer [4, §10], Subsection titled "THE PROBABILISTIC LENS: Counting Subgraphs" the following is shown. Let $G = (V, E)$ be a $G(n, p)$ graph (i.e., a graph on n vertices with edge probability $p \in (0, 1)$) and let $S \subseteq V$ be a subset of k vertices. Let G_S be the induced subgraph of G to the vertices in S. Then, the probability that there is an isomorphic copy to G_S in G besides the one induced by S is at most

$$\sum_{g=1}^{t} \left(n^2 \cdot p^{1/3 - k/6} \right)^g .$$

Letting $p = 1/2 \pm n/\binom{n}{2}$ and $k = 100 \log n$, we get that the above probability is bounded by $1/n^2$.

The above holds for a $G(n, p)$ graph while our graph is sampled from a different distribution. Our distribution is such that the given graph is a random one conditioned on having $\binom{n}{2}/2 \pm n$ edges. But, a random $G(n, p)$ graph has inverse-polynomial probability of ending up with $p \cdot \binom{n}{2}$ edges so we can get the result for these graphs by conditioning. More generally, it is known that random graphs with m edges have similar properties to ones with edge probability $p = m/\binom{n}{2}$; see, for example, Frieze and Karonski [23, Lemma 1.2]. ◁

▷ Claim 4.7. With probability less than $1/n$, there exist distinct $v, v' \in V \setminus A$ such that $v(A) = v'(A)$.

Proof. For every two distinct $v, v' \in V \setminus A$ the probability that $v(A) = v'(A)$ is 2^{-k}. Applying a union bound over all possible such pairs, we get that there exist distinct $v, v' \in V \setminus A$ such that $v(A) = v'(A)$ with probability at most

$$|V \setminus A|^2 \cdot 2^{-k} \leq n^2/2^{c \cdot \log n} \leq 1/n^2,$$

where the last inequality follows for $c > 4$. ◁

This concludes the proof that majority is not instance optimal in this sense. ◀

Lastly, we show that even when considering graph properties (that is, the permutation group considered is over the vertices of the graph), there is a (monotone) graph property which is unlabeled R-instance optimizable:

▶ **Lemma 4.8** (Item 3 of Theorem 1.6)**.** *There exists a monotone graph property ("there exists at least one edge") which is unlabeled R-instance optimizable. But not labelling R-instance optimal.*

Proof. The graph property is that the graph $G = (V, E)$ contains at least one edge. Let $n = |V|$, and let m be the number of edges.The instance optimal randomized decision tree algorithm is the one that just samples random edges in the graph (without repetition) and queries on them until it either finds an edge or until it exhausts all the edges in the graph. If the input graph has $m \geq 1$ edges, the expected number of queries the algorithm makes is bounded by $\binom{n}{2}/m$.

In the unlabeled certificate case, we need to argue that for any $m \in \{0, \ldots, \binom{n}{2}\}$ and any graph G no randomized decision tree algorithm given G', an isomorphic copy of G, as an unlabeled certificate can be correct (on all graphs) with probability at least $2/3$ by querying in expectation $o\left(\binom{n}{2}/m\right)$.

Consider a distribution $\mathcal{D}_G$ on graphs such that with probability $1/2$ the output is a random isomorphic copy of the graph G and with probability $1/2$ the output is the empty graph (no edges at all). Consider the first k queries of any decision tree T. We will argue that for $k = \binom{n}{2}/(8m)$, with high probability T will not query any edge (when the graph it queries is a random isomorphic copy of G), and thus have no way of telling if the graph is indeed an isomorphic copy of G' or the empty graph. Note that as long as T doesn't query any edges, T is non-adaptive. Since T makes k queries, the probability that it queries at least one edge is the same as the probability that a k-edge graph H and a random permutation of a m-edge graph G intersect. We argue that this is roughly on the order of $k \cdot m/n^2$. More precisely, by a union bound and assuming the first $i-1$ rounds returned no edges, if T makes k queries, then with probability at most:

$$\sum_{i=1}^{k} \frac{m}{\binom{n}{2} - i + 1} \leq \frac{2mk}{n^2 - 2n - 2k} \leq 1/4$$

it will query an edge. Therefore, with probability at least $2/3$ it will not query any edge and thus have no way of knowing whether the graph has an edge or not. This means that $\Omega\left(\binom{n}{2}/m\right)$ queries must be made in expectation, same as the naive algorithm that does not get a certificate nor does it know m and therefore the property of having at least one edge is unlabeled R-instance optimizable.

The function is not labelling R-instance optimal, since if there exists one edge the algorithm with a (labeled) certificate requires only $O(1)$ queries. ◀

4.2.3 Labelling Instance Optimal

All instance optimal functions are also labelling instance optimal since for every x $\mathrm{C}(f,x) \leq \mathrm{AC}(f,\Gamma,x) \leq \mathrm{D}(f,x)$ and $\mathrm{RC}(f,x) \leq \mathrm{RAC}(f,\Gamma,x) \leq \mathrm{R}(f,x)$. We show that, up to log factors, the converse is not true; i.e, there exists a function that is within $O(\log(n))$ of being labelling D-instance optimal (resp. labelling R-instance optimal) but is not D-instance optimal (resp. R-instance optimal).

▶ **Lemma 4.9** (Item 4 of Theorem 1.6). *The function defined in Construction 4.2 is within $O(\log(n))$ of being labelling D-instance optimal, and labelling R-instance optimal, but is neither D-instance optimal nor R-instance optimal.*

Proof. We know the function isn't D nor R instance optimal since in the worst case the algorithm that has access to an unlabeled certificate does significantly better than the algorithm that has no certificate due to Theorem 4.1.

We will show that for every input $\mathrm{RC}(f,x) \in \Omega(n)$. Since for all inputs $\mathrm{AC}(f,x) \in \widetilde{O}(n)$ both D and R instance optimality follows. Notice that all the conditions except for Condition 3 make a requirement about the degree of a given vertex (or multiple vertices). $\Omega(n)$ queries are necessary to determine the degree of a given vertex with high probability. Condition 3 claims that we have a binary tree of size $O(\log(n))$ and thus this condition also requires $\Omega(n)$ queries with a certificate. ◀

4.2.4 Unlabeled Instance Optimality of Proximity Property Testing

It turns out that *every* proximity property, given an (unlabeled) certificate, can be done with at most $\tilde{O}(\sqrt{n})$ queries (with two-sided errors). This follows from a result of Fischer and Matsliah [22] who studied the query complexity of graph isomorphism in the proximity property testing model. One of their results determines the query complexity of this problem where one of the graphs is "known" to the algorithm and while allowing two-sided errors (this is in the dense graph model). They showed that, in this setting, $\tilde{O}(\sqrt{n})$ queries suffice and $\Omega(\sqrt{n})$ queries are necessary in general. Thus, in the unlabeled model, we can use this algorithm to first test proximity between the object and the given unlabeled certificate (using $\tilde{O}(\sqrt{n})$ queries). Then, if they are close, answer according to whether the certificate possess the property. If they are far, query the whole object and just compute the output precisely.

So, for a proximity property to be instance optimizable, it is necessary to prove that $O(\sqrt{n})$ queries are sufficient in the worst case.

5 Open Questions and Future Research

This work raises several open problems and research directions. Conjecture 3.15 is about the lack of monotone graph properties which are R-instance optimal.

▶ **Conjecture 5.1** (Conjecture 3.15, repeated). *Every non-trivial monotone graph property is not R-instance optimizable.*

In section Section 3.5 we showed that given a truth-table representing a function, we can test if a given algorithm is instance optimal in time polynomial in the size of the truth-table. It remains open if we can test if a given function is instance optimizable in time polynomial in the size of the truth table. Another question is, given a decision tree (instead of a truth table), can we test whether the function it evaluates is instance optimizable.

▶ **Question 5.2** (Question 3.22, repeated). *What is the complexity of testing whether a given function f is instance optimizable (within some c).*

As we observed in Section 3.6, there is a clean characterization of (randomized) instance optimality in the context of proximity testing. Also, we know that unlabeled certificates help (with properties which require many queries to test), but we do not have a full characterization.

▶ **Question 5.3.** *Are there unlabeled instance optimizable properties in the proximity testing model, with worst case complexity $\omega(1)$?*

Another question is whether analogues of the Aanderaa-Rosenberg conjecture holds for unlabeled certificate decision tree complexity. The randomized variant of this question for monotone properties is also open.

▶ **Question 5.4.** *Can the result of Rivest and Vuillemin [44] be extended to algorithms with an unlabeled certificate? Namely, do deterministic decision tree algorithms require querying a constant fraction of the edges for any non-trivial monotone graph property, even given a permutation of the given graph?*

In this work we considered two types of hints regarding the input, i.e. side information where the competing algorithm is measured only when it is correct: the full input and a permutation of the input. A natural question is what other types of hints are useful to study, e.g. partial inputs (that do not contain a certificate), such as the degree sequence.

References

1. Scott Aaronson. Quantum certificate complexity. *J. Comput. Syst. Sci.*, 74(3):313–322, 2008. doi:10.1016/j.jcss.2007.06.020.
2. Peyman Afshani, Jérémy Barbay, and Timothy M. Chan. Instance-Optimal Geometric Algorithms. *J. ACM*, 64(1):3:1–3:38, 2017. doi:10.1145/3046673.
3. Noga Alon, Eldar Fischer, Ilan Newman, and Asaf Shapira. A Combinatorial Characterization of the Testable Graph Properties: It's All About Regularity. *SIAM J. Comput.*, 39(1):143–167, 2009.
4. Noga Alon and Joel Spencer. *The Probabilistic Method*. John Wiley, third edition, 2008.
5. Andris Ambainis, Kaspars Balodis, Aleksandrs Belovs, Troy Lee, Miklos Santha, and Juris Smotrovs. Separations in Query Complexity Based on Pointer Functions. *J. ACM*, 64(5):32:1–32:24, 2017.
6. Andris Ambainis and Xiaoming Sun. New separation between *s(f)* and *bs(f)*. *Electronic Colloquium on Computational Complexity (ECCC)*, 18:116, 2011.
7. Ilya Baran and Erik D. Demaine. Optimal adaptive algorithms for finding the nearest and farthest point on a parametric black-box curve. In *Proceedings of the 20th ACM Symposium on Computational Geometry, SOCG*, pages 220–229. ACM, 2004.
8. Robert Beals, Harry Buhrman, Richard Cleve, Michele Mosca, and Ronald de Wolf. Quantum lower bounds by polynomials. *J. ACM*, 48(4):778–797, 2001.
9. R. Best, P. van Emde Boas, and H.W. Lenstra. *A Sharpened Version of the Aanderaa-Rosenberg Conjecture*. Mathematisch Centrum Amsterdam. Afdeling Zuivere Wiskunde: ZW. Stichting Mathematisch Centrum, 1974. URL: http://books.google.co.il/books?id=9KLcHAAACAAJ.
10. Manuel Blum and Russell Impagliazzo. Generic Oracles and Oracle Classes (Extended Abstract). In *28th Annual Symposium on Foundations of Computer Science, FOCS*, pages 118–126. IEEE Computer Society, 1987.
11. Allan Borodin and Ran El-Yaniv. *Online computation and competitive analysis*. Cambridge University Press, 1998.

12 Harry Buhrman and Ronald de Wolf. Complexity measures and decision tree complexity: a survey. *Theor. Comput. Sci.*, 288(1):21–43, 2002. doi:10.1016/S0304-3975(01)00144-X.

13 Amit Chakrabarti and Subhash Khot. Improved lower bounds on the randomized complexity of graph properties. *Random Struct. Algorithms*, 30(3):427–440, 2007.

14 Stephen A. Cook, Cynthia Dwork, and Rüdiger Reischuk. Upper and Lower Time Bounds for Parallel Random Access Machines without Simultaneous Writes. *SIAM J. Comput.*, 15(1):87–97, 1986.

15 Constantinos Daskalakis and Yasushi Kawase. Optimal Stopping Rules for Sequential Hypothesis Testing. In *25th Annual European Symposium on Algorithms, ESA*, pages 32:1–32:14, 2017.

16 Erik D. Demaine, Dion Harmon, John Iacono, Daniel M. Kane, and Mihai Patrascu. The geometry of binary search trees. In *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 496–505, 2009.

17 Erik D. Demaine, Alejandro López-Ortiz, and J. Ian Munro. Adaptive set intersections, unions, and differences. In *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 743–752. ACM/SIAM, 2000.

18 Cynthia Dwork and Yoram Moses. Knowledge and Common Knowledge in a Byzantine Environment: Crash Failures. *Inf. Comput.*, 88(2):156–186, 1990. doi:10.1016/0890-5401(90)90014-9.

19 Ronald Fagin, Amnon Lotem, and Moni Naor. Optimal aggregation algorithms for middleware. *J. Comput. Syst. Sci.*, 66(4):614–656, 2003. doi:10.1016/S0022-0000(03)00026-6.

20 Edward Farhi, Jeffrey Goldstone, Sam Gutmann, and Michael Sipser. Limit on the Speed of Quantum Computation in Determining Parity. *Phys. Rev. Lett.*, 81:5442–5444, December 1998.

21 Amos Fiat and Gerhard J. Woeginger, editors. *Online Algorithms, The State of the Art (the book grow out of a Dagstuhl Seminar, June 1996)*, volume 1442 of *Lecture Notes in Computer Science*. Springer, 1998. doi:10.1007/BFb0029561.

22 Eldar Fischer and Arie Matsliah. Testing Graph Isomorphism. *SIAM J. Comput.*, 38(1):207–225, 2008.

23 Alan M. Frieze and Michal Karonski. *Introduction to Random Graphs*. Cambridge University Press, 2015. doi:10.1017/CBO9781316339831.

24 Justin Gilmer, Michael Saks, and Srikanth Srinivasan. Composition limits and separating examples for some boolean function complexity measures. *Combinatorica*, 36(3):265–311, June 2016. doi:10.1007/s00493-014-3189-x.

25 Oded Goldreich. *Introduction to Property Testing*. Cambridge University Press, 2017.

26 Oded Goldreich, Shafi Goldwasser, and Dana Ron. Property Testing and its Connection to Learning and Approximation. *J. ACM*, 45(4):653–750, 1998.

27 Oded Goldreich and Dana Ron. Algorithmic Aspects of Property Testing in the Dense Graphs Model. In *Property Testing - Current Research and Surveys*, pages 295–305. Springer, 2010.

28 Mika Göös, Toniann Pitassi, and Thomas Watson. Query-to-Communication Lifting for BPP. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 132–143. IEEE Computer Society, 2017.

29 Lov K. Grover. A Fast Quantum Mechanical Algorithm for Database Search. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, pages 212–219. ACM, 1996.

30 Péter Hajnal. An $\Omega(n^{4/3})$ lower bound on the randomized complexity of graph properties. *Combinatorica*, 11(2):131–143, 1991.

31 Joseph Y. Halpern, Yoram Moses, and Orli Waarts. A Characterization of Eventual Byzantine Agreement. In *Proceedings of the Ninth Annual ACM Symposium on Principles of Distributed Computing*, pages 333–346. ACM, 1990.

32 Hao Huang. Induced subgraphs of hypercubes and a proof of the Sensitivity Conjecture. *CoRR*, abs/1907.00847, 2019. arXiv:1907.00847.

33 Rahul Jain and Shengyu Zhang. The Influence Lower Bound Via Query Elimination. *Theory of Computing*, 7(1):147–153, 2011.

34 Stasys Jukna. *Boolean Function Complexity: Advances and Frontiers*. Springer Berlin Heidelberg, 2012.

35 Jeff Kahn, Michael E. Saks, and Dean Sturtevant. A topological approach to evasiveness. *Combinatorica*, 4(4):297–306, 1984.

36 Valerie King. An $\Omega(n^{5/4})$ lower bound on the randomized complexity of graph properties. *Combinatorica*, 11(1):23–32, 1991.

37 Silvio Micali and Rafael Pass. Local zero knowledge. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing, STOC*, pages 306–315. ACM, 2006. Revision `http://www.cs.cornell.edu/~rafael/papers/preciseZK.pdf`.

38 Yoram Moses and Mark R. Tuttle. Programming Simultaneous Actions Using Common Knowledge. *Algorithmica*, 3:121–169, 1988.

39 Rajeev Motwani and Prabhakar Raghavan. Randomized Algorithms. In Allen B. Tucker, editor, *The Computer Science and Engineering Handbook*, pages 141–161. CRC Press, 1997.

40 Noam Nisan. CREW prams and decision trees. *SIAM J. Comput.*, 20(6):999–1007, 1991.

41 Ryan O'Donnell. *Analysis of Boolean Functions*. Cambridge University Press, 2014. URL: `http://www.cambridge.org/de/academic/subjects/computer-science/algorithmics-complexity-computer-algebra-and-computational-g/analysis-boolean-functions`.

42 Pekka Orponen, Ker-I Ko, Uwe Schöning, and Osamu Watanabe. Instance Complexity. *J. ACM*, 41(1):96–121, 1994. `doi:10.1145/174644.174648`.

43 Ran Raz and Pierre McKenzie. Separation of the Monotone NC Hierarchy. *Combinatorica*, 19(3):403–435, 1999.

44 Ronald L. Rivest and Jean Vuillemin. A Generalization and Proof of the Aanderaa-Rosenberg Conjecture. In *Proceedings of the 7th Annual ACM Symposium on Theory of Computing, STOC*, pages 6–11. ACM, 1975.

45 David Rubinstein. Sensitivity vs. Block Sensitivity of Boolean Functions. *Combinatorica*, 15(2):297–299, 1995.

46 Michael Saks and Avi Wigderson. Probabilistic Boolean Decision Trees and the Complexity of Evaluating Game Trees. In *27th Annual Symposium on Foundations of Computer Science*, SFCS '86, pages 29–38. IEEE Computer Society, 1986. `doi:10.1109/SFCS.1986.44`.

47 Daniel Dominic Sleator and Robert Endre Tarjan. Self-Adjusting Binary Search Trees. *J. ACM*, 32(3):652–686, 1985. `doi:10.1145/3828.3835`.

48 Avishay Tal. Properties and applications of boolean function composition. In *Innovations in Theoretical Computer Science, ITCS*, pages 441–454. ACM, 2013.

49 Gábor Tardos. Query complexity, or why is it difficult to seperate $NP^A \cap coNP^A$ from P^A by random oracles A? *Combinatorica*, 9(4):385–392, 1989.

50 Gregory Valiant and Paul Valiant. Instance optimal learning of discrete distributions. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC*, pages 142–155. ACM, 2016.

51 Gregory Valiant and Paul Valiant. An Automatic Inequality Prover and Instance Optimal Identity Testing. *SIAM J. Comput.*, 46(1):429–455, 2017.

52 Andrew Chi-Chih Yao. Probabilistic Computations: Toward a Unified Measure of Complexity (Extended Abstract). In *18th Annual Symposium on Foundations of Computer Science, FOCS*, pages 222–227. IEEE Computer Society, 1977.

53 Andrew Chi-Chih Yao. Lower Bounds to Randomized Algorithms for Graph Properties. *J. Comput. Syst. Sci.*, 42(3):267–287, 1991.

A Efficiently Testing Scorpion Graphs

▷ Claim A.1 (Restatement of Claim 3.12, [9]). Testing whether a given n vertex graph is a scorpion graph takes at most $O(n)$ queries.

Proof. Let $G = (V, E)$ be an n vertex graph. We give an algorithm that does at most $O(n)$ queries to the graph G. Denote by $d(v)$ the degree of a vertex $v \in V$. Observe that once we find one of the special vertices (i.e., the body, the tail or the sting), we can locate all other special vertices with at most $3n$ queries and then check that the graph is a scorpion graph. For example, once we find a vertex b with $d(b) = n - 2$, then the vertex must be the body (if the graph is a scorpion). Going over all the neighbors of b we locate the only vertex $s \in V$ that is not a neighbor of b. This must be the sting vertex (again, if the graph is a scorpion). Then, going over all the edges adjacent to s, we locate the tail vertex t and verify that $d(t) = 2$. In total, we made $3n$ queries to the edges in G. Thus, the goal now is to locate one of the special vertices.

We begin with an arbitrary vertex $v \in V$. If $d(v) \in \{0, n-1\}$ then G is not a scorpion graph. Otherwise, if $d(v) \in \{1, 2\}$ then either v is one of the special vertices or a neighbor of v is one of the special vertices. If $d(v) = n - 2$ then v must be the body vertex. In any such case we can identify all the special vertices and checking if the graph is a scorpion graph using at most $4n$ queries to G.

Assume now that $3 \leq d(v) \leq n - 3$. Denote by N_0 the set of neighbors of v. Let $N_1 = \bar{N_0} \setminus \{v\}$. The body of G (if G is a scorpion) must be in N_0 and the sting and the tail must be in N_1. At this point we iteratively choose $x \in N_0$ and $y \in N_1$ such that if $(x, y) \in E$ then $N_1 = N_1 \setminus \{y\}$ (as y can not be the sting) and choose a new $y \in N_1$, and otherwise (i.e., if $(x, y) \notin E$), $N_0 = N_0 \setminus \{x\}$ (as x can not be the body unless y is the sting) and choose a new $x \in N_0$. This process terminates after at most n queries since after every query a vertex is deleted. Moreover, if G is a scorpion then at the end of the iterations $N_0 = \emptyset$ and y is the sting. To see this, notice that the body can not be deleted from N_0 by any vertex in N_1 that is not the sting, and once the sting is encountered all the vertices in N_0 will be deleted. ◁

B Instance Optimality in Other Settings

The term "Instance optimality" was coined by Fagin, Lotem and Naor [19] in the context of finding items with the top k aggregate scores in a database of sorted lists.[16] It has appeared in the theoretical computer science literature in several other contexts and forms. Here are a number of examples:

- **Competitive online analysis.** The competitive ratio of an algorithm A is its worst case performance relative to the best offline (see [11, 21]). This is a slightly different form of instance optimality since the comparison here is to an algorithm that is not necessarily from the same class.
- **Approximation algorithms.** Comparing the size (or value) that the best algorithm can find to the one the approximating algorithm finds.
- **Self-adjusting data structures.** Is it possible to construct an optimal binary search tree, i.e., with smallest possible search time, for an access sequence given online? Sleator and Tarjan [47] conjectured that Splay Trees are instance optimal in this sense and resolving it has been a central issue in the area. See also Demaine et al. [16].

[16] We note that the term "Instance Complexity" has been used by Orponen et al. [42] in a different meaning, one related to Kolmogorov Complexity.

- **Database operations.** Demaine et al. [17] studied the problem of finding intersections, unions, or differences of a collection of sorted sets. While the worst-case complexity of these problems is straightforward, they consider algorithms whose complexity depends on the particular instance.
- **Computational geometry.** Baran and Demaine [7] gave an instance optimal algorithm for the nearest-point-on-curve problem (where one needs to find a point on a curve that is nearest to a given point). Instance optimal algorithms for convex hull and set maxima were given by Afshani et al. [2]. Their notion is particularly related to the unlabeled model we consider in Section 4, where the algorithm is competing against the best one for the given set of points in some order.
- **Learning Theory.** A fundamental question in learning theory is to produce an accurate as possible approximation of an unknown distribution (over a discrete support) given independent draws from it. Valiant and Valiant [50] gave an algorithm that outputs an approximation whose expected distance from the real distribution is equal to the minimum possible expected error that could be obtained by any algorithm that *knows* the true yet unlabeled distribution and simply needs to assign labels. Another work of Valiant and Valiant [51] studied the identity testing problem: Given the explicit description of a distribution, decide whether a set of samples was drawn from it or from a distribution with some distance from it. They gave an algorithm in which the number of queries depends on the given distribution.
- **Distributed computing.** A series of works studied the possibility of instance optimal algorithms for eventual Byzantine agreement (a.k.a. Consensus) and simultaneous Byzantine agreement [18, 38, 31]. In these problems the inputs are the votes of the players and the failure pattern. They provided several positive and negative results for various notions of instance optimality appropriate for the setting.
- **Cryptography.** The notion of *precise zero knowledge* bounds the knowledge gained by a player in terms of its actual computation rather than standard zero knowledge that bounds the knowledge of a player in terms of his *potential* computational power (see Micali and Pass [37]).

In this work, we concentrated on cases where the domain is full and there is no promise regarding the input (as opposed to, say the work of Fagin et al. [19], where the assumption is that each column is sorted).

On the Impossibility of Probabilistic Proofs in Relativized Worlds

Alessandro Chiesa
UC Berkeley, CA, USA
alexch@berkeley.edu

Siqi Liu
UC Berkeley, CA, USA
sliu18@berkeley.edu

Abstract

We initiate the systematic study of probabilistic proofs in relativized worlds, where the goal is to understand, for a given oracle, the possibility of "non-trivial" proof systems for deterministic or nondeterministic computations that make queries to the oracle.

This question is intimately related to a recent line of work that seeks to improve the efficiency of probabilistic proofs for computations that use functionalities such as cryptographic hash functions and digital signatures, by instantiating them via constructions that are "friendly" to known constructions of probabilistic proofs. Informally, negative results about probabilistic proofs in relativized worlds provide evidence that this line of work is inherent and, conversely, positive results provide a way to bypass it.

We prove several impossibility results for probabilistic proofs relative to natural oracles. Our results provide strong evidence that tailoring certain natural functionalities to known probabilistic proofs is inherent.

2012 ACM Subject Classification Theory of computation → Computational complexity and cryptography

Keywords and phrases probabilistically checkable proofs, relativization

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.57

Related Version A full version of the paper is available at `https://ia.cr/2019/1430`.

Funding This research was supported in part by: the Berkeley Haas Blockchain Initiative, and donations from the Ethereum Foundation and the Interchain Foundation.

1 Introduction

The study of relativized complexity classes originally aspired to shed light on the structural relationships between *un*relativized complexity classes. However, it was soon realized that many interesting complexity classes have contradictory relativization results. For instance, Baker et al. [9] showed that there exist oracles A and B such that $\mathrm{P}^A = \mathrm{NP}^A$ and $\mathrm{P}^B \neq \mathrm{NP}^B$.

Subsequent works sought to circumvent this difficulty by considering relativized worlds where the oracle is sampled from a "natural" distribution, and thereby avoid specially-crafted oracles that can force an equality/inequality on the complexity classes being compared. For instance, Bennett and Gill [20] proved that, with probability 1 over a random oracle R, it holds that $\mathrm{P}^R \neq \mathrm{NP}^R \neq \text{co-}\mathrm{NP}^R$ and $\mathrm{P}^R = \mathrm{BPP}^R$. Since these relativization results agreed with what people believed to be true in the unrelativized case, Bennett and Gill proposed the *Random Oracle Hypothesis*, which states that structural relationships between complexity classes that hold with probability 1 over a random oracle also hold in the unrelativized case. However, this hypothesis was later disproved by Chang et al. [22], who showed that, with probability 1 over a random oracle R, $\mathrm{IP}^R \neq \mathrm{PSPACE}^R$. (We know that, without oracles, IP = PSPACE [34, 41].)

These works indicate that, in general, relativization results are not helpful for understanding the relationships between unrelativized complexity classes. At best they provide us with relativization barriers, which nowadays are not considered so strong since we know of non-relativizing techniques.

1.1 New motivation: the efficiency of probabilistic proofs

We revisit relativization with a new motivation: the efficiency of probabilistic proofs. Superficially, relativization and probabilistic proofs seem unrelated. Yet they are deeply connected, as we explain.

Probabilistic proofs such as interactive proofs [28] and probabilistically checkable proofs [8] have played important roles in the study of hardness of approximation since the seminal work of [25]. In recent years, they became the subject of intense study due to their application to constructing highly-efficient cryptographic proofs (such as succinct arguments), and a major research goal today is to improve the efficiency of probabilistic proofs. We now illustrate, via an example, how relativization results tell us important facts about the efficiency of probabilistic proofs.

▶ **Example 1.** Let $\mathcal{H} = \{H_s \colon \{0,1\}^{|s|} \to \{0,1\}^{|s|}\}_{s\in\{0,1\}^*}$ be a family of "hash functions" (the precise security property in this discussion is unimportant), and consider the following NP language:

$$L_s = \{(n, y) \in \mathbb{N} \times \{0,1\}^{|s|} \mid \exists\, x \in \{0,1\}^{|s|} \text{ s.t. } H_s^n(x) = y\} \ .$$

The efficiency measures (proof length, randomness complexity, query complexity, and others) of a PCP for the language L_s typically depend on the size of an arithmetic circuit that iteratively applies H_s, for n times, to a candidate witness x and checks if the result is y. The size of such a circuit is $\Omega(n\,|H_s|)$, i.e., it depends on the size of a circuit for expressing the computation of H_s. Since there are many NP languages of interest to practitioners that involve cryptographic computations such as hash functions, researchers have been designing specialized families of hash functions that can be represented via small arithmetic circuits [6, 2, 29, 4, 3, 30].

We ask: *is optimizing the arithmetic circuit complexity of hash functions necessary?*

We now explain why the answer to this question is connected to relativization statements about probabilistic proofs. Informally, suppose that for any family of hash functions $\mathcal{H}$ it holds that $\mathrm{NP}^{\mathcal{H}} \subseteq \mathrm{PCP}^{\mathcal{H}}$. In other words, every language that can be decided by a nondeterministic polynomial-time machine that makes oracle calls to a hash function has a PCP verifier that may make oracle calls to the same hash function. Now the "oracle language" $\mathcal{L} = \{L_s\}_{s\in\{0,1\}^*}$, which is in the relativized complexity class $\mathrm{NP}^{\mathcal{H}}$, can be decided via a computation that involves n calls to the hash function but does not depend on the complexity of the hash function itself (as this computation happens inside the oracle). Hence, since we assumed that $\mathrm{NP}^{\mathcal{H}} \subseteq \mathrm{PCP}^{\mathcal{H}}$, we can obtain a probabilistic proof for $\mathcal{L}$ whose efficiency does *not* depend on the complexity of the hash function (which is wonderful).

In sum, probabilistic proofs that "relativize with respect to hash functions" obviate the need to design hash functions with small complexity and, conversely, negative results about such relativizations provide strong evidence that efforts to design "PCP-friendly" hash functions are inherent.

The above example illustrates a general connection. On the one hand, constructing probabilisitic proofs in relativized worlds could provide drastic efficiency improvements to probabilistic proofs. On the other hand, ruling out probabilistic proofs in relativized worlds

would provide a complexity-theoretic justification for why practitioners may be "stuck" with the task of designing "PCP-friendly" realizations of various functionalities (hash functions, signatures, encryption, and so on).

1.2 Our question: are there PCPs for computations in relativized worlds?

We initiate the systematic study of probabilistic proofs for relativized computations.

In this work an *oracle* is a collection $\mathcal{A} = \{\mathcal{A}_n\}_{n\in\mathbb{N}}$ where each $\mathcal{A}_n$ is a distribution over functions on n-bit inputs. A *sample* from $\mathcal{A}$ is a function $A\colon \{0,1\}^* \to \{0,1\}^*$ obtained by sampling a function f_n from each $\mathcal{A}_n$ and then setting A to equal f_n for n-bit inputs. (See Section 2.1 for definitions.)

We wish to understand for what oracles $\mathcal{A}$ there are probabilistically checkable proofs (PCPs) in a relativized world where all machines have oracle access to a sample from $\mathcal{A}$. Below we make this question more precise, distinguishing between the case of PCPs for relativized *nondeterministic* computations ("do PCPs provide any savings in witness length?") and the case of PCPs for relativized *deterministic* computations ("do PCPs provide any savings in computation length?").

The complexity classes that we study are the natural relativized extensions of DTIME, NTIME, and PCP. Note that complexity classes relative to an oracle $\mathcal{A}$ are sets of *oracle languages* (see Definition 11) rather than sets of languages, because the sample from $\mathcal{A}$ affects whether a particular instance is in the language or not. The informal definitions below are made precise in Section 2.4.

$\mathrm{DTIME}(t(n))^{\mathcal{A}}$	oracle languages that are decidable by a *deterministic* machine that runs in time $O(t(n))$ and has oracle access to a sample from $\mathcal{A}$
$\mathrm{NTIME}(t(n))^{\mathcal{A}}$	oracle languages that are decidable by a *nondeterministic* machine that runs in time $O(t(n))$ and has oracle access to a sample from $\mathcal{A}$
$\mathrm{PCP}(t(n), q(n))^{\mathcal{A}}$	oracle languages that are decidable by a *PCP verifier* that runs in time $O(t(n))$, makes $O(q(n))$ queries to a proof string, and has oracle access to a sample from $\mathcal{A}$

We introduce two incomparable questions, which concern the (im)possibility of "non-trivial" PCPs for relativized nondeterministic computations and for relativized deterministic computations.

1. **PCPs for NTIME.** For every oracle $\mathcal{A}$ it holds that $\mathrm{NTIME}(t(n))^{\mathcal{A}} \subseteq \mathrm{PCP}(t(n), t(n))^{\mathcal{A}}$ because a PCP verifier can read in full a witness provided in the PCP proof, and then run the nondeterministic decider on the witness. We ask: for what oracles $\mathcal{A}$ can we have *any* non-trivial improvement on this trivial inclusion? Namely, we consider PCP verifiers that may make $o(t(n))$ queries to the PCP proof, which in general prevents the PCP verifier from reading a witness from the PCP proof. We additionally allow the PCP verifier to incur a polynomial blow-up in running time: it may run in time $\mathrm{poly}(t(n))$, and in particular can make $\mathrm{poly}(t(n))$ queries to the sample from $\mathcal{A}$. (The queries to the PCP proof are still $o(t(n))$.) This amounts to asking:

   Given an oracle $\mathcal{A}$, is it the case that $\mathrm{NTIME}(t(n))^{\mathcal{A}} \subseteq \mathrm{PCP}(\mathrm{poly}(t(n)), o(t(n)))^{\mathcal{A}}$*?*

   We will say that an oracle $\mathcal{A}$ *separates* NTIME and PCP if the answer to this question is negative.
2. **PCPs for DTIME.** For every oracle $\mathcal{A}$ it holds that $\mathrm{DTIME}(t(n))^{\mathcal{A}} \subseteq \mathrm{PCP}(t(n), 0)^{\mathcal{A}}$ because a PCP verifier can simply run the deterministic decider. We similarly ask: for what oracles $\mathcal{A}$ can we have *any* non-trivial improvement on this trivial inclusion? Namely,

we consider PCP verifiers that run in time $o(t(n))$, which in general prevents a PCP verifier from simply running the deterministic decider. We additionally allow the PCP verifier to ask any number of queries to the proof or to the sample from $\mathcal{A}$ (as bounded by its running time). This amounts to asking:

Given an oracle $\mathcal{A}$, is it the case that $\mathrm{DTIME}(t(n))^{\mathcal{A}} \subseteq \mathrm{PCP}(o(t(n)), o(t(n)))^{\mathcal{A}}$*?*

We will say that an oracle $\mathcal{A}$ *separates* DTIME and PCP if the answer to this question is negative.

What is known? Recall that, for unrelativized complexity classes, we have excellent PCPs. All nondeterministic computations have a constant-query PCP verifier that runs in polylogarithmic time: $\mathrm{NTIME}(t(n)) \subseteq \mathrm{PCP}(\mathrm{poly}(n, \log t(n)), O(1))$ [24, 19, 37]. In particular, PCPs simultaneously provide exponential savings in witness length and in computation length.

However, for relativized complexity classes, known relativization results tell us very little. The main relevant prior work is by Hartmanis et al. [31], who claim that, with probability 1 over a random function $R\colon \{0,1\}^* \to \{0,1\}$, $\mathrm{NP}^R \not\subseteq \mathrm{PCP}(\mathrm{poly}(n), \log n)^R$. This provides a negative result for the special case where $\mathcal{A}$ is a "random oracle", $t(n)$ is polynomially bounded, and the PCP verifier makes $O(\log n)$ queries to the PCP proof. (In Section 1.4 we discuss other related work.)

However, even for the case of a random oracle, our goal is to rule out *any* non-trivial PCP for *any* nondeterministic computation (ruling out any savings in witness length), and also for any *deterministic* computation (ruling out any savings in computation length, even if there is no witness).

More generally, we are interested to answer these questions for oracles beyond random oracles.

Some intuition. If the PCP verifier could "learn" the oracle in a small number of queries, then we may be able to rely on known techniques to construct PCPs for *un*relativized computations because each oracle call could be replaced by a subroutine that simulates the learned oracle. Conversely, if the oracle is "hard" to learn, then known techniques do not seem to apply because it is not clear how they could deal with oracle calls, and so we may expect that non-trivial PCPs in this case are impossible. Our goal will be to show that, for hard-enough oracles, non-trivial PCPs are indeed impossible (regardless of the techniques that could be used to construct the PCPs).

Beyond PCPs. There are several models of probabilistic proofs beyond PCPs, such as interactive proofs (IPs) [28], interactive PCPs (IPCPs) [33], and interactive oracle proofs (IOPs) [18, 39]. One may ask: why do we focus only on PCPs in our presentation? The answer is that, for the goals of this paper, the PCP model is equivalent to the IOP model (see Remark 6), and the IOP model subsumes the other models as special cases. So, for the goals of this paper, it suffices to study PCPs. All results in Section 1.3 directly translate to IPs, IPCPs, and IOPs.

1.3 Our results

We prove that, for several oracles of cryptographic interest, non-trivial PCPs for relativized computations do not exist – this holds both for deterministic computations (DTIME) and for nondeterministic computations (NTIME). Moreover, we establish several structural results about "hard oracles" for PCPs. These initial results provide us with valuable insights into the efficiency limitations of PCPs, and provide a useful starting point for further investigations into this new direction.

We now summarize our results in more detail.

(1) Random functions. We begin with the oracle that intuition suggests is the "hardest" oracle for PCPs because it is "maximally unlearnable", the *random oracle*. Namely, we consider the oracle $\mathcal{R} = \{\mathcal{R}_n\}_{n\in\mathbb{N}}$ where each $\mathcal{R}_n$ is the uniform distribution over all functions $R_n \colon \{0,1\}^n \to \{0,1\}$.[1] Our first result shows that the intuition is correct, i.e., we prove that the oracle $\mathcal{R}$ separates DTIME and PCP and also separates NTIME and PCP.

▶ **Theorem 2** (informal). *Let $\mathcal{R}$ be the random oracle. For any $t \colon \mathbb{N} \to \mathbb{N}$,*

$$\mathrm{DTIME}(t)^{\mathcal{R}} \not\subseteq \mathrm{PCP}(o(t), o(t))^{\mathcal{R}} \quad \textit{and} \quad \mathrm{NTIME}(t)^{\mathcal{R}} \not\subseteq \mathrm{PCP}(\mathrm{poly}(t), o(t))^{\mathcal{R}} \ .$$

The above theorem tells us that we cannot, in general, expect to construct PCPs for cryptographic computations that involve random oracles, such as Fiat–Shamir signatures [26]. The natural alternative would be to somehow instantiate the random oracle, and incur, within the PCP, the cost of the hash function used in place of the random oracle. This is indeed what Valiant [43] did in his construction of *incrementally verifiable computation* (IVC): Valiant needed to construct a PCP for the computation of a SNARK verifier that uses random oracles and, lacking suitable PCPs for this relativized computation, considered instead the SNARK verifier obtained by instantiating the random oracle. Our Theorem 2 rules out PCPs for computations that use random oracles, and in particular gives strong evidence that Valiant's approach was in some sense justified.

One may argue that, while they give us useful insights, random oracles do not tell us much about other oracles because they are too special in that they have no structure. We now consider two oracles with structure: one with group structure and another with low-degree structure.

(2) Random generic groups. Many group-based cryptographic primitives are stated (and sometimes also analyzed) with respect to a *generic group*. This means that the primitive relies only on the fact that a certain prime-order group is available but does not rely on whether the group is instantiated, say, with a multiplicative subgroup of a finite field or an elliptic curve group. This motivates the question of whether there are PCPs with respect to a *random (generic) group oracle*, which is the oracle $\mathcal{O} = \{\mathcal{O}_n\}_{n\in\mathbb{N}}$ where each $\mathcal{O}_n$ is a random presentation of a group of order n. We prove that the answer is negative, i.e., that the oracle $\mathcal{O}$ separates NTIME and PCP.

▶ **Theorem 3** (informal). *Let $\mathcal{O}$ be the random group oracle. For any $t \colon \mathbb{N} \to \mathbb{N}$,*

$$\mathrm{NTIME}(t)^{\mathcal{O}} \not\subseteq \mathrm{PCP}(\mathrm{poly}(t), o(t))^{\mathcal{O}} \ .$$

The above theorem tells us that, in general, the representation of a group matters to a PCP. For example, if we return to the iterative hash computation of Example 1 and set the hash function to be the Pedersen hash function (a function that is collision resistant over any group where extracting discrete logarithms is hard), we should pick a group tailored to the PCP at hand. This is consistent with the fact that applied cryptographers working with probabilistic proofs have had to carefully design group instantiations for such computations.

[1] More generally, we consider the uniform distribution over functions $R_n \colon \{0,1\}^n \to \{0,1\}^{\ell(n)}$ for some $\ell(n)$.

E.g., `Jubjub` [44] is an elliptic curve in the Zcash cryptocurrency that is used to instantiate a Pedersen hash function in a way that is "friendly" to probabilistic proofs. Our theorem provides strong evidence that these efforts are necessary.

(3) Random low-degree functions. Probabilistic proofs are typically achieved by relying on low-degree functions that encode information associated to the computation being checked. This fact extends to relativized complexity classes in the sense that results such as IP = PSPACE *algebrize* with respect to every oracle [1]. In the language of this paper, this means that for every oracle $\mathcal{A}$, it holds that $\mathrm{PSPACE}^{\mathcal{A}} \subseteq \mathrm{IP}^{\hat{\mathcal{A}}}$ where $\hat{\mathcal{A}}$ is the low-degree extension of $\mathcal{A}$ (each sample in $\mathcal{A}$ is replaced with some low-degree extension of it).

However in this paper we are interested to understand relativization results, not algebrization results, and the above discussion raises the question of what happens when we compare DTIME/NTIME and PCP in a relativized world where the oracle is a random low-degree function. Namely, we consider the oracle $\mathcal{P} = \{\mathcal{P}_n\}_{n\in\mathbb{N}}$ where $\mathcal{P}_n$ is the uniform distribution over all low-degree polynomials on n variables (for given field and degree parameters). Note that $\mathcal{P}$ can be viewed as a low-degree extension of the random oracle $\mathcal{R}$.

We prove that $\mathcal{P}$ separates NTIME and PCP.

▶ **Theorem 4** (informal). *Let $\mathcal{P}$ be the random low-degree oracle. For any $t\colon \mathbb{N}\to\mathbb{N}$,*

$$\mathrm{NTIME}(t)^{\mathcal{P}} \not\subseteq \mathrm{PCP}(\mathrm{poly}(t), o(t))^{\mathcal{P}} \ .$$

Interlude on separation types. We have so far considered relativized complexity classes in which a single machine is granted oracle access to a sample $A\colon \{0,1\}^* \to \{0,1\}^*$ from the oracle $\mathcal{A}$, and is required to "work" for the language defined by A with probability 1 over the choice of A. For example, $\mathrm{DTIME}(t)^{\mathcal{A}}$ is the class of all oracle languages $\mathcal{L} = \{L_A\}_{A\in\mathcal{A}}$ for which there exists a deterministic machine M, which runs in time $O(t(n))$, such that

$$\Pr_{A\leftarrow\mathcal{A}}\Big[\, M^A \text{ decides the language } L_A \,\Big] = 1 \ .$$

We use analogous definitions for NTIME and PCP, as discussed in Section 2.4. We consider these definitions to be the natural ones to use for the goals of this paper. We sometimes refer to separations between these complexity classes as *uniform separations*, to distinguish them from those below.

We could alternatively study separations where all machines are allowed to *non-uniformly depend* on A, thereby granting all machines more power. In this direction, there are two natural definitions.

- *Somewhere separation.* We say that $\mathcal{A}$ provides a somewhere separation for DTIME and PCP if there exists $A\in\mathcal{A}$ such that $\mathrm{DTIME}(t)^A \not\subseteq \mathrm{PCP}(o(t), o(t))^A$. And similarly for NTIME.
- *Almost-everywhere separation.* We say that $\mathcal{A}$ provides an almost-everywhere separation for DTIME and PCP if $\mathrm{DTIME}(t)^A \not\subseteq \mathrm{PCP}(o(t), o(t))^A$ holds with probability 1 over a random choice of sample $A\leftarrow\mathcal{A}$. (Note that this leaves open the possibility that there is no separation for a set of functions of measure 0 in $\mathcal{A}$.) And similarly for NTIME.

An almost-everywhere separation is, in general, strictly stronger than an a somewhere separation. However, the relation between these and uniform separations is not a priori clear.

We provide clarity on this comparison: in the full version we prove that uniform separations are equivalent to somewhere separations, at least when comparing DTIME/NTIME and PCP. Namely, we prove that, for any oracle $\mathcal{A}$, $\mathrm{DTIME}(t)^{\mathcal{A}} \not\subseteq \mathrm{PCP}(o(t), o(t))^{\mathcal{A}}$ if and only if there exists a function A in $\mathcal{A}$ such that $\mathrm{DTIME}(t)^A \not\subseteq \mathrm{PCP}(o(t), o(t))^A$. And similarly for NTIME.

Almost-everywhere separation for random functions. For the random oracle $\mathcal{R}$, we strengthen the separations in Theorem 2 to almost-everywhere separations (via a different, longer proof). We learn that the random oracle $\mathcal{R}$ is particularly "hard" for PCPs.

▶ **Theorem 5** (informal). *Let $\mathcal{R}$ be the random oracle. For any $t\colon \mathbb{N} \to \mathbb{N}$,*

$$\Pr_{R \leftarrow \mathcal{R}} \left[\mathrm{DTIME}(t)^R \not\subseteq \mathrm{PCP}(o(t), o(t))^R \right] = 1$$
$$\textit{and} \quad \Pr_{R \leftarrow \mathcal{R}} \left[\mathrm{NTIME}(t)^R \not\subseteq \mathrm{PCP}(\mathrm{poly}(t), o(t))^R \right] = 1 \ .$$

The above theorem also directly improves on the classical work of Hartmanis et al. [31], who showed that $\Pr_{R \leftarrow \mathcal{R}}[\mathrm{NP}^R \not\subseteq \mathrm{PCP}(\mathrm{poly}(n), \log n)^R] = 1$. The improvement is that our result rules out any non-trivial PCP for any nondeterministic computation, and also rules out any non-trivial PCP for any deterministic computation.

We prove Theorem 5 by building on techniques of Chang et al. [22] that were used to prove that $\Pr_{R \leftarrow \mathcal{R}}[\mathrm{IP}^R \neq \mathrm{PSPACE}^R] = 1$. These techniques rely on the fact that every function R in $\mathcal{R}$ has many other functions in $\mathcal{R}$ that are close to it in all but finitely many points.

We do not know how to extend these techniques to oracles such as the random group oracle $\mathcal{O}$ or the random low-degree oracle $\mathcal{P}$, because in these cases any two samples are far from one another. In this light, we view the techniques that we use to prove Theorems 2 to 4 as more flexible. Moreover, we consider the separations proved in these theorems as sufficient for our motivations.

Structural results: beyond $\mathcal{R}$, $\mathcal{O}$, $\mathcal{P}$. Our results thus far concern separations for specific oracles of interest. There are other oracles of interest that demand understanding (e.g., pseudorandom functions) and, more generally, the study of separations could benefit from general statements. In Section 3 we prove several useful structural results about oracles that are "hard" for PCPs.

1. *Robustness.* We prove that the separating property is "robust" with respect to small perturbations. In more detail, we prove that for every oracle $\mathcal{A}$ that separates DTIME/NTIME and PCP there exists a distance function ϵ such that any other oracle that is ϵ-close to $\mathcal{A}$ also separates DTIME/NTIME and PCP. This statement can also be viewed as telling us that the set of separating oracles is *open* with respect to statistical distance (see Definition 8).
   We can apply the above result to any of the separations that we have proved. For example, if apply it to Theorem 2 then we learn that all oracles that are "almost" uniformly random (close enough to the random oracle $\mathcal{R}$) separate DTIME/NTIME and PCP. In fact, in the full version, we use additional techniques to quantify (a bound on) this distance threshold, proving that all oracles that are $\frac{1}{3e}$-close to uniformly random separate NTIME and PCP.
2. *Monotonicity.* We prove that the separating property is "monotone" in that, for every oracle $\mathcal{A}$ that separates DTIME/NTIME and PCP, if another oracle $\mathcal{B}$ contains $\mathcal{A}$ as a marginal distribution then $\mathcal{B}$ also separates DTIME/NTIME and PCP. I.e., $\mathcal{B}$ inherits the hardness of $\mathcal{A}$.
   We rely on monotonicity in the proof of Theorem 4, where we reduce the problem of showing separation for random *low-degree* polynomials to the problem of showing separation for random *multilinear* polynomials (which we then solve).

3. *Conditioning.* We prove that the separating property is preserved by (finite) conditioning. Namely, given an oracle $\mathcal{A}$ and a function $\mathfrak{f}\colon D \to \{0,1\}^*$ over a finite domain D, we denote by $\mathcal{A}^{D,\mathfrak{f}}$ the oracle where samples are conditioned to equal $\mathfrak{f}$ on D. We prove that if $\mathcal{A}$ separates DTIME/NTIME and PCP then $\mathcal{A}^{D,\mathfrak{f}}$ also separates DTIME/NTIME and PCP.

▶ Remark 6 (beyond PCPs). Research in the last few years has shown that using known PCPs is not the best choice for constructing efficient succinct arguments. Instead, it is better to construct succinct arguments from IOPs [18, 39], which are a multi-round generalization of PCPs that enables significant improvements in asymptotic and concrete efficiency [15, 14, 13, 10, 12, 11, 17, 16, 40]. So the reader may rightfully ask: why did we prove all of our results for PCPs instead of IOPs, if these latter are more powerful?

The answer is that *all of our results extend, in a generic way, to IOPs as well.* This is because our results about PCPs only consider the PCP verifier's time complexity and query complexity, and IOPs do *not* provide any benefits over PCPs when only considering these complexity measures. Indeed, any IOP can be "unrolled" into a PCP, possibly of exponentially larger size, while preserving the verifier's time complexity and query complexity, regardless of oracle. In particular, for every oracle $\mathcal{A}$, the complexity class $\mathrm{IOP}(T,q)^{\mathcal{A}}$ (oracle languages for $\mathcal{A}$ decidable by an IOP verifier with time complexity T and query complexity q) equals the complexity class $\mathrm{PCP}(T,q)^{\mathcal{A}}$ (oracle languages for $\mathcal{A}$ decidable by a PCP verifier with time complexity T and query complexity q).

Finally, we additionally obtain analogous impossibility results for interactive proofs (IPs) [28] and interactive PCPs (IPCPs) [33], as both are special cases of IOPs.

1.4 Related work

NP vs. PCP in relativized worlds. Fortnow [27] uses diagonalization to obtain a function $R\colon \{0,1\}^* \to \{0,1\}$ such that, for every $k \in \mathbb{N}$, $\mathrm{NP}^R \not\subseteq \mathrm{PCP}(\mathrm{poly}(n), n^k)^R$. Hartmanis et al. [31] report a stronger result: with probability 1 over a random function $R\colon \{0,1\}^* \to \{0,1\}$, $\mathrm{NP}^R \not\subseteq \mathrm{PCP}(\mathrm{poly}(n), \log n)^R$. We do not know of a version of [31] that contains a proof of this result, so we cannot comment on the techniques used to prove it. As already discussed, our Theorem 5 strengthens this latter result to hold for any non-trivial PCP.

Barriers for relativization and others. PCP constructions involve the use of non-relativizing techniques. A line of works [27, 5, 1, 32, 7] has developed frameworks that seek to capture the class of such techniques, along with other non-relativizing ones, within formal models, in order to prove barriers for these techniques (e.g., to show that they do not suffice to resolve the P vs. NP question or other difficult questions in complexity theory).

The emphasis and techniques in this work are complementary to the foregoing line of works.

Our emphasis is on establishing impossibility results for PCPs *regardless* of techniques used, as opposed to proving barriers for the PCP techniques that are known today.

Moreover, the axiomatic approaches employed in some of the works cited above cannot be used to even formulate questions that involve PCP verifiers with specific running times or query complexities. E.g., they rely on Cobham's axiomatization of the notion of polynomial time [23], so cannot express exact running times. This means that we would not be able to phrase questions about non-trivial PCPs (as we do in Section 1.2).

1.5 Open problems

The separations that we prove in this paper are for "information-theoretic" oracles $\mathcal{A}$. What can be said about hard "cryptographic" oracles $\mathcal{A}$? E.g., if $\mathcal{A}$ is a pseudo-random function, then must it be the case that $\mathcal{A}$ separates DTIME/NTIME and PCP? What about if $\mathcal{A}$ is a decryption oracle?

More generally, the holy grail in this research direction would be to distill a crisp, and operationally useful, criterion that gives sufficient and necessary condition for an oracle $\mathcal{A}$ that separates DTIME/NTIME and PCP.

2 Definitions

2.1 Oracles

An oracle is a collection of distributions over functions, with one distribution per input length.

▶ **Definition 7.** *An* **oracle** *with output length $\ell\colon \mathbb{N} \to \mathbb{N}$ (with $\ell(n) > 0$ for every $n \in \mathbb{N}$) is a collection $\mathcal{A} = \{\mathcal{A}_n\}_{n\in\mathbb{N}}$ where each $\mathcal{A}_n$ is a distribution over functions $f\colon \{0,1\}^n \to \{0,1\}^{\ell(n)}$.*

We can obtain a *sample* $A\colon \{0,1\}^* \to \{0,1\}^*$ from $\mathcal{A}$ by sampling a function f_n from each $\mathcal{A}_n$ and then setting A to equal f_n for inputs of size n. We write "$A \leftarrow \mathcal{A}$" to denote that A is a sample that follows this distribution, and "$A \in \mathcal{A}$" to denote that A is in the support of this distribution. We denote by $\mathrm{supp}(\mathcal{A})$ the support of $\mathcal{A}$.

An oracle $\mathcal{A}$ induces a corresponding probability measure $\mu_{\mathcal{A}}$ over the space of functions from binary strings to binary strings: given a subset $S \subseteq \{0,1\}^*$ and a set X of functions from S to $\{0,1\}^*$, $\mu_{\mathcal{A}}(X)$ is the probability that the restriction to S of a sample A from $\mathcal{A}$ belongs to X.

▶ **Definition 8.** *Two oracles $\mathcal{A} = \{\mathcal{A}_n\}_{n\in\mathbb{N}}$ and $\mathcal{B} = \{\mathcal{B}_n\}_{n\in\mathbb{N}}$ have (statistical) distance $\epsilon\colon \mathbb{N} \to \mathbb{N}$ if, for every $n \in \mathbb{N}$, the statistical distance between the distributions $\mathcal{A}_n$ and $\mathcal{B}_n$ is at most $\epsilon(n)$.*

We write "$g \leftarrow \mathcal{A}_{\leq n}$" to denote that g is a function on $\{0,1\}^{\leq n}$ that is sampled from the distribution $\mathcal{A}_{\leq n} := \mathcal{A}_1 \times \cdots \times \mathcal{A}_n$. We write "$g \in \mathcal{A}_{\leq n}$" to denote that g is in the support of this distribution, and denote by $\mathrm{supp}(\mathcal{A}_{\leq n})$ the support of $\mathcal{A}_{\leq n}$.

▶ **Definition 9.** *An oracle $\mathcal{B}$* **contains** *an oracle $\mathcal{A}$ if for all $n \in \mathbb{N}$ it holds that $\mu_{\mathcal{B}}(\mathrm{supp}(\mathcal{A}_{\leq n})) > 0$ and, for every $\mathfrak{f} \in \mathrm{supp}(\mathcal{A}_{\leq n})$, $\mu_{\mathcal{B}}(\mathfrak{f}) = \mu_{\mathcal{B}}(\mathrm{supp}(\mathcal{A}_{\leq n})) \cdot \mu_{\mathcal{A}}(\mathfrak{f})$.*

The definition below provides an operation to *condition* an oracle to take known values.

▶ **Definition 10.** *Fix a subset $D \subseteq \{0,1\}^*$ and function $\mathfrak{f}\colon D \to \{0,1\}^*$.*

- *Given a function $A\colon \{0,1\}^* \to \{0,1\}^*$, we define $A^{D,\mathfrak{f}}\colon \{0,1\}^* \to \{0,1\}^*$ to be the function obtained by setting the values of A on D to $\mathfrak{f}$:*

$$A^{D,\mathfrak{f}} = \begin{cases} \mathfrak{f}(x) & \text{if } x \in D \\ A(x) & \text{if } x \notin D \end{cases}.$$

- *Given an oracle $\mathcal{A} = \{\mathcal{A}_n\}_{n\in\mathbb{N}}$ (such that there exists some $A \in \mathrm{supp}(\mathcal{A})$ agreeing with $\mathfrak{f}$ on D), we define $\mathcal{A}^{D,\mathfrak{f}} = \{\mathcal{A}_n^{D,\mathfrak{f}}\}_{n\in\mathbb{N}}$ to be the oracle where samples are conditioned to equal $\mathfrak{f}$ on D. In more detail, each distribution $\mathcal{A}_n^{D,\mathfrak{f}}$ equals the distribution $\mathcal{A}_n$ conditioned on the event that the sampled function agrees with $\mathfrak{f}$ on $D \cap \{0,1\}^n$.*

2.2 Languages and oracle languages

A language L is a subset of $\{0,1\}^*$. We denote by $L(\mathbb{x})$ the bit that specifies whether a string $\mathbb{x} \in \{0,1\}^*$ is in L ($L(\mathbb{x}) = 1$) or not ($L(\mathbb{x}) = 0$).

We also consider oracle languages because we study relativized complexity classes.

▶ **Definition 11.** *Let $\mathcal{U} := \{F\colon \{0,1\}^* \to \{0,1\}^*\}$ be the set of all functions on binary strings. An* **oracle language** *$\mathcal{L}$ is a collection of languages indexed by functions $F \in \mathcal{U}$, namely, $\mathcal{L} = \{L_F\}_{F\in\mathcal{U}}$ where each L_F is a subset of $\{0,1\}^*$.*

A language L can be viewed as a special case of an oracle language $\{L_F\}_{F\in\mathcal{U}}$ where each $L_F = L$.

▶ **Definition 12.** *Let $\ell\colon \mathbb{N} \to \mathbb{N}$ be a computable function. An oracle language $\mathcal{L} = \{L_F\}_{F\in\mathcal{U}}$ is* **ℓ-bounded** *if, for all functions $F \in \mathcal{U}$ and inputs $\mathbb{x} \in \{0,1\}^*$, whether $\mathbb{x} \in L_F$ only depends on F's values at locations of size at most $\ell(|\mathbb{x}|)$. Namely, $L_F(\mathbb{x}) = L_{F'}(\mathbb{x})$ for every F' that agrees with F on the set $\bigcup_{1\le i\le \ell(|\mathbb{x}|)}\{0,1\}^i$.*

2.3 Machines that query oracles

We consider several notions of (Turing) machines that query oracles, as defined below. Informally, an *oracle machine* is given black-box access to a function $A\colon \{0,1\}^* \to \{0,1\}^*$, which the machine can query, any number of times, at any input of its choice. Each query costs the machine a single computational step, regardless of the function A. In more detail, we consider the following definition.

▶ **Definition 13.** *An* **oracle machine** *M is a machine that has two special tapes called* oracle query tape *and* oracle answer tape*, and two special states called* Query *and* Answer*. The special tapes are in addition to the machine's regular read/write tapes (of which there can be one or multiple) and the special states are in addition to the machine's regular start, accept, reject, and other states. We denote by $M^A(\mathbb{x})$ the output of M on input $\mathbb{x} \in \{0,1\}^*$ and with access to oracle $A\colon \{0,1\}^* \to \{0,1\}^*$, which is computed as follows. The input $\mathbb{x}$ is written in a designated read/write tape, and execution proceeds as normal except if the machine enters the* Query *state. Let $y \in \{0,1\}^*$ be the contents of the oracle query tape when this happens. In the following step, the contents of the oracle answer tape are replaced with $A(y) \in \{0,1\}^*$, and the machine enters the* Answer *state.*

Definition 13 considers *deterministic* oracle machines. In Section 2.4 we use these machines to extend the notion of languages decidable in deterministic bounded time to work with oracles.

We also use *nondeterministic* oracle machines, which are defined similarly as above except that they can, in any computational step, choose to make a nondeterministic choice as in the standard definition of a nondeterministic machine. In Section 2.4 we use these machines to extend the notion of languages decidable in nondeterministic bounded time to work with oracles.

We also use *probabilistic* oracle machines that use randomness and can make queries to a proof string $\pi \in \{0,1\}^*$ (in addition to the oracle A). These machines are defined similarly as above except that they can, in any computational step, receive a bit of randomness, or query a location of the proof string π via two dedicated tapes (a *proof query tape* and a *proof answer tape*). In Section 2.4 we use these machines to extend the notion of probabilistic proofs to work with oracles.

Throughout this paper we call oracle machines simply "machines", as it will be clear from context when we are referring to an oracle machine (of one of the foregoing types).

▶ Remark 14. Oracle machines are often defined with *one* special tape, instead of *two* as in Definition 13. The machine writes its query to the oracle in this one tape, and in the following step the tape's contents are *replaced* with the oracle's answer. Note that, with one oracle tape, the machine has to write each query "from scratch" because the prior query was deleted. This difference is not significant, because writing each query from scratch is at most quadratically slower than not having to do that (due to having two oracle tapes). However, this difference matters when studying questions that are sensitive to such costs. Thus in this paper we use machines with two oracle tapes.

2.4 Complexity classes with oracles

We define, for a given oracle $\mathcal{A}$, the complexity classes $\mathrm{DTIME}(t)^{\mathcal{A}}$, $\mathrm{NTIME}(t)^{\mathcal{A}}$, and $\mathrm{PCP}(t,q)^{\mathcal{A}}$.

Deterministic time. A deterministic machine M is a *D-decider* for a language L if for every $\mathbb{x} \in \{0,1\}^*$ it holds that $M(\mathbb{x}) = L(\mathbb{x})$. The complexity class $\mathrm{DTIME}(t)$ consists of all languages L for which there exists a deterministic machine M that runs in time $O(t(n))$ and is a D-decider for L. We now provide a definition that considers the more general case of oracle languages that are decidable by deterministic machines with access to an oracle.

▶ **Definition 15.** *Let $\mathcal{A} = \{\mathcal{A}_n\}_{n\in\mathbb{N}}$ be an oracle and let $t\colon \mathbb{N} \to \mathbb{N}$ be a function.* $\mathrm{DTIME}(t)^{\mathcal{A}}$ *is the class of all oracle languages $\mathcal{L} = \{L_A\}_{A\in\mathcal{A}}$ for which there exists a deterministic machine M, which runs in time $O(t(n))$, such that*

$$\Pr_{A\leftarrow\mathcal{A}}\left[M^A \text{ is a D-decider for } L_A\right] = 1 \ .$$

Nondeterministic time. A nondeterministic machine M is a *ND-decider* for a language L if for every $\mathbb{x} \in \{0,1\}^*$ it holds that $M(\mathbb{x}) = L(\mathbb{x})$. The complexity class $\mathrm{NTIME}(t)$ consists of all languages L for which there exists a nondeterministic machine M that runs in time $O(t(n))$ and is a ND-decider for L. We now provide a definition that considers the more general case of oracle languages that are decidable by nondeterministic machines with access to an oracle.

▶ **Definition 16.** *Let $\mathcal{A} = \{\mathcal{A}_n\}_{n\in\mathbb{N}}$ be an oracle and let $t\colon \mathbb{N} \to \mathbb{N}$ be a function.* $\mathrm{NTIME}(t)^{\mathcal{A}}$ *is the class of all oracle languages $\mathcal{L} = \{L_A\}_{A\in\mathcal{A}}$ for which there exists a nondeterministic machine M, which runs in time $O(t(n))$, such that*

$$\Pr_{A\leftarrow\mathcal{A}}\left[M^A \text{ is a ND-decider for } L_A\right] = 1 \ .$$

Probabilistic proofs. A probabilistic machine M is a *PCP-verifier* for a language L if: for every $\mathbb{x} \in L$ there exists $\pi \in \{0,1\}^*$ such that $\Pr[M^\pi(\mathbb{x}) = 1] \geq 2/3$; for every $\mathbb{x} \notin L$ and $\pi \in \{0,1\}^*$ it holds that $\Pr[M^\pi(\mathbb{x}) = 0] \geq 2/3$. The complexity class $\mathrm{PCP}(t,q)$ consists of all languages L for which there exists a probabilistic machine M that runs in time $O(t(n))$, makes $O(q(n))$ queries to the proof string, and is a PCP-verifier for L. Below we consider the more general case of oracle languages that are decidable by probabilistic machines with access to an oracle.

▶ **Definition 17.** *Let $\mathcal{A} = \{\mathcal{A}_n\}_{n\in\mathbb{N}}$ be an oracle and let $t, q\colon \mathbb{N} \to \mathbb{N}$ be functions.* $\mathrm{PCP}(t,q)^{\mathcal{A}}$ *is the class of all oracle languages $\mathcal{L} = \{L_A\}_{A\in\mathcal{A}}$ for which the there exists a probabilistic machine M, which runs in time $O(t(n))$ and makes $O(q(n))$ queries to the proof string, such that*

$$\Pr_{A\leftarrow\mathcal{A}}\left[\, M^A \text{ is a PCP-verifier for } L_A \,\right] = 1 \ .$$

▶ **Definition 18.** *Let $\mathcal{L} = \{L_A\}_{A\in\mathcal{A}}$ be an oracle language. An oracle machine M* **fails** *on an input $\mathbb{x} \in \{0,1\}^*$ and function $A \in \mathcal{A}$ for $\mathcal{L}$ if the following holds: if $\mathbb{x} \in L_A$ then $\Pr_r[M^{A,\pi}(\mathbb{x};r) = 1] < \frac{2}{3}$ for every proof π; else if $\mathbb{x} \notin L_A$ then $\Pr_r[M^{A,\pi}(\mathbb{x};r) = 1] > \frac{1}{3}$ for some proof π.*

We conclude this section with several technical remarks.

▶ Remark 19 (bounded oracle languages). For every oracle $\mathcal{A}$ and oracle language $\mathcal{L}$ in the complexity class $\mathrm{DTIME}(t)^{\mathcal{A}}$, $\mathrm{NTIME}(t)^{\mathcal{A}}$, or $\mathrm{PCP}(t,q)^{\mathcal{A}}$, there exists a $O(t)$-bounded oracle language $\mathcal{L}^*$ such that for every oracle $A \in \mathcal{A}$ it holds that $L_A = L^*_A$. The language $\mathcal{L}^*$ is naturally defined by $\mathcal{L}$'s $\mathrm{DTIME}^{\mathcal{A}}$ decider, $\mathrm{NTIME}^{\mathcal{A}}$ decider, or $\mathrm{PCP}^{\mathcal{A}}$ verifier. In particular, we can assume without loss of generality that oracle languages in these complexity classes are $O(t)$-bounded.

▶ Remark 20 (index over $\mathcal{A}$ instead of $\mathcal{U}$). We sometimes define an oracle language $\mathcal{L}$ in $\mathrm{DTIME}(t)^{\mathcal{A}}$ or in $\mathrm{NTIME}(t)^{\mathcal{A}}$ only for functions in (the support of) an oracle $\mathcal{A}$. In this case it is understood that L_F is defined by the $\mathrm{DTIME}^{\mathcal{A}}$ or $\mathrm{NTIME}^{\mathcal{A}}$ decider of $\{L_A\}_{A\in\mathcal{A}}$ for all functions $F \in \mathcal{U} \setminus \mathcal{A}$.

▶ Remark 21 (relativized classes for a single function). Definitions 15 to 17 capture, as a special case, relativized classes where the oracle is a single function $A\colon \{0,1\}^* \to \{0,1\}^*$ rather than a distribution over functions (let $\mathcal{A}$ be the oracle that puts all the probability mass on A). In this case the relativized classes can be "collapsed" to sets of languages rather than sets of oracle languages.

3 Structural properties of separation

We prove structural properties about the non-containments $\mathrm{DTIME}(t)^{\mathcal{A}} \not\subseteq \mathrm{PCP}(T,q)^{\mathcal{A}}$ (Theorem 22) and $\mathrm{NTIME}(t)^{\mathcal{A}} \not\subseteq \mathrm{PCP}(T,q)^{\mathcal{A}}$ (Theorem 23). We use these in later sections.

▶ **Theorem 22** (DTIME & PCP)**.** *Let $t, T\colon \mathbb{N} \to \mathbb{N}$ be time bound functions and $q\colon \mathbb{N} \to \mathbb{N}$ a query bound function. Let $\mathcal{A}$ be an oracle such that $\mathrm{DTIME}(t)^{\mathcal{A}} \not\subseteq \mathrm{PCP}(T,q)^{\mathcal{A}}$. Then the following holds.*

1. Robustness: *there exists a positive function $\epsilon\colon \mathbb{N} \to \mathbb{R}$ ($\epsilon(n) > 0$ for every $n \in \mathbb{N}$) such that, for every oracle $\mathcal{B}$ that is ϵ-close to $\mathcal{A}$, it also holds that $\mathrm{DTIME}(t)^{\mathcal{B}} \not\subseteq \mathrm{PCP}(T,q)^{\mathcal{B}}$.*
2. Monotonicity: *for every oracle $\mathcal{B}$ that contains $\mathcal{A}$, it also holds that $\mathrm{DTIME}(t)^{\mathcal{B}} \not\subseteq \mathrm{PCP}(T,q)^{\mathcal{B}}$.*
3. Conditioning: *for every function $\mathfrak{f}\colon D \to \{0,1\}^*$, it also holds that $\mathrm{DTIME}(t)^{\mathcal{A}^{D,\mathfrak{f}}} \not\subseteq \mathrm{PCP}(T,q)^{\mathcal{A}^{D,\mathfrak{f}}}$.*

▶ **Theorem 23** (NTIME & PCP)**.** *Theorem 22 also holds with $\mathrm{NTIME}(t)$ in place of $\mathrm{DTIME}(t)$.*

The above theorems are direct corollaries of general properties that we prove, as we now explain.

For the rest of this section we fix: (a) an oracle language $\mathcal{L}$ that is ℓ-bounded for some $\ell\colon \mathbb{N} \to \mathbb{N}$; (b) a time bound function $T\colon \mathbb{N} \to \mathbb{N}$; (c) a query bound function $q\colon \mathbb{N} \to \mathbb{N}$; (d) an oracle $\mathcal{A}$.

We shall provide an equivalent formulation for the condition "$\mathcal{L} \notin \mathrm{PCP}(T,q)^{\mathcal{A}}$" (Claim 26), and then use it to derive several general properties about this condition: robustness (Lemma 27), monotonicity (Lemma 28), and conditioning (Lemma 29).

By taking $\mathcal{L}$ to be an oracle language in either $\mathrm{DTIME}(t)^{\mathcal{A}}$ or $\mathrm{NTIME}(t)^{\mathcal{A}}$ (which implies that the oracle language is $O(t)$-bounded), we can then derive the corresponding property in Theorem 22 or Theorem 23 respectively. We are left to state and prove the claim and lemmas mentioned above.

▶ **Definition 24.** *We denote by $\mathbf{M}_{T,q}$ the set of probabilistic oracle machines that, on inputs of length n, read $O(q(n))$ proof bits and run in $O(T(n))$ time.*

▶ **Definition 25.** *For every $n \in \mathbb{N}$, $s(n) := \max\{\ell(n), 2^{T(n)}\}$ and $S_n := \bigcup_{1 \le i \le s(n)} \{0,1\}^i$.*

▷ Claim 26. The following two conditions are equivalent:

1. $\mathcal{L} \notin \mathrm{PCP}(T,q)^{\mathcal{A}}$.
2. For every machine $M \in \mathbf{M}_{T,q}$ there exist an input $\mathbb{x} \in \{0,1\}^*$ and function $\mathfrak{f}\colon S_{|\mathbb{x}|} \to \{0,1\}^*$ with $\mu_{\mathcal{A}}(\mathfrak{f}) > 0$ such that, for every function $F\colon \{0,1\}^* \to \{0,1\}^*$ that agrees with $\mathfrak{f}$ on $S_{|\mathbb{x}|}$, M fails on input $\mathbb{x}$ and function F for $\mathcal{L}$. (The function F need not be in $\mathrm{supp}(\mathcal{A})$.)

Proof. We separately consider the two directions.

(1) ⇒ (2). If $\mathcal{L} \notin \mathrm{PCP}(T,q)^{\mathcal{A}}$, then for every $M \in \mathbf{M}_{T,q}$ there exists a function $F \in \mathrm{supp}(\mathcal{A})$ such that M^F fails to verify L_F on some input $\mathbb{x}$. Consider the function $\mathfrak{f}\colon S_{|\mathbb{x}|} \to \{0,1\}^*$ obtained by restricting F to $S_{|\mathbb{x}|}$. Since the running time of M^F on input $\mathbb{x}$ is at most $O(T(|\mathbb{x}|))$, it cannot distinguish between having access to the function F and access to any other function F' that agrees with $\mathfrak{f}$. Moreover, since $\mathcal{L}$ is ℓ-bounded, $L_F(\mathbb{x}) = L_{F'}(\mathbb{x})$ for every F' that agrees with $\mathfrak{f}$. Therefore, $M^{F'}(\mathbb{x})$ fails for every F' that agrees with $\mathfrak{f}$. The set of all such oracles has positive measure in $\mathcal{A}$ (i.e., $\mu_{\mathcal{A}}(\mathfrak{f}) > 0$), because any finite prefix of any function in $\mathcal{A}$ has positive measure.

(2) ⇒ (1). The condition directly implies that, for every $M \in \mathbf{M}_{T,q}$, M^A is not a PCP-verifier for L_A for a set of functions A with positive measure in $\mathcal{A}$. Therefore $\mathcal{L} \notin \mathrm{PCP}(T,q)^{\mathcal{A}}$. ◀

▶ **Lemma 27** (robustness). *If $\mathcal{L} \notin \mathrm{PCP}(T,q)^{\mathcal{A}}$, then there exists a positive function $\epsilon\colon \mathbb{N} \to \mathbb{R}$ ($\epsilon(n) > 0$ for every $n \in \mathbb{N}$) such that, for every oracle $\mathcal{B}$ that is ϵ-close to $\mathcal{A}$, $\mathcal{L} \notin \mathrm{PCP}(T,q)^{\mathcal{B}}$.*

Proof. For every $n \in \mathbb{N}$, define $X_n := \{\mathfrak{f}\colon S_n \to \{0,1\}^* \mid \mu_{\mathcal{A}}(\mathfrak{f}) > 0\}$ to be the set of all functions over S_n that have positive measure in $\mathcal{A}$. We define the distance function as $\epsilon(n) := \min_{\mathfrak{f} \in X_n} \mu_{\mathcal{A}}(\mathfrak{f})$.

By Claim 26, from $\mathcal{L} \notin \mathrm{PCP}(T,q)^{\mathcal{A}}$ we deduce that for any $M \in \mathbf{M}_{T,q}$ there exist an input $\mathbb{x}$ and function $\mathfrak{f}\colon S_{|\mathbb{x}|} \to \{0,1\}^*$ with $\mu_{\mathcal{A}}(\mathfrak{f}) > 0$ such that, for every function $F\colon \{0,1\}^* \to \{0,1\}^*$ that agrees with $\mathfrak{f}$ on $S_{|\mathbb{x}|}$, M fails on input $\mathbb{x}$ and function F for $\mathcal{L}$. Since the oracle $\mathcal{B}$ is ϵ-close to $\mathcal{A}$ we deduce, from the definition of ϵ, that $\mu_{\mathcal{B}}(\mathfrak{f}) > 0$ as well.

Since the above holds for every $M \in \mathbf{M}_{T,q}$, by Claim 26 we conclude that $\mathcal{L} \notin \mathrm{PCP}(T,q)^{\mathcal{B}}$. ◀

▶ **Lemma 28** (monotonicity). *If $\mathcal{L} \notin \mathrm{PCP}(T,q)^{\mathcal{A}}$, then, for every oracle $\mathcal{B}$ that contains $\mathcal{A}$, it holds that $\mathcal{L} \notin \mathrm{PCP}(T,q)^{\mathcal{B}}$.*

Proof. From Claim 26, since $\mathcal{L} \notin \mathrm{PCP}(T,q)^{\mathcal{A}}$, we know that for every $M \in \mathbf{M}_{T,q}$ there exist an input $\mathbb{x}$ and function $\mathfrak{f}\colon S_{|\mathbb{x}|} \to \{0,1\}^*$ with $\mu_{\mathcal{A}}(\mathfrak{f}) > 0$ such that, for every function $F\colon \{0,1\}^* \to \{0,1\}^*$ that agrees with $\mathfrak{f}$ on $S_{|\mathbb{x}|}$, M fails on input $\mathbb{x}$ and function F for $\mathcal{L}$. Due to containment (Definition 9), since $\mathfrak{f} \in \mathrm{supp}(\mathcal{A}_{\leq s(|\mathbb{x}|)})$, we deduce that

$$\mu_{\mathcal{B}}(\mathfrak{f}) = \mu_{\mathcal{B}}\left(\mathrm{supp}(\mathcal{A}_{\leq s(|\mathbb{x}|)})\right) \cdot \mu_{\mathcal{A}}(\mathfrak{f}) > 0 \ .$$

Since the above holds for every $M \in \mathbf{M}_{T,q}$, by Claim 26 we conclude that $\mathcal{L} \notin \mathrm{PCP}(T,q)^{\mathcal{B}}$. ◀

▶ **Lemma 29** (conditioning). *Let $I \subseteq \mathbb{N}$ be finite, and set $D := \bigcup_{i\in I}\{0,1\}^i$. If $\mathcal{L} \notin \mathrm{PCP}(T,q)^{\mathcal{A}}$, then there exists a function $\mathfrak{g}\colon D \to \{0,1\}^*$ such that, letting $\mathcal{L}^{\mathfrak{g}} := \{L_F^{\mathfrak{g}}\}_{F\in\mathcal{U}}$ where each $L_F^{\mathfrak{g}} := L_{F^{D,\mathfrak{g}}}$, for every $\mathfrak{f}\colon D \to \{0,1\}^*$ in the support of $\mathcal{A}$ it holds that $\mathcal{L}^{\mathfrak{g}} \notin \mathrm{PCP}(T,q)^{\mathcal{A}^{D,\mathfrak{f}}}$.*

Proof. Consider the following set of functions over D:

$$G_{D,\mathcal{A}} := \{\mathfrak{g}\colon D \to \{0,1\}^* \mid \exists A \in \mathrm{supp}(\mathcal{A}) \text{ that agrees with } \mathfrak{g} \text{ on } D\} \ .$$

First, we argue that, since $\mathcal{L} \notin \mathrm{PCP}(T,q)^{\mathcal{A}}$, there exists $\mathfrak{g} \in G_{D,\mathcal{A}}$ such that $\mathcal{L} \notin \mathrm{PCP}(T,q)^{\mathcal{A}^{D,\mathfrak{g}}}$. Suppose by way of contradiction that $\mathcal{L} \in \mathrm{PCP}(T,q)^{\mathcal{A}^{D,\mathfrak{g}}}$ for every $\mathfrak{g} \in G_{D,\mathcal{A}}$. Then every oracle $\mathcal{A}^{D,\mathfrak{g}}$ has a PCP-verifier $M_{\mathfrak{g}} \in \mathbf{M}_{T,q}$ for $\mathcal{L}$. We use the PCP-verifiers $\{M_{\mathfrak{g}}\}_{\mathfrak{g}\in G_{D,\mathcal{A}}}$ to construct a PCP-verifier M that shows that $\mathcal{L} \in \mathrm{PCP}(T,q)^{\mathcal{A}}$ (a contradiction): $M^{A,\pi}(\mathbb{x})$ first queries all locations in D to identify which $\mathfrak{g} \in G_{D,\mathcal{A}}$ is consistent with A; then it rules according to $M_{\mathfrak{g}}^{A,\pi}(\mathbb{x})$. By construction, the machine M is in $\mathbf{M}_{T,q}$ because querying all locations in D takes a constant amount of time and involves a constant number of queries. (The size of D is a finite constant.)

Next, we use $\mathcal{L} \notin \mathrm{PCP}(T,q)^{\mathcal{A}^{D,\mathfrak{g}}}$ to argue that $\mathcal{L}^{\mathfrak{g}} \notin \mathrm{PCP}(T,q)^{\mathcal{A}^{D,\mathfrak{f}}}$. By definition of $\mathcal{L}^{\mathfrak{g}}$, for every $F \in \mathrm{supp}(\mathcal{A}^{D,\mathfrak{g}})$ we have $L_F = L^{\mathfrak{g}}_{F^{D,\mathfrak{f}}}$. Moreover, since D is the union of binary strings of certain lengths, there is a bijection between functions $F \in \mathrm{supp}(\mathcal{A}^{D,\mathfrak{g}})$ and functions $F^{D,\mathfrak{f}} \in \mathrm{supp}(\mathcal{A}^{D,\mathfrak{f}})$, and $\mu_{\mathcal{A}^{D,\mathfrak{g}}}(F) = \mu_{\mathcal{A}^{D,\mathfrak{f}}}(F^{D,\mathfrak{f}})$. This means that if the oracle $\mathcal{A}^{D,\mathfrak{f}}$ has a PCP-verifier $M_{\mathfrak{f}} \in \mathbf{M}_{T,q}$ for $\mathcal{L}^{\mathfrak{g}}$, then we can construct a machine $M_{\mathfrak{g}}$ that, relative to the oracle $\mathcal{A}^{D,\mathfrak{g}}$, is a PCP-verifier for $\mathcal{L}$: $M_{\mathfrak{g}}^{A,\pi}(\mathbb{x})$ runs $M_{\mathfrak{f}}(\mathbb{x})$ except that $M_{\mathfrak{g}}$ answers any query $y \in D$ from $M_{\mathfrak{f}}$ with $\mathfrak{f}(y)$ instead of $A(y)$. One can verify that $M_{\mathfrak{g}} \in \mathbf{M}_{T,q}$, which means that $\mathcal{L} \in \mathrm{PCP}(T,q)^{\mathcal{A}^{D,\mathfrak{g}}}$, a contradiction. ◀

4 Separations for random functions

We define the notion of a random oracle and then state our separation results for it.

▶ **Definition 30.** *A* **random oracle** *with output length $\ell\colon \mathbb{N} \to \mathbb{N}$ is the oracle $\mathcal{R} = \{\mathcal{R}_n\}_{n\in\mathbb{N}}$ where each $\mathcal{R}_n$ is the uniform distribution over functions $f\colon \{0,1\}^n \to \{0,1\}^{\ell(n)}$.*

The probability measure of $\mathcal{R}$ is uniform, in the sense that, for any choice of distinct binary strings $x_1,\ldots,x_m$ of lengths $n_1,\ldots,n_m$ and choice of binary strings $b_1,\ldots,b_m$ of lengths $\ell(n_1),\ldots,\ell(n_m)$, the set $S := \{A \mid A(x_1) = b_1,\ldots,A(x_m) = b_m\}$ has measure $\mu_{\mathcal{R}}(S) = 1/(\Pi_{i=1}^m 2^{\ell(n_i)})$.

▶ **Theorem 31.** *Let $\mathcal{R}$ be the random oracle with output length $\ell\colon \mathbb{N} \to \mathbb{N}$.*

1. *For any function $t\colon \mathbb{N} \to \mathbb{N}$ with $t(n) \in \Omega(n)$,*

$$\mathrm{NTIME}(t)^{\mathcal{R}} \not\subseteq \mathrm{PCP}(\mathrm{poly}(t), o(t))^{\mathcal{R}} \ .$$

2. *For any function $t\colon \mathbb{N} \to \mathbb{N}$ with $t(n) \in \Omega(n) \bigcap o(2^n)$,*

$$\mathrm{DTIME}(t)^{\mathcal{R}} \not\subseteq \mathrm{PCP}(o(t), o(t))^{\mathcal{R}} \ .$$

▶ Remark 32. In Section 7 we prove a stronger (almost-everywhere) separation result. We provide a standalone proof of Theorem 31 because the techniques used to prove it can be modified to apply to other oracles. We do not know how to prove stronger separations for other oracles.

4.1 Proof of Part 1 of Theorem 31

We exhibit an oracle language $\mathcal{L}$ that is in $\mathrm{NTIME}(t)^{\mathcal{R}}$ but not in $\mathrm{PCP}(\mathrm{poly}(t), o(t))^{\mathcal{R}}$.

Oracle language. Let $e_{k,i}$ denote the $\lceil \log k \rceil$-bit string that represents the index $i \in [k]$. The oracle language $\mathcal{L} = \{L_R\}_{R \in \mathcal{R}}$ is defined as follows:

$$L_R := \left\{ 0^n \;\middle|\; \exists\, w \in \{0,1\}^{t(n)} \text{ s.t. } \begin{array}{c} R(w \| e_{t(n),1})_1 = 0 \\ R(w \| e_{t(n),2})_1 = 0 \\ \vdots \\ R(w \| e_{t(n),t(n)})_1 = 0 \end{array} \right\} .$$

Note that L_R does not contain any string that is not all-zeros. Strings of the form 0^n may or may not be in L_R, depending on the answers from R.

In NTIME. We argue that $\mathcal{L}$ is in $\mathrm{NTIME}(t)^{\mathcal{R}}$. Consider the nondeterministic machine that, for inputs of the form 0^n, expects as nondeterministic witness a string $w \in \{0,1\}^{t(n)}$ and checks, via $t(n)$ calls to R, if R returns a string whose first bit is zero on input $w \| e_{t(n),i}$ for every $i \in \{1, \ldots, t(n)\}$. The machine rejects any input not of the form 0^n. This machine, for any given R, decides the language L_R on every input. The machine's running time is $O(t(n))$: writing the first query costs $O(t(n))$ steps and updating the query tape with each new subsequent query costs $O(1)$ steps.

Not in PCP. We argue that $\mathcal{L}$ is not in $\mathrm{PCP}(\mathrm{poly}(t), o(t))^{\mathcal{R}}$. Suppose by way of contradiction that $\mathcal{L}$ has a PCP-verifier $M \in \mathbf{M}_{\mathrm{poly}(t),o(t)}$, and denote by $T(n)$ the running time of M on input of size n. For every $n \in \mathbb{N}$, define the domain $S_n := \bigcup_{1 \le i \le T(n)} \{0,1\}^i$ and the following set X_n of functions over S_n:

$$X_n = \left\{ f \colon S_n \to \{0,1\}^* \;\middle|\; \begin{array}{l} \exists\, ! \, w \in \{0,1\}^{t(n)} \text{ such that both conditions below hold} \\ \bullet\ \forall i \in [t], f(x \| e_{t,i}) = 01^{\ell(t + \lceil \log t \rceil) - 1} \\ \bullet\ \forall y \notin \{w \| e_{t,j}\}_{j \in [t]}, f(y) = 1^{\ell(|y|)} \end{array} \right\} .$$

Note that, for every $n \in \mathbb{N}$, every function $f \in X_n$ has measure $\mu_{\mathcal{R}}(\{f\}) > 0$.

We also use $\mathbf{1}$ to denote the all-one function that on an input $z \in \{0,1\}^*$ returns $1^{\ell(|z|)}$. We derive a contradiction from the following two (contradicting) statements.

- Lemma 33: For every $n \in \mathbb{N}$ and function R agreeing with some $f \in X_n$ there exists a proof π such that $M^{\mathbf{1},\pi}(0^n)$ queries $\mathbf{1}$ at some "witness location" $w \| e_{t(n),i}$ of R with probability at least $\frac{1}{3}$.
- Lemma 34: For every $n \in \mathbb{N}$ there exists a function R agreeing with some $f \in X_n$ such that for any proof π it holds that $M^{\mathbf{1},\pi}(0^n)$ queries $\mathbf{1}$ at some "witness location" $w \| e_{t(n),i}$ of R with probability only $o(1)$.

We are left to prove the lemmas. We abbreviate $t(n)$ with t as the choice of n is clear from context.

▶ **Lemma 33.** *If $M \in \mathbf{M}_{\mathrm{poly}(t),o(t)}$ is a PCP-verifier for $\mathcal{L}$, then for every $n \in \mathbb{N}$ and function R agreeing with some $f \in X_n$ there exists a proof π s.t.*

$$\Pr_r\left[M^{\mathbf{1},\pi}(0^n;r) \text{ queries } \mathbf{1} \text{ at some } w\|e_{t,i} \text{ s.t. } R(w\|e_{t,i})_1 = 0\right] > \frac{1}{3} \ .$$

Proof. By definition of X_n, $0^n \in L_R$ for any R agreeing with some $f \in X_n$. Therefore, for every such R, there exists a proof π such that $M^{R,\pi}(0^n)$ accepts with probability at least $\frac{2}{3}$. We also note that since $M^{R,\pi}(0^n)$ has running time $T(n)$, it cannot make oracle queries outside the set S_n.

We now argue that for every R agreeing with some $f \in X_n$ there exists a proof π such that

$$\Pr_r\left[M^{R,\pi}(0^n;r) \text{ queries } R \text{ at some } w\|e_{t,i} \text{ s.t. } R(w\|e_{t,i}) = 0\right] > \frac{1}{3} \ . \tag{1}$$

Suppose Equation (1) does not hold. Then more than $\frac{2}{3}$ of the time $M^{R,\pi}(0^n;r)$ does not query the witness bits. If we change R slightly by flipping the first bit of $R(w\|e_{t,i})$ from 0 to 1, and denote the new oracle by R_i, then $0^n \notin L_{R_i}$. However, the machine $M^{R_i,\pi}(0^n;r)$ cannot detect the change in more than $\frac{2}{3}$ of the time. So $M^{R_i,\pi}(0^n;r)$ accepts with probability at least $\frac{1}{3}$. Moreover, $M^{R'_i,\pi}(0^n;r)$ makes the same mistake for any function R'_i that agrees with R_i on S_n. This conclusion contradicts the fact that M verifies $L_{R'_i}$ on 0^n for all such functions R'_i. So Equation (1) holds.

Furthermore, if $w\|e_{t,i}$ is the first query made by $M^{R,\pi}(0^n;r)$ such that $R(w\|e_{t,i}) = 01^{\ell(t+\lceil \log t \rceil)-1}$, then $M^{\mathbf{1},\pi}(0^n;r)$ would also make the query $w\|e_{t,i}$. This is because M has the same view in the two cases at the time it makes the query $w\|e_{t,i}$. The lemma follows. ◀

▶ **Lemma 34.** *If $M \in \mathbf{M}_{\mathrm{poly}(t),o(t)}$ is a PCP-verifier for $\mathcal{L}$, then for every $n \in \mathbb{N}$ there exists R agreeing with some $f \in X_n$ such that for every proof $\pi \in \{0,1\}^*$,*

$$\Pr_r\left[M^{\mathbf{1},\pi}(0^n;r) \text{ queries } \mathbf{1} \text{ at some } w\|e_{t,i} \text{ s.t. } R(w\|e_{t,i})_1 = 0\right] \in o(1) \ .$$

Proof. For the sake of contradiction, suppose there exists some $n \in \mathbb{N}$ such that for every R agreeing with some $f \in X_n$ there exists a proof π for which the above probability is $\Omega(1)$. Then, by averaging, there exists some randomness r^* such that, for a $\Omega(1)$-fraction of the functions R agreeing with some $f \in X_n$, there exists a proof π such that $M^{\mathbf{1},\pi}(0^n;r^*)$ queries some $w\|e_{t,i}$ s.t. $R(w\|e_{t,i})_1 = 0$. So across all possible proofs, $M^{\mathbf{1},\cdot}(0^n;r^*)$ need to make at least $\Omega(|X_n|) = \Omega(2^t)$ distinct queries.

However, since the randomness is fixed and M is in $\mathbf{M}_{\mathrm{poly}(t),o(t)}$, $M^{\mathbf{1},\cdot}(0^n;r^*)$ can only make at most $2^{o(t)} \cdot \mathrm{poly}(t)$ distinct queries, which leads to a contradiction because $\Omega(2^t) \gg 2^{o(t)} \cdot \mathrm{poly}(t)$. ◀

4.2 Proof of Part 2 of Theorem 31

We exhibit an oracle language $\mathcal{L}$ that is in $\mathrm{DTIME}(t)^{\mathcal{R}}$ but not in $\mathrm{PCP}(o(t),o(t))^{\mathcal{R}}$.

Oracle language. Let $u_{n,i}$ denote the n-bit string whose i-th bit is 1 and all other bits are zero. The oracle language $\mathcal{L} = \{L_R\}_{R\in\mathcal{R}}$ is defined as follows:

$$L_R := \left\{(x,y) \in \{0,1\}^n \times \{0,1\}^n \;\middle|\; F_R^{\frac{t(n)}{n}}(x) = y\right\} \ ,$$

where $F_R(x) := R(x \oplus u_{n,1})_1 \| R(x \oplus u_{n,2})_1 \| \ldots \| R(x \oplus u_{n,n})_1$.

In DTIME. We argue that $\mathcal{L}$ is in $\mathrm{DTIME}(t)^{\mathcal{R}}$. Consider the deterministic machine that on input (x, y): (a) copies $x_0 := x$ to the query tape; (b) for $j \in \{1, \ldots, t(n)/n\}$, calls R on inputs $\{x_{j-1} \oplus u_{n,i}\}_{i \in [n]}$ to get $x_j := F_R(x_{j-1})$, and copies x_j to the query tape; (c) accepts if $y = x_{t(n)/n}$. Each of the $t(n)/n$ iterations takes time $O(n)$, so the running time of the machine is $O(t(n))$.

Not in PCP. We argue that $\mathcal{L}$ is not in $\mathrm{PCP}(o(t), o(t))^{\mathcal{R}}$. We begin with a combinatorial claim and some notation.

- Recall that the entropy function $H\colon [0,1] \to [0,1]$ is $H(z) := -z\log_2(z) - (1-z)\log_2(1-z)$. For every $\epsilon \in \left(0, \frac{1}{2}\right)$ and $n \in \mathbb{N}$, there exist a list $C_{\epsilon,n} = \{a_0, a_1, \ldots, a_k\}$ of $\Omega\left(\sqrt{n}2^{(1-H(\epsilon))n}\right)$ distinct n-bit binary strings such that the relative hamming distance between any two distinct strings in $C_{\epsilon,n}$ is at least ϵn. These binary strings are the codewords obtained by the greedy approach of constructing a code in $\{0,1\}^n$ with minimum distance ϵn.
- Choose $\epsilon^* \in \left(0, \frac{1}{2}\right)$ such that for all large enough $n \in \mathbb{N}$ it holds that $|C_{\epsilon^*,n}| > \frac{t(n)}{n}$. Define the domain $S_n := \cup_{1 \le i \le t(n)}\{0,1\}^i$ and a set of functions X_n on S_n:

$$X_n := \{g\colon S_n \to \{0,1\}^* \mid \forall\, a_i \in C_{\epsilon^*,n}, a_{i+1} = g(a_i \oplus u_{n,1})_1 \| \cdots \| g(a_i \oplus u_{n,n})_1\} \quad .$$

  Therefore for every $g \in X_n$ and $a_i \in C_{\epsilon^*,n}$ it holds that $F_g(a_i) = a_{i+1}$, and $F_g^{\frac{t(n)}{n}}(a_0) = a_{t(n)/n}$.
- For every $i \in [k]$ and $g \in X_n$, define the function $g^{(i)}\colon S_n \to \{0,1\}^*$ to be

$$g^{(i)}(z) := \begin{cases} (a_0)_j \| 0^{\ell(|z|)-1} & \text{if } z = a_i \oplus u_{n,j} \text{ for some } j \in [n] \\ g(z) & \text{if } z \neq a_i \oplus u_{n,j} \text{ for every } j \in [n] \end{cases} \quad .$$

  Therefore for every $i' \neq i$ and $a_{i'} \in C_{\epsilon^*,n}$, $F_{g^{(i)}}(a_{i'}) = a_{i'+1}$, and $F_{g^{(i)}}(a_i) = a_0$.

We argue that any PCP-verifier M for $\mathcal{L}$ must have running time $\Omega(t(n))$.

- Since M is a PCP-verifier for $\mathcal{L}$, for every $n \in \mathbb{N}$ and every $g \in X_n$ there exists a proof π such that $M^{g,\pi}(a_0, a_{t(n)/n})$ accepts with probability at least $\frac{2}{3}$. Then, via Lemma 35 below, we deduce that for every n large enough there exists a randomness r such that $M^{g,\pi}(a_0, a_{t(n)/n}; r)$ makes at least $\frac{t(n)}{3n}$ queries of the form $F_g^i(a_0)$ for some $i \in [\frac{t(n)}{n}]$.
- Since the relative hamming distance between each pair of queries $F_g^i(a_0)$ and $F_g^j(a_0)$ is at least ϵn, the running time of $M^{g,\pi}(a_0, a_{t(n)/n}; r)$ is at least $\frac{t(n)}{3n} \cdot \epsilon n \in \Omega(t(n))$.

We have shown that any PCP-verifier for $\mathcal{L}$ has running time in $\Omega(t(n))$, and so $\mathcal{L} \notin \mathrm{PCP}(o(t), o(t))^{\mathcal{R}}$.

▶ **Lemma 35.** *If M is a PCP-verifier for $\mathcal{L}$ then, for every large enough $n \in \mathbb{N}$ and every $g \in X_{\epsilon^*,n}$, there exists a proof π and randomness r such that $M^{g,\pi}(a_0, a_{t(n)/n}; r)$ makes at least $\frac{t(n)}{3n}$ queries of the form $F_g^i(a_0)$ for some $i \in [\frac{t(n)}{n}]$.*

Proof. Since M is a PCP-verifier for $\mathcal{L}$, for large enough n, $(a_0, a_{t(n)/n}) \in L_R$ for every R that agrees with some $g \in X_n$. So for every $g \in X_n$ there exists a proof π such that $M^{g,\pi}(a_0, a_{t(n)/n})$ accepts with probability at least $\frac{2}{3}$. If we change one assignment of g to obtain $g^{(i)}$, then $F_{g^{(i)}}^{t(n)/n}(a_0) \neq a_{t(n)/n}$. So $M^{g^{(i)},\pi}(a_0, a_{t(n)/n})$ should accept with probability at most $\frac{1}{3}$. This implies that for at least $\frac{1}{3}$ fraction of randomness r, $M^{g,\pi}(a_0, a_{t(n)/n}; r)$ queries a_i. By averaging, there exists r^* such that $M^{g,\pi}(a_0, a_{t(n)/n}; r^*)$ makes $\frac{t(n)}{3n}$ distinct queries of the form $F_g^i(a_0)$. ◀

5 Separation for random low-degree functions

We define the notion of a random *low-degree* oracle and then state our separation result for it.

▶ **Definition 36.** *Let q be a prime power, $\mathbb{F}_q$ the finite field of size q, and $d \in \mathbb{N}$ a degree bound. The* **random oracle over $\mathbb{F}_q$ with degree d** *is the oracle $\mathcal{P}[q,d] = \{\mathcal{P}_n\}_{n\in\mathbb{N}}$ where each $\mathcal{P}_n$ is the uniform distribution over polynomials $f\colon \mathbb{F}_q^n \to \mathbb{F}_q$ of degree at most d in each variable. (In particular, $\mathcal{P}[2,1]$ equals the random oracle $\mathcal{R}$ from Definition 30.)*

▶ **Theorem 37.** *For any function $t\colon \mathbb{N} \to \mathbb{N}$ with $t(n) \in \Omega(n)$, prime power $q \in \mathbb{N}$, and degree bound $d \in \mathbb{N}$*

$$\mathrm{NTIME}(t)^{\mathcal{P}[q,d]} \not\subseteq \mathrm{PCP}(\mathrm{poly}(t), o(t))^{\mathcal{P}[q,d]} \ .$$

Proof. We first focus on the special case of $d = 1$, i.e., the case of random multilinear polynomials. We show in Lemma 38 that $\mathcal{P}[q,1]$ separates NTIME and PCP. Next, we simply observe that for any field $\mathbb{F}_q$ and degree d, the low degree random oracle $\mathcal{P}[q,d]$ contains the multilinear random oracle $\mathcal{P}[q,1]$. Therefore, by Lemma 28, $\mathcal{P}[q,d]$ also separates NTIME and PCP. ◀

Now we prove the separation for the special case of $d = 1$.

▶ **Lemma 38.** *For any function $t\colon \mathbb{N} \to \mathbb{N}$ with $t(n) \in \Omega(n)$, and any prime power $q \in \mathbb{N}$,*

$$\mathrm{NTIME}(t)^{\mathcal{P}[q,1]} \not\subseteq \mathrm{PCP}(\mathrm{poly}(t), o(t))^{\mathcal{P}[q,1]} \ .$$

Proof. We exhibit $\mathcal{L}$ that is in $\mathrm{NTIME}(t)^{\mathcal{P}[q,1]}$ but not in $\mathrm{PCP}(\mathrm{poly}(t), o(t))^{\mathcal{P}[q,1]}$.

Let $e_{k,i}$ denote the vector in $\mathbb{F}_q^k$ that has 1 in the i-th coordinate and 0 everywhere else. Let $\mathcal{L} = \{L_P\}_{P\in\mathcal{P}[q,1]}$ be the oracle language where

$$L_P := \left\{ 0^n \;\middle|\; \exists\, x \in \mathbb{F}_q^{t(n)} \text{ s.t. } P(x) = 1 \text{ and } \begin{array}{c} P(x + e_{t(n),1}) = 0 \\ P(x + e_{t(n),2}) = 0 \\ \vdots \\ P(x + e_{t(n),t(n)}) = 0 \end{array} \right\} .$$

It's clear that L_P is in $\mathrm{NTIME}(t)^P$ for every $P \in \mathcal{P}[q,1]$. Let $q\colon \mathbb{N} \to \mathbb{N}$ be some superpolynomial function. For every $n \in \mathbb{N}$, we define the following set of functions over the domain $S_n = \{\mathbb{F}_q^i \mid i \le q(t(n))\}$.

$$X_n = \left\{ f\colon S_n \to \mathbb{F}_q \;\middle|\; \begin{array}{l} \forall x \in \mathbb{F}_q^{t(n)},\ f(x) = \prod_{i=1}^{t(n)} (b_i - x_i) \text{ where } b \in \mathbb{F}_q^{t(n)} \\ \forall x \in S_n \setminus \mathbb{F}_q^{t(n)},\ f(x) = 0 \end{array} \right\} .$$

Note for every $n \in \mathbb{N}$, every function $f \in X_n$ has its measure $\mu_{\mathcal{P}[q,1]}(\{f\}) > 0$.

We also use $\mathbf{0}$ to denote the all zero function.

Suppose by way of contradiction that $\mathcal{L}$ has a PCP-verifier $M \in \mathbf{M}_t$. Use $T(n)$ to denote the running time of M on input of size n. Note that $T(n) \in \mathrm{poly}(t(n))$, so there exists a number $n^* \in \mathbb{N}$ such that $\forall n \ge n^*, q(t(n)) \ge T(n)$. We derive a contradiction from the following two steps. First in Claim 39, we show that for every $n \ge n^*$ and oracle polynomial P agreeing with some $f \in X_n$ there exists some proof π such that the PCP-verifier $M^{\mathbf{0},\pi}$, which has oracle access to $\mathbf{0}$ and π, queries $\mathbf{0}$ at some x satisfying $P(x) \neq 0$ with probability at least $\frac{1}{3}$. Next, in Claim 40, we show that for every $n \ge n^*$ there exists some oracle polynomial

P agreeing with some $f \in X_n$ such that for any proof $\pi \in \{0,1\}^*$, the PCP-verifier $M^{\mathbf{0},\pi}$, which has oracle access to $\mathbf{0}$ and π, queries P at some x satisfying $P(x) \neq 0$ with probability only $o(1)$. These two statements are in contradiction. Thus $\mathcal{L}$ does not have a PCP-verifier. Therefore $\mathcal{P}[q,1]$ separates NTIME and PCP. ◀

In the proofs of the two lemmas we abbreviate $t(n)$ with t whenever the choice of n is clear from the context.

▷ Claim 39. If $M \in \mathbf{M}_t$ is the PCP-verifier for $\mathcal{L}$, then for every $n \geq n^*$ and oracle polynomial P agreeing with some $f \in X_n$, there exists π s.t.

$$\Pr_r[M^{\mathbf{0},\pi}(0^n;r) \text{ queries } \mathbf{0} \text{ at some } y \in \mathbb{F}_q^{t(n)} \text{ s.t. } P(y) \neq 0] > \frac{1}{3} \ .$$

Proof. We first observe that for every function $f(x) = \prod_{i=1}^{t(n)}(b_i - x_i)$ in X_n, the element $y = (b_1 - 1) \| \ldots \| (b_{t(n)} - 1) \in \mathbb{F}_q^{t(n)}$ satisfies $f(y) = 1$ and $f(y + e_{t(n),i}) = 0$ for every $i \in [t(n)]$. Therefore for every P agreeing with f over $\mathbb{F}_q^{t(n)}$, $0^n \in L_P$. As a result, for every such P, there exists some proof $\pi \in \{0,1\}^*$ such that $M^{P,\pi}(0^n)$ accepts with probability at least $\frac{2}{3}$. We also note that since $M^{P,\pi}(0^n)$ has running time $T(n) \leq q(t(n))$, M cannot make oracle queries outside the set S_n.

For every P agreeing with some $f \in X_n$, use π_P to denote the accepting proof for P. We additionally note that for every oracle P' agreeing with $\mathbf{0}$ over S_n, it holds that $0^n \notin L_{P'}$. So for any π_P, $M^{P',\pi_P}(0^n)$ accepts with probability at most $\frac{1}{3}$.

This implies that

$$\Pr_r[M^{P,\pi_P}(0^n;r) \text{ queries } P \text{ at } x \text{ s.t. } P(x) \neq \mathbf{0}(x) = 0] \geq \frac{1}{3} \ . \tag{2}$$

Let y be the first oracle query made by $M^{P,\pi_P}(0^n;r)$ such that $P(y) \neq \mathbf{0}(y) = 0$. If we replace P with $\mathbf{0}$, $M^{\mathbf{0},\pi_P}(0^n;r)$ would still make the oracle query y. We deduce that

$$\Pr_r[M^{\mathbf{0},\pi_P}(0^n;r) \text{ queries } P_0 \text{ at } x \text{ s.t. } P(x) \neq \mathbf{0}(x) = 0] \geq \frac{1}{3} \ . \quad \triangleleft$$

▷ Claim 40. If $M \in \mathbf{M}_t$ is the PCP-verifier for $\mathcal{L}$, there exists P agreeing with some $f \in X_n$ s.t. for all $\pi \in \{0,1\}^*$,

$$\Pr_r[M^{\mathbf{0},\pi}(0^n;r) \text{ queries } \mathbf{0} \text{ at } x \text{ s.t. } P(x) \neq 0] \in o(1) \ .$$

Proof. Suppose for every P agreeing with some $f \in X_n$ there exists a proof π that the aforementioned probability is $\Omega(1)$. Then, by an averaging argument, there exists some randomness r^* such that for $\Omega(1)$ fraction of oracles P agreeing with some $f \in X_n$, there exists π s.t. $M^{\mathbf{0},\pi}(0^n;r^*)$ queries some x s.t. $P(x) \neq 0$. Additionally, for any $x \in \mathbb{F}_q^{t(n)}$, there are exactly $(q-1)^{t(n)}$ multilinear functions $f \in X_n$ such that $f(x) \neq 0$. So across all possible proofs, $M^{\mathbf{0},\cdot}(0^n;r^*)$ need to make at least $\Omega(|X_n| / (q-1)^t \text{poly}(t)) = \Omega((q/(q-1))^t) \in \Omega(\exp(t))$ distinct queries.

However, since the randomness is fixed and M is in $\mathbf{M}_t$, $M^{\mathbf{0},\cdot}(0^n;r^*)$ can make at most $2^{o((t)} \cdot \text{poly}(t)$ distinct queries. However, $\Omega(\exp(t)) \gg 2^{o(t)} \cdot \text{poly}(t)$, so we derive a contradiction. $\triangleleft$

6 Separation for random generic groups

We present the definition of the generic group model (GGM) [38, 42, 21, 35, 36].

▶ **Definition 41** (groups and their representations). *An abelian group of order p is a pair $\mathbb{G} = (S, +)$ where S is a set of size p and $+ \colon S \times S \to S$ is a function that satisfies the axioms of a group operation. We denote by $\mathbf{0}$ the identity of $\mathbb{G}$. A representation of $\mathbb{G}$ is an injective function $\sigma \colon S \to \{0,1\}^{\lceil \log_2 p \rceil}$, and its inverse $\sigma^{-1} \colon \{0,1\}^{\lceil \log_2 p \rceil} \to S$ maps each image $\sigma(g) \in \{0,1\}^{\lceil \log_2 p \rceil}$ to its pre-image $g \in S$ and each string $s \in \{0,1\}^{\lceil \log_2 p \rceil} \setminus \sigma(S)$ to the identity $\mathbf{0} \in S$.*

▶ **Definition 42** (group oracles). *Let $\mathbb{G}$ be a group of order p, and σ a representation of $\mathbb{G}$. The group oracle corresponding to $(\mathbb{G}, \sigma)$ is the function $O \colon \{0,1\}^{4\lceil \log_2 p \rceil} \to \{0,1\}^{\lceil \log_2 p \rceil}$ such that $O(c_a, c_b, a, b) = \sigma(c_a \times \sigma^{-1}(a) + c_b \times \sigma^{-1}(b))$. To obtain the identity element of the group simply query $O(0^{\lceil \log_2 p \rceil}, 0^{\lceil \log_2 p \rceil}, a, b) = \sigma(\mathbf{0})$.*

▶ **Definition 43.** *The* **random group oracle** *is the oracle $\mathcal{O} = \{\mathcal{O}_p\}_{p \in \mathbb{N}}$ where each $\mathcal{O}_p$ is the uniform distribution over all group oracles for groups of size p. Namely, a sample from $\mathcal{O}_p$ is obtained as follows: sample a random group $\mathbb{G}$ of order p, sample a random representation σ of $\mathbb{G}$, and output the group oracle $O \colon \{0,1\}^{4\lceil \log_2 p \rceil} \to \{0,1\}^{\lceil \log_2 p \rceil}$ corresponding to $(\mathbb{G}, \sigma)$.*

▶ **Theorem 44.** *In the generic group model, $t \colon \mathbb{N} \to \mathbb{N}$ with $t(n) \in \Omega(n)$,*

$$\mathrm{NTIME}(t)^{\mathcal{O}} \not\subseteq \mathrm{PCP}(\mathrm{poly}(t), o(t))^{\mathcal{O}} \ .$$

Proof. We exhibit an oracle language $\mathcal{L}$ that is in $\mathrm{NTIME}(t)^{\mathcal{O}}$ but not in $\mathrm{PCP}(\mathrm{poly}(t), o(t))^{\mathcal{O}}$.

Oracle language.

Define $p(n)$ to be the largest prime number no larger than $2^{t(n)}$, so we know that $p(n) \in (2^{t(n)-1}, 2^{t(n)}]$. Let $\mathcal{L} = \{L_O\}_{O \in \mathcal{O}}$ be the oracle language where

$$L_O := \left\{ 0^n \;\middle|\; \exists\, x \in \{0,1\}^{t(n)}, x < p(n) - 1, \text{ s.t. } \sigma^{-1}(x) + \sigma^{-1}(x^{\oplus t(n)}) = \mathbf{0} \right\} \ ,$$

where $x^{\oplus t(n)}$ is the string identical to x everywhere except for the $t(n)$-th bit.

In NTIME. We note that numbers in $[p(n)]$ can be represented by binary strings of length $\Theta(t(n))$. So it's clear that L_O is in $\mathrm{NTIME}(t)^O$ for every $O \in \mathcal{O}$.

Not in PCP. We argue that $\mathcal{L}$ is not in $\mathrm{PCP}(\mathrm{poly}(t), o(t))^{\mathcal{O}}$. Consider an oracle O s.t. $0^n \notin L_O$. We show that for any PCP-verifier $M \in \mathbf{M}_t$, if M^O is correct on 0^n then we can construct another oracle O' for which $M^{O'}$ is not correct on 0^n (Lemma 45). Therefore there exists some O^* for which M^{O^*} is not correct on 0^n. Additionally, the language $\mathcal{L}$ is by definition $4t$-bounded and the running time of M is bounded by $2^{t(n)}$. Since M^{O^*} fails on 0^n, for every function F that agrees with O^* on $\bigcup_{1 \le i \le 2^{t(n)}} \{0,1\}^i$, M^F also fails on 0^n. So by Claim 26, we conclude that $\mathcal{L} \notin \mathrm{PCP}(\mathrm{poly}(t), o(t))^{\mathcal{O}}$. ◀

▶ **Lemma 45.** *For any PCP-verifier $M \in \mathbf{M}_t$ and any oracle $O \in \mathcal{O}$ such that $0^n \notin L_O$, if M^O is correct on 0^n then there exists O' for which $M^{O'}$ is not correct on 0^n.*

Proof. Use σ to denote O's representation of the order $p(n)$ group. Define the set of pairs of strings

$$I(O) := \{(u, v) \in (\{0,1\}^{t(n)})^2 \ \mid \ u, v < p(n) - 1,\ \sigma^{-1}(u) + \sigma^{-1}(v^{\oplus t(n)}) = \mathbf{0}\} \ .$$

Note that if we use $O^{(u,v)}$ to denote the oracle identical to O except for its permutation function for the group of order $p(n)$, which is defined as

$$\sigma^{(u,v)}(g) := \begin{cases} u & \text{if } \sigma(g) = v \\ v & \text{if } \sigma(g) = u \\ \sigma(g) & \text{otherwise} \end{cases} .$$

Note that for any $(u,v) \in I(O)$, we have $(\sigma^{(u,v)})^{-1}(v) + (\sigma^{(u,v)})^{-1}(v^{\oplus t(n)}) = \mathbf{0}$. So $0^n \in L_{O^{(u,v)}}$.

Consider the pairs of strings $(u,v) \in I(O)$ for which M^O queries all of (u,v) with "low" probability:

$$X_\star(O) := \left\{ (u,v) \in I(O) \;\middle|\; \text{for } i = u,v, \sum_{\pi \in \{0,1\}^*} \Pr_r \left[\begin{array}{l} M^{O,\pi}(0^n;r) \text{ queries } O \text{ at } i \\ \text{or gets back } i \text{ from } O \end{array} \right] < \frac{1}{6} \right\} .$$

We argue that $|X_\star(O)| > 0$, and for every $(u,v) \in X_\star(O)$ it holds that $M^{O^{(u,v)}}$ is *not* correct on 0^n.

Consider the set of strings $u \in \{0,1\}^{\lceil \log_2 p(n) \rceil}$, $u < p(n) - 1$ that M^O queries with "high" probability:

$$X_c(O) := \left\{ u \in \{0,1\}^{t(n)} \;\middle|\; u < p(n) - 1 \,, \sum_{\pi \in \{0,1\}^*} \Pr_r \left[\begin{array}{l} M^{O,\pi}(0^n;r) \text{ queries } O \text{ at } u \\ \text{or gets back } u \text{ from } O \end{array} \right] \geq \frac{1}{6} \right\} .$$

Observe that

$$|X_c(O)| \leq \Big(c \cdot t(n)\Big) \cdot \Big(2^{o(t(n))} \cdot \mathrm{poly}(t(n))\Big) = 2^{o(t(n))} \;. \tag{3}$$

This is because, in any given execution, M can make at most $\mathrm{poly}(t(n))$ queries to O (also can get at most $\mathrm{poly}(t(n))$ symbols from O) and $o(t(n))$ queries to the given proof string, which means that

$$\sum_{u \in \{0,1\}^{t(n)},\, u < p(n)-1} \; \sum_{\pi \in \{0,1\}^*} \Pr_r \left[\begin{array}{l} M^{O,\pi}(0^n;r) \text{ queries } O \text{ at } u \\ \text{or gets back } u \text{ from } O \end{array} \right] \leq 2^{o(t(n))} \mathrm{poly}(t(n)) \;.$$

We deduce that $|X_\star(O)|$ is large:

$$|X_\star(O)| \geq (p(n) - 3) - 2 \cdot 2|X_c(O)| = 2^{t(n)} - 2^{o(t(n))} \;.$$

Next, for every $(u,v) \in X_\star(O)$ it holds that

$$\forall\, \pi \in \{0,1\}^*, \text{ for } i = u,v, \Pr_r \left[\begin{array}{l} M^{O,\pi}(0^n;r) \text{ queries } O \text{ at } i \\ \text{or gets back } i \text{ from } O \end{array} \right] < \frac{1}{6} \;.$$

Therefore,

$$\begin{aligned} \forall\, \pi \in \{0,1\}^*, \quad & \Pr_r[M^{O,\pi}(0^n;r) \text{ queries } O \text{ at } u \text{ or } v] \\ & \leq \sum_{i=u,v} \Pr_r \left[\begin{array}{l} M^{O,\pi}(0^n;r) \text{ queries } O \text{ at } i \\ \text{or gets back } i \text{ from } O \end{array} \right] \\ & < 2 \cdot \frac{1}{6} = \frac{1}{3} \;. \end{aligned}$$

This means that for every $(u,v) \in X_\star(O)$ it holds that M cannot distinguish between O and $O^{u,v}$ with probability greater than or equal to $\frac{1}{3}$.

We know that M^O is correct on 0^n, namely, for every proof string π it holds that $M^{O,\pi}(0^n)$ accepts with probability no more than $1/3$. We also know that for every $(u,v) \in I(O)$ it holds that 0^n is in $L_{O^{(u,v)}}$. But the foregoing argument tells us that for every $(u,v) \in X_\star(O)$ it holds that for every proof string π we have that $M^{O^{(u,v)},\pi}(0^n)$ accepts with probability less than $1/3 + 1/3 = 2/3$. We deduce that, for every $(u,v) \in X_\star(O)$, $M^{O^{(u,v)}}$ is *not* correct on 0^n. ◀

7 Almost-everywhere separation for random functions

We strengthen the separation for random functions in Theorem 31. The difference is that now the choice of machine M, which is the candidate PCP-verifier for L_R, *depends* on the sample R.

▶ **Theorem 46.** *Let $\mathcal{R}$ be a random oracle with output length $\ell\colon \mathbb{N} \to \mathbb{N}$.*

1. *For any function $t\colon \mathbb{N} \to \mathbb{N}$ with $t(n) \in \Omega(n)$,*

$$\Pr_{R \leftarrow \mathcal{R}}\left[\mathrm{NTIME}(t)^R \not\subseteq \mathrm{PCP}(\mathrm{poly}(t), o(t))^R\right] = 1 \ .$$

2. *For any function $t\colon \mathbb{N} \to \mathbb{N}$ with $t(n) \in \Omega(n) \bigcap o(2^n)$,*

$$\Pr_{R \leftarrow \mathcal{R}}\left[\mathrm{DTIME}(t)^R \not\subseteq \mathrm{PCP}(o(t), o(t))^R\right] = 1 \ .$$

7.1 Proof of Part 1 of Theorem 46

We prove the statement by arguing that, for every oracle R in a certain set of measure 1 (derived below), there exists a language L_R that is in $\mathrm{NTIME}(t)^R$ but not in $\mathrm{PCP}(\mathrm{poly}(t), o(t))^R$.

We first define the language $L_R \subseteq \{0,1\}^*$ for any $R \in \mathcal{R}$. The language is defined as follows:

$$L_R = \left\{ 0^n \;\middle|\; \exists\, w \in \{0,1\}^{t(n)} \text{ s.t. } \begin{array}{l} R(w \| e_{t(n),1})_1 = 0 \\ R(w \| e_{t(n),2})_1 = 0 \\ \quad \vdots \\ R(w \| e_{t(n),t(n)})_1 = 0 \end{array} \right\} \ .$$

The language L_R is in $\mathrm{NTIME}(t)^R$ for every $R \in \mathcal{R}$ (via the same argument as in Section 4.1). We are left to argue that L_R is not in $\mathrm{PCP}(\mathrm{poly}(t), o(t))^R$ for R in a certain set of measure 1. For this, we state a lemma (which we prove later on below), and then conclude the proof of the theorem.

▶ **Lemma 47.** *For every $M \in \mathbf{M}_{\mathrm{poly}(t),o(t)}$, $\Pr_{R \leftarrow \mathcal{R}}\left[M^R \text{ is a PCP-verifier for } L_R\right] = 0$.*

Let S_M be the set of oracles $R \in \mathcal{R}$ for which M^R is a PCP-verifier for L_R. Lemma 47 tells us that S_M has measure zero, that is, $\mu_{\mathcal{R}}(S_M) = 0$. Since the set $\mathbf{M}_{\mathrm{poly}(t),o(t)}$ is countable (it is a subset of the countable set of all machines) and measures are countably sub-additive, we deduce that

$$\mu_{\mathcal{R}}\left(\bigcup_{M \in \mathbf{M}_{\mathrm{poly}(t),o(t)}} S_M\right) \leq \sum_{M \in \mathbf{M}_{\mathrm{poly}(t),o(t)}} \mu_{\mathcal{R}}(S_M) = 0 \ .$$

We conclude that

$$\Pr_{R \leftarrow \mathcal{R}} \left[\exists\, M \in \mathbf{M}_{\mathrm{poly}(t),o(t)} \text{ s.t. } M^R \text{ is a PCP-verifier for } L_R \right] = 0 \ ,$$

which shows that L_R is not in $\mathrm{PCP}(\mathrm{poly}(t), o(t))^R$ for all R in a set of measure 1.

This completes the proof, and so we are only left with proving Lemma 47.

Before proving Lemma 47, we define two disjoint sets of oracles, $S_{n,0}$ and $S_{n,1}$, and then prove certain properties about them (see Lemmas 50 to 52 below).

▶ **Definition 48.** *For every $n \in \mathbb{N}$, function $R \in \mathcal{R}_n$, and string $w \in \{0,1\}^{t(n)}$, we define the function $\mathbf{F}[R,w]\colon \{0,1\}^* \to \{0,1\}^*$ to be*

$$\mathbf{F}[R,w](z)_j := \begin{cases} 0 & \text{if } j = 1 \text{ and } z = w \| e_{t(n),i} \text{ for some } i \in [t(n)] \\ R(z)_j & \text{otherwise} \end{cases} .$$

Moreover, given $S \subseteq \mathcal{R}_n$, we define $\mathbf{F}[S, \{0,1\}^{t(n)}]$ to be the set $\{\mathbf{F}[R,w] \,|\, R \in S\,,\, w \in \{0,1\}^{t(n)}\}$.

▶ **Definition 49.** *For every $n \in \mathbb{N}$, $S_{n,0}$ is the set of functions $R \in \mathcal{R}_n$ for which $0^n \notin L_R$, that is, for which for every $w \in \{0,1\}^{t(n)}$ there exists an index $i \in [t(n)]$ such that $R(w \| e_{t(n),i})_1 \neq 0$. Also, $S_{n,1}$ equals the set $\mathbf{F}[S_{n,0}, \{0,1\}^{t(n)}]$, which is disjoint from $S_{n,0}$ (since $0^n \in L_R$ for every $R \in S_{n,1}$).*

▶ **Lemma 50.** *For every subset $S \subseteq S_{n,0}$, we have $\mu_{\mathcal{R}}(S) \leq \mu_{\mathcal{R}}(\mathbf{F}[S, \{0,1\}^{t(n)}])$.*

Proof. Each $R \in S$ yields $2^{t(n)}$ distinct functions $\mathbf{F}[R,w]$ as w ranges over $\{0,1\}^{t(n)}$. On the other hand, each $R' \in \mathbf{F}[S, \{0,1\}^{t(n)}]$ has at most $2^{t(n)} - 1$ "pre-images" in S: there exists precisely one w such that $R(w \| e_{t(n),i})_1 = 0$ for all $i \in \{1, \ldots, t(n)\}$. So if $R' = \mathbf{F}[R,w]$, R and R' can only be different in the first bit at locations of the form $w \| e_{t(n),i}$ for $i \in \{1, \ldots, t(n)\}$. There are $2^{t(n)} - 1$ different assignments to the first bits at $w \| e_{t(n),i}$ each of which gives rise to a preimage of R' in S (we exclude the all-zero assignment). We deduce that $2^{t(n)} \mu_{\mathcal{R}}(S) \leq \left(2^{t(n)} - 1\right) \mu_{\mathcal{R}}(\mathbf{F}[S, \{0,1\}^{t(n)}])$, and so $\mu_{\mathcal{R}}(S) \leq \mu_{\mathcal{R}}(\mathbf{F}[S, \{0,1\}^{t(n)}])$. ◀

▶ **Lemma 51.** $\lim_{n \to \infty} \mu_{\mathcal{R}}(S_{n,0}) = 1/e$.

Proof. For any $n \in \mathbb{N}$ the measure of $S_{n,0}$ in $\mathcal{R}$ is $\mu_{\mathcal{R}}(S_{n,0}) = (1 - \frac{1}{2^{t(n)}})^{2^{t(n)}}$. Therefore:

$$\lim_{n \to \infty} \mu_{\mathcal{R}}(S_{n,0}) = \lim_{N \to \infty} \left(1 - \frac{1}{N}\right)^N = \lim_{N \to \infty} e^{N\left(1 - \frac{1}{N}\right)} = \lim_{N \to \infty} e^{\left(1 - \frac{1}{N}\right)' / \left(\frac{1}{N}\right)'} = 1/e \ . \quad \blacktriangleleft$$

▶ **Lemma 52.** *For every function $R \in S_{n,0}$, if M^R is correct on 0^n then there are at least $2^{t(n)} - 2^{o(t(n))}$ strings $w \in \{0,1\}^{t(n)}$ for which $M^{\mathbf{F}[R,w]}$ is* not *correct on 0^n. (Note that $\mathbf{F}[R,w] \in S_{n,1}$.)*

Proof. Fix a constant $c > 0$ to be determined later. Consider the set of strings $w \in \{0,1\}^{t(n)}$ for which M^R queries all of $\{w \oplus e_{t(n),1}, \ldots, w \oplus e_{t(n),t(n)}\}$ with "low" probability:

$$X_\star(R) := \left\{ w \in \{0,1\}^{t(n)} \;\middle|\; \forall\, i \in [t(n)]\,,\ \sum_{\pi \in \{0,1\}^*} \Pr_r[M^{R,\pi}(0^n; r) \text{ queries } R \text{ at } w \oplus e_{t(n),i}] < \frac{1}{c \cdot t(n)} \right\} .$$

We argue that $|X_\star(R)|$ is large, and for every $w \in X_\star(R)$ it holds that $M^{\mathbf{F}[R,w]}$ is *not* correct on 0^n.

Consider the set of strings $w \in \{0,1\}^{t(n)}$ that M^R queries with "high" probability:

$$X_c(R) := \left\{ w \in \{0,1\}^{t(n)} \;\middle|\; \sum_{\pi \in \{0,1\}^*} \Pr_r[M^{R,\pi}(0^n;r) \text{ queries } R \text{ at } w\,] \geq \frac{1}{c \cdot t(n)} \right\} .$$

Observe that

$$|X_c(R)| \leq \Big(c \cdot t(n)\Big) \cdot \Big(2^{o(t(n))} \cdot \mathrm{poly}(t(n))\Big) = 2^{o(t(n))} . \tag{4}$$

This is because, in any given execution, M can make at most $\mathrm{poly}(t(n))$ queries to R and $o(t(n))$ queries to the given proof string, which means that

$$\sum_{w \in \{0,1\}^{t(n)}} \sum_{\pi \in \{0,1\}^*} \Pr_r[M^{R,\pi}(0^n;r) \text{ queries } R \text{ at } w] \leq 2^{o(t(n))}\mathrm{poly}(t(n)) .$$

We deduce, via Equation (4), that $|X_\star(R)|$ is large:

$$|X_\star(R)| \geq 2^{t(n)} - t(n)|X_c(R)| = 2^{t(n)} - 2^{o(t(n))} .$$

Next, for every $w \in X_\star(R)$ it holds that

$$\forall\, \pi \in \{0,1\}^*,\ \forall\, i \in [t(n)],\ \Pr_r[M^{R,\pi}(0^n;r) \text{ queries } R \text{ at } w \oplus e_{t(n),i}] < \frac{1}{c \cdot t(n)} .$$

Therefore,

$$\begin{aligned} \forall\, \pi \in \{0,1\}^*, \quad & \Pr_r[M^{R,\pi}(0^n;r) \text{ queries } R \text{ at any } w \oplus e_{t(n),i}] \\ & \leq \sum_{i \in [t(n)]} \Pr_r[M^{R,\pi}(0^n;r) \text{ queries } R \text{ at } w \oplus e_{t(n),i}] \\ & < t(n) \cdot \frac{1}{c \cdot t(n)} = \frac{1}{c} . \end{aligned}$$

This means that for every $w \in X_\star(R)$ it holds that M cannot distinguish between R and $R' := \mathbf{F}[R,w]$ with probability greater than $\frac{1}{c}$ (the probability is over M's randomness r).

We know that M^R is correct on 0^n, namely, for every proof string π it holds that $M^{R,\pi}(0^n)$ accepts with probability less than $1/3$. We also know that for every $w \in \{0,1\}^{t(n)}$ it holds that 0^n is in $L_{\mathbf{F}[R,w]}$. But the foregoing argument tells us that for every $w \in X_\star(R)$ it holds that for every proof string π we have that $M^{\mathbf{F}[R,w],\pi}(0^n)$ accepts with probability less than $1/3 + 1/c$.

Choosing $c \geq 3$, we deduce that, for every $w \in X_\star(R)$, $M^{\mathbf{F}[R,w]}$ is *not* correct on 0^n. ◀

Proof of Lemma 47. Fix $M \in \mathbf{M}_{\mathrm{poly}(t),o(t)}$. It suffices to show that, for some $\epsilon \in [0,1)$ and every $n \in \mathbb{N}$, M^R is correct on input 0^n for at most an ϵ-fraction of oracles R. Indeed, this fact would imply that:

$$\begin{aligned} & \Pr_{R \leftarrow \mathcal{R}}\left[\, M^R \text{ is a PCP-verifier for } L_R \,\right] \\ & \leq \Pr_{R \leftarrow \mathcal{R}}\left[\, \forall n \in \mathbb{N},\ M^R \text{ is correct on } 0^n \,\right] \\ & = \prod_{n \in \mathbb{N}} \Pr_{R \leftarrow \mathcal{R}}\left[\, M^R \text{ is correct on } 0^n \;\middle|\; M^R \text{ is correct on } 0^i \text{ for all } i < n \,\right] \\ & = \prod_{n \in \mathbb{N}} \Pr_{R \leftarrow \mathcal{R}}\left[\, M^R \text{ is correct on } 0^n \,\right] \\ & \leq \lim_{n \to \infty} \epsilon^n = 0 , \end{aligned}$$

as claimed. Above "correct" on input 0^n means that if $0^n \in L_R$ then there exists a proof string π such that $M^{R,\pi}(0^n)$ accepts with probability at least $2/3$, and if $0^n \notin L_R$ then for every proof string π it holds that $M^{R,\pi}(0^n)$ rejects with probability at least $2/3$.

We are left to argue that M^R is correct on input 0^n for at most an ϵ-fraction of oracles R. Consider the following sets of oracles:

$$U_{n,\mathsf{all}} := \left\{ R \in \mathcal{R} \,|\, M^R \text{ is incorrect on } 0^n \right\} ,$$
$$U_{n,0} := \left\{ R \in S_{n,0} \,|\, M^R \text{ is incorrect on } 0^n \right\} ,$$
$$U_{n,1} := \left\{ R' \in S_{n,1} \,|\, M^{R'} \text{ is incorrect on } 0^n \right\} .$$

Note that $U_{n,0} \cup U_{n,1} \subseteq U_{n,\mathsf{all}}$. Also, $U_{n,0}$ and $U_{n,1}$ are disjoint, because $S_{n,0}$ and $S_{n,1}$ are disjoint.

We want to prove that $\mu_{\mathcal{R}}(U_{n,\mathsf{all}}) > 1 - \epsilon$. We do so as follows:

$$\begin{aligned}
\mu_{\mathcal{R}}(U_{n,\mathsf{all}}) &\geq \mu_{\mathcal{R}}(U_{n,0}) + \mu_{\mathcal{R}}(U_{n,1}) \\
&\geq \Big(\mu_{\mathcal{R}}(S_{n,0}) - \mu_{\mathcal{R}}(S_{n,0} \setminus U_{n,0})\Big) + \mu_{\mathcal{R}}(U_{n,1}) \\
&\geq_{[a]} \Big(\mu_{\mathcal{R}}(S_{n,0}) - \mu_{\mathcal{R}}(S_{n,0} \setminus U_{n,0})\Big) + \frac{2^{t(n)} - 2^{o(t(n))}}{2^{t(n)}} \cdot \mu_{\mathcal{R}}\left(\mathbf{F}[S_{n,0} \setminus U_{n,0}, \{0,1\}^{t(n)}]\right) \\
&\geq_{[b]} \Big(\mu_{\mathcal{R}}(S_{n,0}) - \mu_{\mathcal{R}}(S_{n,0} \setminus U_{n,0})\Big) + \frac{2^{t(n)} - 2^{o(t(n))}}{2^{t(n)}} \cdot \mu_{\mathcal{R}}(S_{n,0} \setminus U_{n,0}) \\
&= \mu_{\mathcal{R}}(S_{n,0}) - \frac{2^{o(t(n))}}{2^{t(n)}} \cdot \mu_{\mathcal{R}}(S_{n,0} \setminus U_{n,0}) \\
&\geq \mu_{\mathcal{R}}(S_{n,0}) - \frac{2^{o(t(n))}}{2^{t(n)}} \cdot \mu_{\mathcal{R}}(S_{n,0}) \\
&= \left(1 - \frac{2^{o(t(n))}}{2^{t(n)}}\right) \cdot \mu_{\mathcal{R}}(S_{n,0}) \ .
\end{aligned}$$

Above, the third inequality (labeled $[a]$) follows from Lemma 52; the fourth inequality (labeled $[b]$) follows from Lemma 50 applied to the set $S_{n,0} \setminus U_{n,0}$.

Finally, by Lemma 51 we know that $\lim_{n\to\infty} \mu_{\mathcal{R}}(S_{n,0}) = 1/e > 1/3$, so if we set $\epsilon := 2/3$ then the above expression is greater than $1 - \epsilon$ for large enough n. ◀

7.2 Proof of Part 2 of Theorem 46

We prove the statement by arguing that, for every oracle R in a certain set of measure 1 (derived below), there exists a language L_R that is in $\mathrm{DTIME}(t)^R$ but not in $\mathrm{PCP}(o(t), o(t))^R$.

We first define the language $L_R \subseteq \{0,1\}^*$ for any $R \in \mathcal{R}$. The language is defined as follows:

$$L_R := \{(x,y) \in \{0,1\}^n \times \{0,1\}^n \,|\, F_{R,n}(x) = y\} \ ,$$

where $F_{R,n}(x)_i := \bigoplus_{j \in \left\{(i-1)\frac{t(n)}{n}+1,\ldots,i\frac{t(n)}{n}\right\}} R(x \| e_{t(n),j})_1$ for $i \in \{1,\ldots,n\}$.

We argue that the language L_R is in $\mathrm{DTIME}(t)^R$ for every $R \in \mathcal{R}$. Consider the deterministic machine that on input (x,y): (a) copies $x \| e_{t(n),1}$ to the query tape; (b) for $i \in \{1,\ldots,n\}$, calls R on inputs $\{x \| e_{t(n),j}\}_{j \in \left\{(i-1)\frac{t(n)}{n}+1,\ldots,i\frac{t(n)}{n}\right\}}$ to get $z_i := F_{R,n}(x)_i$; (c) accepts if $y = z$. Writing down x takes time n, querying all bits of the form $x \| e_{t(n),j}$ takes $O(t(n))$ time, computing z and comparing it with y takes $O(t(n))$ time. So the running time of the machine is $O(t(n))$. We are left to argue that L_R is not in $\mathrm{PCP}(o(t), o(t))^R$ for R in a certain set of measure 1. For this, we state a lemma (which we prove later on below), and then conclude the proof of the theorem.

▶ **Lemma 53.** *For every* $M \in \mathbf{M}_{o(t),o(t)}$, $\Pr_{R\leftarrow\mathcal{R}}\left[M^R \text{ is a PCP-verifier for } L_R\right] = 0$.

Using the same arguement as in the proof of Theorem 46, we conclude that

$$\Pr_{R\leftarrow\mathcal{R}}\left[\exists\, M \in \mathbf{M}_{o(t),o(t)} \text{ s.t. } M^R \text{ is a PCP-verifier for } L_R\right] = 0 \ ,$$

which shows that L_R is not in $\mathrm{PCP}(o(t), o(t))^R$ for all R in a set of measure 1.

This completes the proof, and so we are only left with proving Lemma 53.

Before proving Lemma 53, we define for every $n \in \mathbb{N}$ 2^n disjoint sets of oracles, $S_{n,y}$ where $y \in \{0,1\}^n$, and then prove certain properties about them (see Remark 56 and Lemma 57 below).

▶ **Definition 54.** *For every* $n \in \mathbb{N}$ *and* $y \in \{0,1\}^n$, $S_{n,y}$ *is the set of functions* $R \in \mathcal{R}_n$ *for which* $(0^n, y) \in L_R$, *that is, for which* $F_{R,n}(0^n) = y$. *We note that by definition the sets are disjoint.*

▶ **Definition 55.** *For every* $n \in \mathbb{N}$, *function* $R \in \mathcal{R}_n$, *index* $i \in [n]$ *and coordinate index* $j \in \left[\frac{t(n)}{n}\right]$, *we define the function* $\mathbf{F}[R,i,j]\colon \{0,1\}^* \to \{0,1\}^*$ *to be*

$$\mathbf{F}[R,i,j](z)_k := \begin{cases} 1 - R(z)_k & \text{if } k = 1 \text{ and } z = 0^n \| e_{t(n),(i-1)\frac{t(n)}{n}+j} \\ R(z)_k & \text{otherwise} \end{cases} \ .$$

Moreover, given $S \subseteq \mathcal{R}_n$, *we define* $\mathbf{F}[S,i,j]$ *to be the set* $\{\mathbf{F}[R,i,j] \mid R \in S\}$.

From the definitions of the set $S_{n,y}$ and the map $\mathbf{F}[\ ,\ ,\ ]$, we immediately obtain the following claim.

▶ Remark 56. We note that for any $R \in S_{n,y}$, $i \in [n]$ and $j \in [t(n)/n]$, (a) $\mathbf{F}[R,i,j] \notin S_{n,y}$, since flipping the first bit at $0^n \| e_{t(n),(i-1)t(n)/n+j}$ results in $F_{R,n}(0^n)_i \neq F_{\mathbf{F}[R,i,j],n}(0^n)_i$; (b) the number of preimages of R under the maps $\{\mathbf{F}[\ ,i,j]\}_{i\in[n],j\in[t(n)/n]}$ is exactly $t(n)$.

▶ **Lemma 57.** *For every function* $R \in S_{n,y}$, *if* M^R *is correct on* $(0^n, y)$ *then there are at least* $t(n) - o(t(n))$ *pairs* $(i,j)_{i\in[n],j\in[t(n)/n]}$ *for which* $M^{\mathbf{F}[R,i,j]}$ *is* not *correct on* $(0^n, y)$.

Proof. Fix a constant $c > 0$. Let π be the proof such that $\Pr_r[M^{R,\pi}(0^n, y; r)] \geq \frac{2}{3}$. Consider the set of $t(n)$ queries

$$X(R) := \left\{0^n \| e_{t(n),(i-1)t(n)/n+j} \;\middle|\; i \in [n],\ j \in [t(n)/n]\right\} \ .$$

Define the subset of $X(R)$ which $M^{R,\pi}$ queries with "low" probability:

$$X_\star(R) := \left\{ w \in X(R) \;\middle|\; \Pr_r[M^{R,\pi}(0^n; r) \text{ queries } R \text{ at } w] < \frac{1}{c} \right\} \ .$$

We argue that $|X_\star(R)|$ is large, and for every $w \in X_\star(R)$, $w = 0^n \| e_{t(n),(i-1)t(n)/n+j}$ it holds that $M^{\mathbf{F}[R,i,j],\pi}$ is *not* correct on $(0^n, y)$.

Consider the set of strings $w \in X(R)$ that M^R queries with "high" probability:

$$X_c(R) := \left\{ w \in X(R) \;\middle|\; \Pr_r[M^{R,\pi}(0^n; r) \text{ queries } R \text{ at } w\ ] \geq \frac{1}{c} \right\} \ .$$

Observe that

$$|X_c(R)| \leq c \cdot o(t(n)) \in o(t(n)) \ . \tag{5}$$

This is because, in any given execution, M can make at most $o(t(n))$ queries to R which means that

$$\sum_{w \in X(R)} \Pr_r[M^{R,\pi}(0^n; r) \text{ queries } R \text{ at } w] \leq o(t(n)) \ .$$

We deduce, via Equation (5), that $|X_\star(R)|$ is large:

$$|X_\star(R)| \geq t(n) - |X_c(R)| = t(n) - o(t(n)) \ .$$

Next, for every $w \in X_\star(R)$ it holds that

$$\Pr_r[M^{R,\pi}(0^n; r) \text{ queries } R \text{ at } w] < \frac{1}{c} \ .$$

This means that for every $w \in X_\star(R)$, $w = 0^n \| e_{t(n),(i-1)t(n)/n+j}$, it holds that M cannot distinguish between R and $R' := \mathbf{F}[R, i, j]$ with probability greater than $\frac{1}{c}$ (the probability is over M's randomness r).

We know that $M^{R,\pi}$ is correct on $(0^n, y)$, namely, $M^{R,\pi}(0^n)$ accepts with probability at least $1/3$. We also know that for every $(i, j) \in [n] \times [t(n)/n]$ it holds that $(0^n, y) \notin L_{\mathbf{F}[R,i,j]}$ (part (a) of Remark 56). But the foregoing argument tells us that for every $0^n \| e_{t(n),(i-1)t(n)/n+j} \in X_\star(R)$ it holds that $M^{\mathbf{F}[R,i,j],\pi}(0^n)$ accepts with probability greater than $1/3 + 1/c$.

Choosing $c \geq 3$, we deduce that, for every $0^n \| e_{t(n),(i-1)t(n)/n+j} \in X_\star(R)$, $M^{\mathbf{F}[R,i,j]}$ is *not* correct on $(0^n, y)$. ◀

Proof of Lemma 53. Fix $M \in \mathbf{M}_{o(t),o(t)}$. It suffices to show that, for some $\epsilon \in [0, 1)$ and every $n \in \mathbb{N}$, M^R is correct on inputs $(0^n, y)_{y \in \{0,1\}^n}$ for at most an ϵ-fraction of oracles R. Indeed, this fact would imply that:

$$\begin{aligned}
&\Pr_{R \leftarrow \mathcal{R}} \left[M^R \text{ is a PCP-verifier for } L_R \right] \\
&\leq \Pr_{R \leftarrow \mathcal{R}} \left[\forall n \in \mathbb{N},\ M^R \text{ is correct on } (0^n, y)_{y \in \{0,1\}^n} \right] \\
&= \prod_{n \in \mathbb{N}} \Pr_{R \leftarrow \mathcal{R}} \left[M^R \text{ is correct on } (0^n, y)_{y \in \{0,1\}^n} \;\middle|\; M^R \text{ is correct on } (0^i, y)_{y \in \{0,1\}^i} \text{ for all } i < n \right] \\
&= \prod_{n \in \mathbb{N}} \Pr_{R \leftarrow \mathcal{R}} \left[M^R \text{ is correct on } (0^n, y)_{y \in \{0,1\}^n} \right] \\
&\leq \lim_{n \to \infty} \epsilon^n = 0 \ ,
\end{aligned}$$

as claimed. Above "correct" on input $(0^n, y)$ means that if $(0^n, y) \in L_R$ then there exists a proof string π such that $M^{R,\pi}(0^n, y)$ accepts with probability at least $2/3$, and if $(0^n, y) \notin L_R$ then for every proof string π it holds that $M^{R,\pi}(0^n, y)$ rejects with probability at least $2/3$.

We are left to argue that M^R is correct on inputs $(0^n, y)_{y \in \{0,1\}^n}$ for at most an ϵ-fraction of oracles R. Consider the following sets of oracles:

$$\begin{aligned}
U_{n,\mathsf{all}} &:= \left\{ R \in \mathcal{R} \,|\, M^R \text{ is incorrect on } (0^n, y) \text{ for some } y \in \{0,1\}^n \right\} \ , \\
U_{y,\mathsf{all}} &:= \left\{ R \in S_{n,y} \,|\, M^R \text{ is incorrect on } (0^n, y') \text{ for some } y' \in \{0,1\}^n \right\} \ , \\
U_{y,y} &:= \left\{ R \in S_{n,y} \,|\, M^R \text{ is incorrect on } (0^n, y) \right\} \ .
\end{aligned}$$

We want to prove that $\mu_{\mathcal{R}}(U_{n,\mathsf{all}}) > 1 - \epsilon = 1/3$. We do so as follows:

$$\begin{aligned}
\mu_{\mathcal{R}}(U_{n,\mathsf{all}}) &= \sum_{y\in\{0,1\}^n} \mu_{\mathcal{R}}(U_{y,\mathsf{all}}) \\
&\geq_{[a]} \frac{1}{t(n)} \cdot \sum_{y\in\{0,1\}^n} \mu_{\mathcal{R}}(S_{n,y} \setminus U_{y,y}) \cdot (t(n) - o(t(n))) \\
&\geq \sum_{y\in\{0,1\}^n} \mu_{\mathcal{R}}(S_{n,y} \setminus U_{y,\mathsf{all}}) \cdot \frac{t(n) - o(t(n))}{t(n)} \\
&= (1 - \mu_{\mathcal{R}}(U_{n,\mathsf{all}})) \cdot \frac{t(n) - o(t(n))}{t(n)} \\
&\geq \frac{t(n) - o(t(n))}{2t(n)} > \frac{1}{3}
\end{aligned}$$

In the inequality labeled $[a]$, the $\frac{1}{t(n)}$ term comes from part (b) of Remark 56 that each $R \in \mathcal{R}_n$ has $t(n)$ preimages under the maps $\{\mathbf{F}[\ , i, j]\}_{i\in[n], j\in[t(n)/n]}$; the term $\mu_{\mathcal{R}}(S_{n,y} \setminus U_{y,y}) \cdot (t(n) - o(t(n)))$ is the measure of all $R' \in \cup_{i,j}\mathbf{F}[S_{n,y}, i, j]$ for which $M^{R',\pi}(0^n, y)$ fails (Lemma 57). ◀

References

1 Scott Aaronson and Avi Wigderson. Algebrization: A New Barrier in Complexity Theory. *ACM Transactions on Computation Theory*, 1(1):2:1–2:54, 2009.

2 Martin R Albrecht, Carlos Cid, Lorenzo Grassi, Dmitry Khovratovich, Reinhard Lüftenegger, Christian Rechberger, and Markus Schofnegger. Algebraic Cryptanalysis of STARK-Friendly Designs: Application to MARVELlous and MiMC. IACR Cryptology ePrint Archive, Report 2019/419, 2019.

3 Martin R Albrecht, Lorenzo Grassi, Léo Perrin, Sebastian Ramacher, Christian Rechberger, Dragos Rotaru, Arnab Roy, and Markus Schofnegger. Feistel Structures for MPC, and More. IACR Cryptology ePrint Archive, Report 2019/397, 2019.

4 Abdelrahaman Aly, Tomer Ashur, Eli Ben-Sasson, Siemen Dhooghe, and Alan Szepieniec. Efficient Symmetric Primitives for Advanced Cryptographic Protocols. IACR Cryptology ePrint Archive, Report 2019/426, 2019.

5 Sanjeev Arora, Russell Impagliazzo, and Umesh Vazirani. Relativizing versus nonrelativizing techniques: The role of local checkability. Manuscript, 1992.

6 Tomer Ashur and Siemen Dhooghe. MARVELlous: a STARK-friendly family of cryptographic primitives. IACR Cryptology ePrint Archive, Report 2018/1098, 2018.

7 Bariş Aydınlıoğlu and Eric Bach. Affine Relativization: Unifying the Algebrization and Relativization Barriers. *ACM Transactions on Computation Theory*, 10(1):1:1–1:67, 2018.

8 László Babai, Lance Fortnow, Leonid A. Levin, and Mario Szegedy. Checking computations in polylogarithmic time. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing*, STOC '91, pages 21–32, 1991.

9 Theodore Baker, John Gill, and Robert Solovay. Relativizations of the P =? NP question. *SIAM Journal on Computing*, 4(4):431–442, 1975.

10 Eli Ben-Sasson, Iddo Bentov, Alessandro Chiesa, Ariel Gabizon, Daniel Genkin, Matan Hamilis, Evgenya Pergament, Michael Riabzev, Mark Silberstein, Eran Tromer, and Madars Virza. Computational integrity with a public random string from quasi-linear PCPs. In *Proceedings of the 36th Annual International Conference on Theory and Application of Cryptographic Techniques*, EUROCRYPT '17, pages 551–579, 2017.

11 Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Scalable Zero Knowledge with No Trusted Setup. In *Proceedings of the 39th Annual International Cryptology Conference*, CRYPTO '19, pages 733–764, 2019.

12 Eli Ben-Sasson, Iddo Bentov, Ynon Horesh, and Michael Riabzev. Fast Reed–Solomon Interactive Oracle Proofs of Proximity. In *Proceedings of the 45th International Colloquium on Automata, Languages and Programming*, ICALP '18, pages 14:1–14:17, 2018.

13 Eli Ben-Sasson, Alessandro Chiesa, Michael A. Forbes, Ariel Gabizon, Michael Riabzev, and Nicholas Spooner. Zero Knowledge Protocols from Succinct Constraint Detection. In *Proceedings of the 15th Theory of Cryptography Conference*, TCC '17, pages 172–206, 2017.

14 Eli Ben-Sasson, Alessandro Chiesa, Ariel Gabizon, Michael Riabzev, and Nicholas Spooner. Interactive Oracle Proofs with Constant Rate and Query Complexity. In *Proceedings of the 44th International Colloquium on Automata, Languages and Programming*, ICALP '17, pages 40:1–40:15, 2017.

15 Eli Ben-Sasson, Alessandro Chiesa, Ariel Gabizon, and Madars Virza. Quasilinear-Size Zero Knowledge from Linear-Algebraic PCPs. In *Proceedings of the 13th Theory of Cryptography Conference*, TCC '16-A, pages 33–64, 2016.

16 Eli Ben-Sasson, Alessandro Chiesa, Lior Goldberg, Tom Gur, Michael Riabzev, and Nicholas Spooner. Linear-Size Constant-Query IOPs for Delegating Computation. In *Proceedings of the 17th Theory of Cryptography Conference*, TCC '19, 2019.

17 Eli Ben-Sasson, Alessandro Chiesa, Michael Riabzev, Nicholas Spooner, Madars Virza, and Nicholas P. Ward. Aurora: Transparent Succinct Arguments for R1CS. In *Proceedings of the 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, EUROCRYPT '19, pages 103–128, 2019. Full version available at `https://eprint.iacr.org/2018/828`.

18 Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive Oracle Proofs. In *Proceedings of the 14th Theory of Cryptography Conference*, TCC '16-B, pages 31–60, 2016.

19 Eli Ben-Sasson and Madhu Sudan. Short PCPs with Polylog Query Complexity. *SIAM Journal on Computing*, 38(2):551–607, 2008. Preliminary version appeared in STOC '05.

20 Charles H Bennett and John Gill. Relative to a Random Oracle A, $\mathrm{P}^A \neq \mathrm{NP}^A \neq \text{co-NP}^A$ with Probability 1. *SIAM Journal on Computing*, 10(1):96–113, 1981.

21 Dan Boneh and Richard J. Lipton. Algorithms for Black-Box Fields and their Application to Cryptography. In *Proceedings of the 16th Annual International Cryptology Conference*, CRYPTO '96, pages 283–297, 1996.

22 Richard Chang, Benny Chor, Oded Goldreich, Juris Hartmanis, Johan Håstad, Desh Ranjan, and Pankaj Rohatgi. The random oracle hypothesis is false. *Journal of Computer and System Sciences*, 49(1):24–39, 1994.

23 Alan Cobham. The Intrinsic Computational Difficulty of Functions. In *Proceedings of the 1964 International Congress in Logic, Methodology and Philosophy of Science*, pages 24–30, 1965.

24 Irit Dinur. The PCP theorem by gap amplification. *Journal of the ACM*, 54(3):12, 2007.

25 Uriel Feige, Shafi Goldwasser, László Lovász, Shmuel Safra, and Mario Szegedy. Approximating clique is almost NP-complete. In *Proceedings of the 32nd Annual Symposium of Foundations of Computer Science*, pages 2–12, 1991.

26 Amos Fiat and Adi Shamir. How to prove yourself: practical solutions to identification and signature problems. In *Proceedings of the 6th Annual International Cryptology Conference*, CRYPTO '86, pages 186–194, 1986.

27 Lance Fortnow. The Role of Relativization in Complexity Theory. *Bulletin of the EATCS*, 52:229–243, 1994.

28 Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989. Preliminary version appeared in STOC '85.

29 Lorenzo Grassi, Daniel Kales, Dmitry Khovratovich, Arnab Roy, Christian Rechberger, and Markus Schofnegger. Starkad and Poseidon: New Hash Functions for Zero Knowledge Proof Systems. IACR Cryptology ePrint Archive, Report 2019/458, 2019.

30 Lorenzo Grassi, Reinhard Lüftenegger, Christian Rechberger, Dragos Rotaru, and Markus Schofnegger. On a Generalization of Substitution-Permutation Networks: The HADES Design Strategy. Cryptology ePrint Archive, Report 2019/1107, 2019.

31 Juris Hartmanis, Richard Chang, Suresh Chari, Desh Ranjan, and Pankaj Rohatgi. Relativization: A revisionistic retrospective. In *Bulletin of the EATCS*, 1992.

32 Russell Impagliazzo, Valentine Kabanets, and Antonina Kolokolova. An axiomatic approach to algebrization. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing*, STOC '09, pages 695–704, 2009.

33 Yael Kalai and Ran Raz. Interactive PCP. In *Proceedings of the 35th International Colloquium on Automata, Languages and Programming*, ICALP '08, pages 536–547, 2008.

34 Carsten Lund, Lance Fortnow, Howard J. Karloff, and Noam Nisan. Algebraic Methods for Interactive Proof Systems. *Journal of the ACM*, 39(4):859–868, 1992.

35 Ueli Maurer and Stefan Wolf. Lower bounds on generic algorithms in groups. In *Proceedings of the 17th Annual International Conference on Theory and Application of Cryptographic Techniques*, EUROCRYPT '98, pages 72–84, 1998.

36 Ueli M. Maurer. Abstract Models of Computation in Cryptography. In *Proceedings of the 10th IMA International Conference on Cryptography and Coding*, IMA '05, pages 1–12, 2005.

37 Thilo Mie. Short PCPPs verifiable in polylogarithmic time with O(1) queries. *Annals of Mathematics and Artificial Intelligence*, 56:313–338, 2009.

38 Vassiliy Ilyich Nechaev. Complexity of a determinate algorithm for the discrete logarithm. *Mathematical Notes*, 55:165–172, 1994.

39 Omer Reingold, Ron Rothblum, and Guy Rothblum. Constant-Round Interactive Proofs for Delegating Computation. In *Proceedings of the 48th ACM Symposium on the Theory of Computing*, STOC '16, pages 49–62, 2016.

40 Noga Ron-Zewi and Ron D. Rothblum. Local Proofs Approaching the Witness Length. Cryptology ePrint Archive, Report 2019/1062, 2019.

41 Adi Shamir. IP = PSPACE. *Journal of the ACM*, 39(4):869–877, 1992.

42 Victor Shoup. Lower bounds for discrete logarithms and related problems. In *Proceedings of the 16th International Conference on the Theory and Application of Cryptographic Techniques*, EUROCRYPT '97, pages 256–266, 1997.

43 Paul Valiant. Incrementally Verifiable Computation or Proofs of Knowledge Imply Time/Space Efficiency. In *Proceedings of the 5th Theory of Cryptography Conference*, TCC '08, pages 1–18, 2008.

44 ZCash. What is Jubjub?, 2017. URL: `https://z.cash/technology/jubjub.html`.

Lower Bounds for (Non-Monotone) Comparator Circuits

Anna Gál
University of Texas at Austin, Austin TX, United States of America
panni@cs.utexas.edu

Robert Robere
Institute for Advanced Study, Princeton NJ, United States of America
rrobere@ias.edu

Abstract

Comparator circuits are a natural circuit model for studying the concept of *bounded fan-out* computations, which intuitively corresponds to whether or not a computational model can make "copies" of intermediate computational steps. Comparator circuits are believed to be weaker than general Boolean circuits, but they can simulate Branching Programs and Boolean formulas. In this paper we prove the first superlinear lower bounds in the general (non-monotone) version of this model for an explicitly defined function. More precisely, we prove that the n-bit Element Distinctness function requires $\Omega((n/\log n)^{3/2})$ size comparator circuits.

2012 ACM Subject Classification Theory of computation → Circuit complexity

Keywords and phrases comparator circuits, circuit complexity, Nechiporuk, lower bounds

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.58

Funding *Robert Robere*: Funding provided by an NSERC Postdoctoral Fellowship and the Charles Simonyi Endowment.

Acknowledgements Some of this work was done while the authors were visiting the Simons Institute for the Theory of Computing in Berkeley, and while R.R. was at DIMACS.

1 Introduction

One of the central open problems in circuit complexity is to prove superlinear lower bounds on the size of general Boolean circuits for explicitly defined functions. While theorists have been outstandingly successful at analyzing some restricted circuit models – for instance, exponential lower bounds are known when the circuit is monotone [22], or bounded-depth [11] – the progress remains modest for less restricted classes of circuits:

1. The best lower bounds for the *Boolean formula size* of explicit functions over n variables are of the form $\Omega(n^{3-o(1)})$ [12, 18, 26, 5, 7, 19, 27, 4, 8]. The first such bound is due to Håstad [12] and the current largest bound (improving lower order terms) is due to Tal [27].
2. The best lower bounds for *branching programs*, which capture small-space computation and can simulate formulas, are $\Omega((n/\log n)^2)$ for the deterministic model by Nechiporuk [21] and $\Omega(n^{3/2}/\log n)$ for the non-deterministic- and parity- models by Pudlák[1] [15], and Karchmer and Wigderson [15], respectively.
3. The best lower bounds for *span programs* over $GF(2)$, which can simulate all of the above models, is $\Omega(n^{3/2}/\log n)$ by Beimel, Gál, and Paterson [3], following the initial $\Omega(n \log\log\log^* n)$ lower bounds by Karchmer and Wigderson [15].
4. Finally (and, perhaps, most notoriously) the best lower bounds for *general* Boolean circuits are $5n - o(n)$, due to Iwama and Morizumi [13].

[1] Pudlák's result is unpublished, and referenced by Karchmer and Wigderson in their paper [15].

Improving any of these lower bounds is an outstanding open problem; however, the problem of proving *any* superlinear lower bound on general Boolean circuits is perhaps the most interesting. Span programs form a remarkable model here, since they can simulate all models above for which we have superlinear lower bounds – this means that improving the lower bounds for span programs is one of the outstanding open question at the "frontier" of current techniques for proving lower bounds in general non-monotone circuit models.

In this paper we identify a new problem at the frontier: the complexity of non-monotone *comparator circuits*. A comparator circuit is a circuit composed purely of *comparator gates*, each of which maps a pair of bits (x, y) to $(x \wedge y, x \vee y)$. It is known that, just like span programs, comparator circuits can efficiently simulate non-deterministic branching programs [6], and thus lower bounds for comparator circuits imply lower bounds on branching programs and formulas. However, like branching programs, comparator circuits are known to be stronger than boolean formulas, as it is easy to construct comparator circuits computing Parity in $O(n)$ size (unlike formulas, which require $\Omega(n^2)$ size [16]). Similarly, like span programs, comparator circuits are believed to be weaker than general non-monotone circuits [6, 24]. However, it appears that comparator circuits may be *incomparable* to, or possibly even stronger than span programs: the class of functions computable by polynomial-size comparator circuits is conjectured to be incomparable to NC [25], and this conjecture is supported by oracle separations [6]. In contrast, the class of functions computable by polynomial-size span programs over finite fields is contained in NC^2. Furthermore, while we already had superlinear lower bounds for span programs [3, 15], there have been *no* superlinear lower bounds known for the comparator circuit size of any explicit function.

Continuing the line of work proving superlinear lower bounds for formulas, branching programs, and span programs, we prove the *first* superlinear lower bounds on the size of comparator circuits computing an explicit Boolean function. Let $n = 2m \log m$, and let $ED_n : \{0,1\}^n \to \{0,1\}$ be the n-bit *Element Distinctness* function, which takes m integers in the range $\{1, 2, \ldots, m^2\}$ as input (each encoded as $2 \log m$ bits) and outputs 1 iff all of the integers are distinct. Our main result is the following:

▶ **Theorem 1.** *The size of any comparator circuit computing the n-bit Element Distinctness function ED_n is at least $\Omega((n/\log n)^{3/2})$.*

In fact, we prove the stronger statement, that the number of *wires* of any comparator circuit computing ED_n is at least $\Omega((n/\log n)^{3/2})$. (See Section 2 for definitions.)

The above result provides a new and natural class of circuits for which we can obtain superlinear size lower bounds, and thus represents progress towards the goal of proving superlinear lower bounds for general non-monotone circuits. We believe that the techniques we employ are also interesting. The current best $\Omega(n^{3-o(1)})$ lower bounds for formulas follow by restriction based techniques, which we do *not* use for our lower bounds. In fact, as we discuss in Section 5, there seem to be obstacles to obtaining superlinear lower bounds for comparator circuits this way. Instead, we use a generalized version of the classic *Nechiporuk method* [21], which was used for obtaining the current best lower bounds for branching programs and span programs stated above. However, it is not possible to apply Nechiporuk's method directly to estimate the number of gates of comparator circuits, because the partial computations corresponding to subcircuits over disjoint subsets of inputs may significantly overlap (see Remark 8 and Section 1.2 for more details.) Instead, we need to exploit some interesting structure of comparator circuits to enable a more general version of the method. For these reasons we believe that our above lower bound (and comparator circuits more generally) are particularly interesting.

1.1 Comparator Circuits and Bounded Fanout Computation

Let us first describe comparator circuits more formally. As stated above, a *comparator gate* computes the map $(x, y) \mapsto (x \wedge y, x \vee y)$ where $x, y \in \{0, 1\}$; that is, the gate outputs both the AND and the OR of its input bits. A *comparator circuit* is a Boolean circuit whose inputs are labeled by input literals (i.e. either variables x_i or their negations $\overline{x}_i$), and is composed entirely of comparator gates; the comparator circuit is said to be *monotone* if no input literal is negated. We emphasize here that every comparator gate in the circuit has exactly two inputs and two outputs, and the (Boolean) output of the circuit is obtained by fixing some designated output of a comparator gate in the circuit.

The definition of a comparator gate suggests a convenient method of visualizing such circuits, depicted in Figure 1. We draw m lines from left-to-right – we call these lines *wires*[2] – that are initially labeled with input literals or the Boolean constants $0, 1$. We connect wires together with *comparator gates*: each gate is drawn connecting a pair of wires; one of the inputs is drawn with a *circle* (representing the output of the $\wedge$ gate) and the other with an *arrow* (representing the output of the $\vee$ gate).

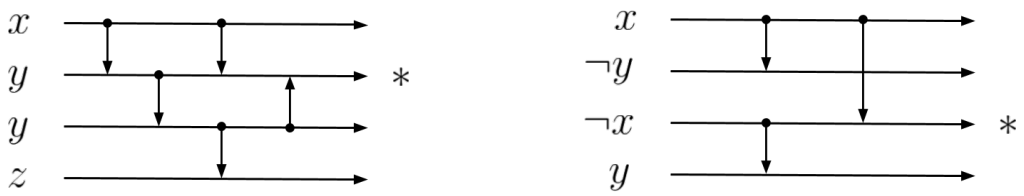


Figure 1 Examples of comparator circuits, with outputs designated. The left is a simple monotone comparator circuit, the right is a non-monotone comparator circuit for parity.

The most studied examples of comparator circuits in the literature are *sorting networks*, which are typically studied in a more general (i.e. non-Boolean) setting. A sorting network is a mathematical model of a sorting algorithm which is *input-oblivious*: the algorithm performs the same sequence of compare-and-swap operations on its input sequence of integers for all possible inputs. There are n wires traveling from left to right, each labeled with a distinct input variable, and a sequence of *comparator gates* connecting the wires. (Note that a comparator gate, intuitively, just *sorts* its inputs in order, with the larger input sent to the $\vee$ output and the smaller to the $\wedge$ output.) The goal is to construct the *fastest* network – where fastest can mean "shallowest", with the smallest depth connecting inputs to outputs, or with the fewest possible comparator gates – that sorts all n inputs. Shallow sorting networks have many applications in theory and practice, and thus have been extensively studied in theoretical computer science, with much effort spent on finding the shallowest possible networks (see [1, 2, 17, 9]). A restricted version of comparator circuits appeared implicitly in a paper by Graham [10]. His model is equivalent to *read once* comparator circuits, where each literal is used on at most one wire. Graham [10] showed that there exists a Boolean function on 11 variables that cannot be computed by this model, and it was proved by Robere [23] that the s, t- connectivity problem cannot be solved by read once monotone comparator circuits.

[2] We note that this differs from the standard usage of the word "wires" in circuit complexity, which is usually used to mean the edges in the underlying directed graph of the circuit.

Studying comparator circuits as a circuit model for general computation arguably began with the work of Subramanian [25], who showed that comparator circuits are a natural model for extending the study of *bounded fan-out* circuits beyond Boolean formulas. Subramanian considered different classes of circuits depending on whether or not the gates in the circuit could simulate a COPY gate: that is, the gate that takes some input x and outputs two copies of x. For example, when a gate can "fan-out" its output then we are allowing the gate to implicitly copy its output many times over. On the other hand, it is easy to see that a circuit constructed of comparator gates can *not* simulate a COPY gate, as the Hamming weight of the *output* of a comparator gate is always the same as the Hamming weight of the *input* of a comparator gate. Thus comparator circuits form an interesting intermediate class between Boolean formulas and Boolean circuits: they cannot create copies of intermediate computations, and so are apparently weaker than Boolean circuits; but, the ability to compute AND and OR simultaneously makes them apparently more powerful than Boolean formulas.

Let CC denote the class of languages computable by AC^0-uniform polynomial-size comparator circuits. This complexity class has an interesting characterization as the class of problems reducible to the *stable marriage* problem [25]. Closer to our purposes, Subramanian showed that if a circuit composed from gates chosen from a set S has "bounded fanout", in the sense that the gates from S cannot simulate a COPY gate, and all the gates in S compute monotone functions, then the corresponding circuit value problem for circuits with gates from S is either in NC or is CC-hard [25]. Despite their inability to copy, polynomial-size comparator circuits can compute everything in NL [20, 6] (which is conjectured to be strictly more powerful than polynomial-size Boolean formulas), and appear to be incomparable with NC [25]. Therefore, CC is a natural class to continue the study of bounded fan-out computation past NC^1, along with being a candidate for a class C satisfying $\mathsf{NL} \subset C \subset \mathsf{P}$ and $C \neq \mathsf{NC}$. The computational complexity of the class CC was recently analyzed by Cook, Filmus and Le [6] where, among other things, the question of uniformity and oracle separations with other classes was addressed.

However, a natural question left open by [6] is that of *lower bounds*. In the monotone case, since monotone comparator circuits are simulated by monotone circuits, exponential lower bounds are implied by the known lower bounds for monotone circuits (for instance, by Razborov [22]), and exponential separations between monotone comparator circuits and monotone circuits have been proved in [24]. Lower bounds for general Boolean circuits imply lower bounds for comparator circuits, but other than such implications, we are not aware of any previous nontrivial lower bounds for general (non-monotone) comparator circuits. In particular, before our work, no superlinear lower bounds have been obtained for comparator circuits computing an explicit Boolean function.

1.2 Techniques

To prove Theorem 1 we recruit a classic tool from non-monotone circuit lower bounds: the *Nechiporuk method* [21]. This method was invented by Nechiporuk to prove lower bounds on the size of deterministic branching programs and formulas, and is currently one of the few techniques capable of proving superlinear lower bounds against non-monotone computation. The idea of the method is as follows, in the setting of branching programs (the argument for formulas is identical, see e.g. [29]). Consider a branching program B computing a Boolean function f on n variables. First, observe that fixing the values outside of a subset $S \subseteq [n]$ of the variables to constants results in a branching program that computes a subfunction of the original function on the variables in the subset S. It follows that as we range over all substitutions to $[n] \setminus S$, the number of different branching programs obtained must be

at least as large as the number of different subfunctions of f on S. Thus, to obtain a lower bound, take any partition of the input variables $[n] = Y_1 \cup Y_2 \cup \cdots \cup Y_n$ and let h_i be the number of nodes of the branching program B labeled with variables from the set Y_i. Recall that the size of a deterministic branching program is the number of its nodes, and each node is labeled by exactly one input variable. Thus, the size of the branching program B is $|B| = \sum_i h_i$ and, in turn, we can lower-bound each h_i by relating it to the number of subfunctions of f on the variables Y_i obtained by restricting $[n] \setminus Y_i$ in all possible ways. This yields lower bounds for functions with large number of different subfunctions.

The Nechiporuk method works well for branching programs as the sets of nodes of the program associated with disjoint subsets of inputs are disjoint. The situation is similar for Boolean formulas: each leaf of the formula is labeled by a variable (or negated variable), and again the sets of leaves associated with disjoint subsets of inputs are disjoint. In fact, this property (that the subcircuits corresponding to disjoint subsets of inputs are disjoint) is true for *all* prior applications of the method. For example, in the application to non-deterministic and parity branching programs [15], the *edges* of the program are now labeled by variables, and thus the sets of edges corresponding to disjoint subsets of variables are disjoint. Similarly, in span programs over $GF(2)$ [3], the rows are labeled by variables and so once again, subsets of rows corresponding to disjoint subset of variables are disjoint.

In principle, one could imagine generalizing this method to other circuit models, as long as the subcomputations corresponding to disjoint subsets of input variables do not overlap much. On the negative side, the method can provably *not* be applied to general Boolean circuits [28], which intuitively makes sense as circuits can copy intermediate computations as many times as they like.

So: what about comparator circuits? At first glance it seems like the method should not be applicable to comparator circuits either, since the circuits contain gates with fanout *two*, and this implies that the computation of the circuits on disjoint sets of variables can badly overlap, just like general Boolean circuits and unlike formulas, branching programs or span programs. Indeed, as we discuss in Remark 8, the set of gates of a comparator circuit contributing to the computation on a given subset of variables can potentially involve *all* the gates of the original circuit.

We can, however, overcome this problem. Our key observation is the following: the idea of Nechiporuk's method may be still applicable in situations where the subcomputations badly overlap, as long as we are able to bound the size of the computation by some (not too fast growing) function of the number of occurrences of input literals. This is simply because the *occurrences of input literals* (or in other words the *input queries*) made by the computation will always be disjoint over disjoint subsets of variables.

Let us elaborate further in the case of comparator circuits. In comparator circuits, the number of *wires* – which, we recall, refers to the number of distinct left-to-right lines in Figure 1 – corresponds exactly to the number of occurrences of input literals in the circuit. The fact that comparator circuits cannot simulate COPY gates suggests the following question. If F is a Boolean formula with ℓ leaves then it is easy to see that the number of internal gates s of F is *exactly* $s = \ell - 1$. In other words, the size of the formula F is *linearly* related to the number of its leaves, and this is a result of F having bounded fanout. Does a similar relation hold for comparator circuits between *gates* and *wires*?

First, observe that since each gate in the comparator circuit connects two wires, in order for each wire to be connected to the output gate we must have $\ell - 1 \leq s$, where s is the number of gates, just like in formulas. However, is it the case that $s = O(\ell)$? Or even $s = O(\text{poly}(\ell))$? Surprisingly, we are able to show that $s \leq \binom{\ell}{2}$ assuming that none of the gates in the circuit

are "useless" in a specific technical sense and can be removed (cf. Section 3); furthermore, we show that this relationship is sharp. For the special case of read once comparator circuits, essentially this statement was proved implicitly by Graham [10].

▶ **Theorem 2.** *Let C be any comparator circuit with ℓ wires and s gates such that every gate in C is useful. Then $s \leq \ell(\ell-1)/2$.*

It is precisely this relationship between wires and gates which allows us to overcome the "overlapping subcomputations" issue in applying Nechiporuk's method, and it seems to be a remarkable property of this model. As far as we know, our results are the first generalization of Nechiporuk's method to such a scenario.

2 Definitions

For any positive integer m let $[m] := \{1, 2, \ldots, m\}$, and if n is an integer satisfying $m \leq n$ let $[m, n] := \{m, m+1, \ldots, n\}$. If S is a set then $\wp(S)$ denotes the power set of S.

Throughout we will be interested in Boolean functions $f : \{0,1\}^n \to \{0,1\}$. If $x \in \{0,1\}^n$ then x_i is the value of the ith bit in x. If $x, y \in \{0,1\}^n$ are binary strings then $x \leq y$ if $x_i \leq y_i$ for all $i \in [n]$. If $x \in \{0,1\}^n$ then $|x|$ is the Hamming weight of x.

A *comparator gate* is the function $C : \{0,1\}^2 \to \{0,1\}^2$ computing the map $(x, y) \mapsto (x \wedge y, x \vee y)$. It is natural to think of a comparator gate C as "sorting" the input (x, y), as the smaller input goes to the first coordinate and the larger input goes to the second. A *comparator circuit* is a circuit composed of comparator gates with arbitrary Boolean literals as input. For later convenience, we use the following, alternative definition of a comparator circuit. A comparator circuit C is defined by a tuple (W, G, I, o), where

- W is a set of elements called *wires*,
- $I : W \to \{x_1, \ldots, x_n, \overline{x}_1, \ldots, \overline{x}_n, 0, 1\}$ is an initial labeling of each wire with an input literal or Boolean constant,
- $G \subseteq W^2$ is an ordered list of *comparator gates*,
- $o \in W$ is the designated output wire.

Each comparator gate $g = (i, j) \in G$ is specified by the pair of wires the gate connects: the AND output of the gate is interpreted as the first coordinate and the OR output of the gate is interpreted as the second coordinate. A comparator circuit computes a Boolean function in the natural way: given an assignment to the input literals, we evaluate the comparator gates one by one in order according to G, and then output the Boolean value labeled on the output wire, o. We define the *size* of the comparator circuit to be the number of gates in the circuit, and note that the number of wires in the circuit will also be an interesting complexity measure.

3 Wires vs. Gates in Comparator Circuits

▶ **Definition 3.** *Let C be a comparator circuit with m wires, let $g = (i, j)$ be a gate in C, and let $v_i(x), v_j(x)$ be the values of the wires i, j, respectively, on input x just before applying the gate g. Then g is* useful *if there is a pair of inputs x, y to C such that $(v_i(x), v_j(x)) = (1, 0)$ and $(v_i(y), v_j(y)) = (0, 1)$. If g is not useful then we say it is* useless.

Equivalently, if a gate g is useless then for every pair of inputs (x, y) either $(v_i(x), v_j(x)) = (v_i(y), v_j(y))$ or $v_i = v_j$ for one of x, y. The next proposition states that gates that are not useful can be removed without loss of generality.

▶ **Proposition 4.** *Let C be a comparator circuit with ℓ wires and s gates computing some Boolean function f. Then there exists a comparator circuit C' with (at most) ℓ wires and $s' \leq s$ gates computing f such that every gate in C' is useful. Moreover, the labels of the wires that are not removed remain the same in C' as in C.*

Proof. Let $g = (i, j)$ be any gate in C such that g is not useful. Let $v_i(x), v_j(x)$ be the values of the wires i, j before applying g on input x. Since g is not useful, then it follows that for every pair of inputs x, y to the circuit either $(v_i(x), v_j(x)) = (v_i(y), v_j(y))$ or one of x, y satisfies $v_i = v_j$.

Let $O \subseteq \{0,1\}^n$ be the set of inputs such that for all $x \in O$, exactly one of $v_i(x), v_j(x)$ is 1. Since g is useless, it follows that either $(v_i(x), v_j(x)) = (0, 1)$ for every x in O or $(v_i(x), v_j(x)) = (1, 0)$ for every x in O.

Assume that the outputs of g are ordered so that $g(0, 1) = (0, 1)$ and $g(1, 0) = (0, 1)$. The proof for the other type of gates is symmetric. Consider the following two cases.

Case 1: For all $x \in O$, $(v_i(x), v_j(x)) = (0, 1)$.

We claim that for all $x \in \{0,1\}^n$, applying g does not change the value of any wire, and so g can be removed without affecting the computation C. This is because for every $x \in \{0,1\}^n$, either $x \in O$ and so applying g does not change any value (since $g(0, 1) = (0, 1)$), or $x \notin O$, and by the definition of the set O we must have $v_i(x) = v_j(x)$, and so applying g will not change the values of the wires.

Case 2: For all $x \in O$, $(v_i(x), v_j(x)) = (1, 0)$.

By definition $g(1, 0) = (0, 1)$, and so we perform the following operation: remove g, and for each wire i, j find the *first* gate h using that wire as an input. If it is the same gate for both wires, then h is redundant, so remove it and continue searching forward along each wire. Otherwise, let $h_i = (k, \ell)$ where exactly one of k, ℓ is i and let $h_j = (k', \ell')$ where exactly one of k', ℓ' is j. Let the wire i feed into the gate h_j and the wire j feed into the gate h_i. Switching the wires this way will simulate the action of the gate g that we just eliminated.

Once you have found a non-redundant gate h for each of the two wires $\{i, j\}$ (or if no such gate exists), then stop.

To see that this operation does not affect the output of the circuit, consider the following two cases:

1. $v_i(x) = v_j(x)$. In this case, performing this operation will not affect the computation of the circuit as either of the h gates will receive the same input, regardless of which wire they are connected to.
2. $v_i(x) \neq v_j(x)$. In this case, $(v_i(x), v_j(x)) = (1, 0)$, and $g(1, 0) = (0, 1)$, so by removing g and switching the inputs of the h gates we ensure that in the new circuit the computation will be the same.

By proceeding from the inputs of the circuit to the outputs we can apply this operation to remove every useless gate in the circuit. In each case, the output of the circuit will not be affected.

Finally, notice that if a wire is not connected to any remaining gate in C', we may remove it, except if the wire is designated as the output of the circuit. The wires that are not removed, retain the same label as in the original circuit C, and the proof of the proposition is complete. ◀

Now we can prove the main theorem of this section, which we restate from the Introduction.

▶ **Theorem 2.** *Let C be any comparator circuit with ℓ wires and s gates such that every gate in C is useful. Then $s \leq \ell(\ell-1)/2$.*

Proof. For $j = 1, 2, \ldots, s$ let g_j denote the jth gate of C, and let C_j be the subcircuit of C consisting of the first j gates of C. Let C_0 be the circuit C with all of its gates removed. We say that two inputs $x, y \in \{0,1\}^n$ are *useful with respect to the pair* (i,j) if $w_i(x) = 1, w_i(y) = 0$ and $w_j(x) = 0, w_j(y) = 1$, where w_i, w_j are the outputs of wires i and j of the comparator circuit C. If there exist two inputs $x, y \in \{0,1\}^n$ that are useful with respect to (i,j) then we say that (i,j) is a *useful pair*.

Let U_j denote the collection of all useful pairs in the circuit C_j. We show that the size of U_j decreases by at least one after applying each gate: that is $|U_j| \leq |U_{j-1}| - 1$ for all j. Since $|U_0| \leq \binom{ell}{2}$, it follows that the number of gates in C is at most $\binom{\ell}{2}$.

Let $1 \leq j \leq s$ be an integer, and suppose the gate g_j connects two wires (a,b). Since g_j is useful we have that (a,b) is a useful pair in U_{j-1}. It is also clear that $(a,b) \notin U_j$, as applying the gate g_j removes all useful inputs with respect to (a,b). However, we are not finished, as applying the gate g_j could have introduced a new useful pair (c,d): so, $(c,d) \in U_j$ and $(c,d) \notin U_{j-1}$. The proof will be complete once we prove the following claim, which states that if such a new useful pair is introduced, then there must exist another distinct useful pair that was removed.

▷ Claim 5. Suppose there exists a useful pair $(c,d) \in U_j$ such that $(c,d) \notin U_{j-1}$. Then there is another useful pair (α, β), uniquely associated with (c,d), such that $(\alpha,\beta) \neq (a,b)$, $(\alpha,\beta) \in U_{j-1}$, and $(\alpha,\beta) \notin U_j$.

Proof of Claim. It is clear that while $(c,d) \neq (a,b)$, it cannot be the case that $c, d \notin \{a,b\}$: that is, at least one of c or d must be equal to a or b. This is because applying the gate g_j only modifies the outputs of wires a and b, and so any new useful pair must include one of these modified outputs.

First assume that $c = a$ (and we note that the other cases ($c = b, d = a, d = b$) follow by nearly identical proofs). It follows that $(a,d) \in U_j$ and $(a,d) \notin U_{j-1}$. We show that this implies that $(b,d) \in U_{j-1}$ and $(b,d) \notin U_j$. For any wire t let v_t denote the output of the tth wire *before* applying the gate g_j, and let w_t denote the output of the tth wire *after* applying the gate g_j. Since (a,d) is a useful pair it follows that there exists a pair of inputs $x, y \in \{0,1\}^n$ such that $w_a(x) = 1, w_d(x) = 0$ and $w_a(y) = 0, w_d(y) = 1$. Now, since $(a,d) \notin U_{j-1}$ we can deduce the values of v_a, v_b, and v_d on inputs x and y. It is obvious that $v_d(x) = w_d(x)$ and $v_d(y) = w_d(y)$ as the gate g_j is not connected to the wire d. We can also conclude that $v_a(x) = v_b(x) = 1$, as the gate g_j connects a to b and $w_a(x) = 1$ (for if $v_b(x) = 0$, then $w_a(x) \neq 1$, as the 1 would have been moved to the $\vee$ output of g_j). Finally, since $(a,d) \notin U_{j-1}$ we know that $v_a(y) = 1$ and $v_b(y) = 0$. For if $v_a(y) = 0$, it would follow that (a,d) is a useful pair in U_{j-1} (a contradiction). Thus $v_a(y) = 1$, and since $w_a(y) = 0$, the only possibility is that the gate g_j moved a 1 from wire a to wire b on input y. We now record the values of v_a, v_b, v_d and w_a, w_b, w_d on inputs x and y:

	x	y
v_a	1	1
v_b	1	0
v_d	0	1
w_a	1	0
w_b	1	1
w_d	0	1

By examining the table it is easy to see that (b, d) is a useful pair in U_{j-1}, as $v_b(x) = 1, v_d(x) = 0$ and $v_b(y) = 0, v_d(y) = 1$. We show that $(b, d) \notin U_j$, proving the claim in this case.

To see this, suppose that $(b, d) \in U_j$. Then there must exist an input z such that $w_b(z) = 0, w_d(z) = 1$, on which we can similarly deduce the values of v_a, v_b, v_d. Clearly $v_d(z) = 1$ since the gate g_j connects the two wires (a, b). Since $w_b(z) = 0$, it must be that $v_a(z) = v_b(z) = 0$. However, this means that $v_a(z) = 0, v_d(z) = 1$, and from the table we can see that $v_a(x) = 1, v_d(x) = 0$. This means that (a, d) is a useful pair in U_{j-1}, a contradiction! ◁

We make a final remark on the uniqueness property. We technically have four cases to prove here, as the new useful pair must be in one of the following forms:

1. (a, d)
2. (b, d)
3. (c, a)
4. (c, b)

In the proof above we showed that if the new useful pair is of the form (a, d), then this implies that $(b, d) \in U_{j-1}$ and $(b, d) \notin U_j$. In the other three cases (which proceed by identical proofs), other uniquely associated pairs are introduced. In general, if $\alpha \in \{a, b\}$ and $\beta \in \{a, b\}$, $\beta \neq \alpha$, and the new useful pair is of the form (α, d) then the same proof shows that (β, d) is in U_{j-1} and not in U_j. Similarly, if the new useful pair is of the form (c, α) then a similar proof shows that (c, β) is in U_{j-1} and not in U_j. These facts together prove uniqueness. ◀

To see that the upper bound given in the previous theorem is sharp, consider the following comparator circuit C_n which sorts its n inputs via the "bubblesort" method: the circuit incrementally bubbles the largest value from the top wire down as far as it can go, and then the second wire, and so on. This circuit has n wires and $\binom{n}{2}$ gates, and it is not hard to see that every gate is useful.

4 Lower Bound for Element Distinctness

Let $n = 2m \log m$, and recall the definition of the Element Distinctness function ED_n: it takes $n = 2m \log m$ input bits divided into m blocks of $2 \log m$ bits each, interpreted as m integers in the range $\{1, \ldots, m^2\}$, and decides whether all m numbers are distinct.

In this section we prove our main lower bound, restated here for convenience:

▶ **Theorem 1.** *The size of any comparator circuit computing the n-bit Element Distinctness function ED_n is at least $\Omega((n/\log n)^{3/2})$.*

We note that the lower bound holds for any function with a similar number of subfunctions; for instance, the *Indirect Storage Access function* [29].

We will need the following Lemma. Recall that in a comparator circuit, each of the wires is labeled with either a constant $0, 1$, some input variable or its negation.

▶ **Lemma 6.** *Let $\ell \geq 1$ be an integer. For any fixed labeling of ℓ wires, the number of different Boolean functions that can be computed by comparator circuits with ℓ wires with the given labeling is at most ℓ^{ℓ^2}.*

Proof. By Proposition 4, every comparator circuit C with ℓ wires is equivalent to another comparator circuit C' with (at most) ℓ wires that has no useless gates. By Theorem 2 the number of gates of C' is at most $\binom{\ell}{2}$. Recall also that C' keeps the same labeling of the wires (that are not removed) as C; and if a wire is not connected to any remaining gate in C', we may remove it, except if the wire is designated as the output of the circuit.

To prove the lemma it is enough to estimate the number of different comparator circuits with at most ℓ wires of a given labeling and at most $\binom{\ell}{2}$ gates. For each gate, there are at most $\binom{\ell}{2}$ choices for the pair of wires it takes as inputs, and two choices for the ordering of the $\wedge$ output and $\vee$ output of the gate. In addition, we have at most ℓ choices to designate one of the wires as the output. Thus the number of possible such comparator circuits is at most $\ell \cdot (2\binom{\ell}{2})^{\binom{\ell}{2}} \leq (\ell)^{(\ell^2)}$. ◀

▶ Remark 7. We can also bound the number of different Boolean functions on n variables that are computed by comparator circuits with s gates by $(2(n+1)s)^{2s}$ using a similar counting argument. Note that here we do not assume a fixed labeling, and we use that the number of wires is at most the number of gates, assuming that each wire is connected to the output gate.

If the subsets of gates used in subcircuits over disjoint subsets of inputs would not overlap much, then a straightforward application of Nechiporuk's method seemingly would yield nearly quadratic lower bounds using this counting argument. However, as we discuss in some more details in Remark 8, the subsets of gates can badly overlap. Thus, even though a counting argument in terms of gates is available, Nechiporuk's argument is not applicable directly to the gates of comparator circuits.

Note that we could have also stated a similar bound for comparator circuits with ℓ wires and n input variables, without considering a fixed labeling of the wires. We find the current version of Lemma 6 more convenient for our purposes. We are now ready to prove Theorem 1.

Proof of Theorem 1. We prove the stronger statement, that the number of wires of any comparator circuit computing ED_n is at least $\Omega((n/\log n)^{3/2})$. Recall that the size of a comparator circuit is the number of its gates, and the number of gates in a comparator circuit with w wires, where each wire is connected to the output gate, is at least $w-1$. Thus, a lower bound on the number of wires implies lower bounds on the size of the comparator circuit.

Partition the $n = 2m\log m$ input variables into m groups of $2\log m$ variables each, such that the variables in the i-th group represent the i-th integer in the input. For $i = 1, \ldots, m$ let S_i be the set of variables in the i-th group. Let N_i denote the number of different subfunctions over the variables in S_i that can be obtained by fixing all variables outside S_i to constants. It is known (see e.g. [14]) that $N_i = 2^{\Omega(n)}$ for each $i = 1, \ldots, m$.

Let C be a comparator circuit computing ED_n with w wires. Let w_i denote the number of wires of C labeled by a variable from the i-th group. Then $w = \sum_{i=1}^{m} w_i$.

For a given $i \in [m]$, consider a fixed assignment α of constants 0 or 1 to all variables outside of S_i. Consider the resulting comparator circuit $C_{i,\alpha}$ over the variables in S_i. Applying Proposition 4 to $C_{i,\alpha}$ we can obtain a comparator circuit $C'_{i,\alpha}$ over variables from S_i with no useless gates. Notice that if a wire is labeled by constant 0 or 1 then any gate directly using this wire must be useless (in the formal sense of Definition 3). Note also that after removing all useless gates, wires with constant label are not connected to any remaining gates. Thus, they can be removed, except when designated as the output wire. Note however, that if a wire with constant label that is not connected to any gate is designated as the output wire, then the function computed is constant 1 or 0. Thus, unless the function computed by $C'_{i,\alpha}$ is constant, all wires in $C'_{i,\alpha}$ are labeled by variables or their negation from S_i, regardless of the particular assignment α.

This fact has two important consequences for us. First, it means that the number of wires of $C'_{i,\alpha}$ is at most w_i. Second, it means that for given i, unless the function computed by $C'_{i,\alpha}$ is constant, the wires of $C'_{i,\alpha}$ have the same labels (by variables or negated variables from S_i) regardless of the particular assignment α. (To see this, recall that in Proposition 4 the labels of the wires that are not removed remain the same as in the original circuit.)

This allows us to conclude using Lemma 6 that $N_i \leq 2 + w_i^{(w_i^2)}$ for $i = 1, \ldots, m$. Thus, we have for $i = 1, \ldots, m$ that

$$w_i^2 \geq \frac{\log(N_i - 2)}{\log w_i} \geq \frac{\Omega(n)}{\log w_i} .$$

Note that if $\log w_i > \frac{1}{2}\log n$ then $w_i \geq \sqrt{n}$. On the other hand, if $\log w_i \leq \frac{1}{2}\log n$ then we get $w_i^2 \geq \frac{\Omega(n)}{\log n}$. Thus, for $i = 1, \ldots, m$ we have $w_i \geq \Omega(\frac{\sqrt{n}}{\sqrt{\log n}})$, which yields $w \geq \Omega((n/\log n)^{3/2})$. ◀

▶ Remark 8. It is crucial in the above argument that the set of *wires* used by the subcircuits $C'_{i,\alpha}$ is the same for fixed i regardless of the assignment α, and that these sets do not overlap for different values of i. One could try to consider a similar argument directly for gates instead of wires. For instance, one could define $G_{i,\alpha}$ as the set of gates participating in the circuit $C'_{i,\alpha}$, and consider $G_i = \cup G_{i,\alpha}$. But for different assignments α, the circuits $C'_{i,\alpha}$ may retain different gates of the original circuit, and the sets G_i may badly overlap. In particular, for some values of i, G_i may contain all gates of the original circuit.

5 Conclusion and Future Work

In this paper we have proved the first superlinear lower bound on the size of comparator circuits computing an explicit Boolean function. As we have remarked above, we actually prove a superlinear lower bound on the number of *wires*, or equivalently, *input queries*, of any comparator circuit for ED_n, which is stronger than a lower bound on the number of gates. Furthermore, by our Theorem 2, there is at most a quadratic separation between the number of wires and the number of gates in any minimal comparator circuit. A natural problem is to try and prove a lower bound on the number of gates directly. However, as we discuss above in Remark 8 this would require a different technique.

We remark that it seems quite difficult to apply restriction techniques to obtain *wire* lower bounds for comparator circuits. This is for a simple reason: observe that restricting one input to a single comparator will restrict exactly one output of the gate and re-wire the other input to the other output. This implies that if we have a comparator circuit with m wires, and we restrict values to t of them, then we are left with a new comparator circuit with exactly $m - t$ unrestricted wires after propagating this rewiring process. Note that some wires could possibly be removed if they are "separated" from the output gate, but, if the topology of the circuit is highly connected (e.g. is an expander) then we should expect this to be very unlikely.

Finally, we remark on a second natural open problem. As we discuss in the introduction, the key structural property of comparator circuits that enabled us to apply Nechiporuk's method to comparator circuits is Theorem 2, relating the number of wires to the number of gates. The crucial intuition in the proof of this Theorem is that comparator circuits *cannot* copy intermediate computations. There is a rich structure of circuit classes extending comparator circuits which cannot copy intermediate computations, as explored by Subramanian [25]. Can one extend any of our results to these more general classes?

References

1. Miklós Ajtai, János Komlós, and Endre Szemerédi. An O(n log n) Sorting Network. In *Proceedings of the 15th Annual ACM Symposium on Theory of Computing, 25-27 April, 1983, Boston, Massachusetts, USA*, pages 1–9, 1983. doi:10.1145/800061.808726.
2. Kenneth E. Batcher. Sorting Networks and Their Applications. In *American Federation of Information Processing Societies: AFIPS Conference Proceedings: 1968 Spring Joint Computer Conference, Atlantic City, NJ, USA, 30 April - 2 May 1968*, pages 307–314, 1968. doi:10.1145/1468075.1468121.
3. Amos Beimel, Anna Gál, and Mike Paterson. Lower Bounds for Monotone Span Programs. *Computational Complexity*, 6(1):29–45, 1997. doi:10.1007/BF01202040.
4. Andrej Bogdanov. Small Bias Requires Large Formulas. In *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, pages 22:1–22:12, 2018. doi:10.4230/LIPIcs.ICALP.2018.22.
5. Ruiwen Chen, Valentine Kabanets, Antonina Kolokolova, Ronen Shaltiel, and David Zuckerman. Mining Circuit Lower Bound Proofs for Meta-Algorithms. *Computational Complexity*, 24(2):333–392, 2015. doi:10.1007/s00037-015-0100-0.
6. Stephen A. Cook, Yuval Filmus, and Dai Tri Man Le. The complexity of the comparator circuit value problem. *TOCT*, 6(4):15:1–15:44, 2014. doi:10.1145/2635822.
7. Irit Dinur and Or Meir. Toward the KRW Composition Conjecture: Cubic Formula Lower Bounds via Communication Complexity. *Computational Complexity*, 27(3):375–462, 2018. doi:10.1007/s00037-017-0159-x.
8. Anna Gál, Avishay Tal, and Adrian Trejo Nuñez. Cubic Formula Size Lower Bounds Based on Compositions with Majority. In *10th Innovations in Theoretical Computer Science Conference, ITCS 2019, January 10-12, 2019, San Diego, California, USA*, pages 35:1–35:13, 2019. doi:10.4230/LIPIcs.ITCS.2019.35.
9. Michael T. Goodrich. Zig-zag sort: a simple deterministic data-oblivious sorting algorithm running in O(n log n) time. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 684–693, 2014. doi:10.1145/2591796.2591830.
10. RL Graham. A mathematical study of a model of magnetic domain interactions. *Bell System Technical Journal*, 49(8):1627–1644, 1970.
11. Johan Håstad. Almost Optimal Lower Bounds for Small Depth Circuits. In *Proceedings of the 18th Annual ACM Symposium on Theory of Computing, May 28-30, 1986, Berkeley, California, USA*, pages 6–20, 1986. doi:10.1145/12130.12132.
12. Johan Håstad. The Shrinkage Exponent of de Morgan Formulas is 2. *SIAM J. Comput.*, 27(1):48–64, 1998. doi:10.1137/S0097539794261556.
13. Kazuo Iwama and Hiroki Morizumi. An Explicit Lower Bound of 5n - o(n) for Boolean Circuits. In *Mathematical Foundations of Computer Science 2002, 27th International Symposium, MFCS 2002, Warsaw, Poland, August 26-30, 2002, Proceedings*, pages 353–364, 2002. doi:10.1007/3-540-45687-2_29.
14. Stasys Jukna. *Boolean Function Complexity - Advances and Frontiers*, volume 27 of *Algorithms and combinatorics*. Springer, 2012. doi:10.1007/978-3-642-24508-4.
15. Mauricio Karchmer and Avi Wigderson. On Span Programs. In *Proceedings of the Eigth Annual Structure in Complexity Theory Conference, San Diego, CA, USA, May 18-21, 1993*, pages 102–111, 1993. doi:10.1109/SCT.1993.336536.
16. Valeriy M Khrapchenko. Method of determining lower bounds for the complexity of P-schemes. *Mathematical Notes*, 10(1):474–479, 1971.
17. Donald Ervin Knuth. *The art of computer programming, , Volume III, 2nd Edition*. Addison-Wesley, 1998. URL: http://www.worldcat.org/oclc/312994415.
18. Ilan Komargodski and Ran Raz. Average-case lower bounds for formula size. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 171–180, 2013. doi:10.1145/2488608.2488630.

19 Ilan Komargodski, Ran Raz, and Avishay Tal. Improved Average-Case Lower Bounds for DeMorgan Formula Size. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 588–597, 2013. doi:10.1109/FOCS.2013.69.

20 Ernst W. Mayr and Ashok Subramanian. The Complexity of Circuit Value and Network Stability. *J. Comput. Syst. Sci.*, 44(2):302–323, 1992. doi:10.1016/0022-0000(92)90024-D.

21 Edward I Nechiporuk. A Boolean function. *Engl. transl. in Sov. Phys. Dokl.*, 10:591–593, 1966.

22 Alexander A Razborov. Lower bounds for the monotone complexity of some Boolean functions. In *Soviet Math. Dokl.*, volume 31, pages 354–357, 1985.

23 Robert Robere. Notes on Comparator Circuits. Unpublished manuscript, 2014.

24 Robert Robere, Toniann Pitassi, Benjamin Rossman, and Stephen A. Cook. Exponential Lower Bounds for Monotone Span Programs. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 406–415, 2016. doi:10.1109/FOCS.2016.51.

25 Ashok Subramanian. *The computational complexity of the circuit value and network stability problems*. PhD thesis, Stanford University, 1990.

26 Avishay Tal. Shrinkage of De Morgan Formulae by Spectral Techniques. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 551–560, 2014. doi:10.1109/FOCS.2014.65.

27 Avishay Tal. Formula lower bounds via the quantum method. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 1256–1268, 2017. doi:10.1145/3055399.3055472.

28 Dietmar Uhlig. Boolean functions with a large number of subfunctions and small complexity and depth. In *International Symposium on Fundamentals of Computation Theory*, pages 395–404. Springer, 1991.

29 Ingo Wegener. *The complexity of Boolean functions*. Wiley-Teubner, 1987. URL: http://ls2-www.cs.uni-dortmund.de/monographs/bluebook/.

A Tight Lower Bound For Non-Coherent Index Erasure

Nathan Lindzey
Department of Computer Science, University of Colorado at Boulder, USA
nathan.lindzey@colorado.edu

Ansis Rosmanis
Graduate School of Mathematics, Nagoya University, Japan
http://rosmanis.com/research/index.html
ansis.rosmanis@math.nagoya-u.ac.jp

Abstract

The *index erasure problem* is a quantum state generation problem that asks a quantum computer to prepare a uniform superposition over the image of an injective function given by an oracle. We prove a tight $\Omega(\sqrt{n})$ lower bound on the quantum query complexity of the *non-coherent* case of the problem, where, in addition to preparing the required superposition, the algorithm is allowed to leave the ancillary memory in an arbitrary function-dependent state. This resolves an open question of Ambainis, Magnin, Roetteler, and Roland (CCC 2011), who gave a tight bound for the coherent case, the case where the ancillary memory must return to its initial state.

To prove our main result, we first extend the so-called *automorphism principle* (Høyer et al. STOC 2007) to the *general adversary method* for state conversion problems (Lee et al. STOC 2011), which allows one to exploit the symmetries of these problems to lower bound their quantum query complexity. Using this method, we establish a strong connection between the quantum query complexity of non-coherent symmetric state generation problems and the well-known *Krein parameters* of *association schemes*. Krein parameters are usually hard to determine, nevertheless, we give a novel way of computing certain Krein parameters of a commutative association scheme defined over partial permutations. We believe the study of this association scheme may also be of independent interest.

2012 ACM Subject Classification Theory of computation → Quantum query complexity; Mathematics of computing → Combinatorics

Keywords and phrases General Adversary Method, Quantum Query Complexity, Association Schemes, Krein Parameters, Representation Theory

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.59

Funding *Ansis Rosmanis*: The author is a JSPS International Research Fellow supported by the JSPS KAKENHI Grant Number JP19F19079. Part of this work was done while he was at the Centre for Quantum Technologies at the National University of Singapore supported by the Singapore Ministry of Education and the National Research Foundation under grant R-710-000-012-135.

Acknowledgements AR would like to thank Aleksandrs Belovs and Jérémie Roland for insightful discussions. NL thanks Chris Godsil for useful discussions.

1 Introduction

For proving lower bounds in the *oracle query model*, one assumes access to an oracle O_f that evaluates a black-box function $f\colon [n] \to [m]$ on input queries, where $[n] := \{1, 2, \cdots, n\}$ and $[m] := \{1, 2, \cdots, m\}$, and the goal is to prove that any algorithm for solving the computational problem at hand must make a certain number of oracle queries. This principle for proving lower bounds applies to both classical and quantum computation, and in the latter we let the oracle to be queried in a superposition.

Quantum query algorithms are known to surpass their classical counterparts for many important classical tasks, such as unstructured search, game tree evaluation, random walks, and others (see [17, 1] for recent surveys). Classical tasks aside, one may also be interested in *quantum mechanical tasks*, such as *quantum state generation*. A quantum state generation problem simply asks for a certain quantum state $|\psi_f\rangle$ to be generated on the target register. In this paper, we consider a particular state generation problem known as Index Erasure.

Given an injective function $f\colon [n] \to [m]$ via a black-box oracle O_f, Index Erasure is the task of preparing the quantum state that is the uniform superposition over the image of f, namely,

$$|\psi_f\rangle := \frac{1}{\sqrt{n}} \sum\nolimits_{x=1}^{n} |f(x)\rangle\,.$$

The name of the problem stems from the fact that a quantum computer can prepare the uniform superposition $\frac{1}{\sqrt{n}} \sum_{x=1}^{n} |x\rangle\,|f(x)\rangle$ using a single query to O_f, yet the task of ignoring or *erasing* the first register that records the *index* x is non-trivial. Indeed, if one could solve Index Erasure using a poly-logarithmic number of queries, one would obtain a time-efficient algorithm for Graph Isomorphism (we present more details in Appendix A).

The question of the complexity of Index Erasure was first raised by Shi in [23], where he already observed that the problem can be solved in $\mathrm{O}(\sqrt{n})$ queries by an algorithm based on Grover's search. In the same paper, Shi also introduced the Set Equality problem, which asks to decide whether two injective functions f, f' given via black-box oracles $O_f, O_{f'}$ have the same image or have disjoint images, given a promise that either is the case. Set Equality can be easily reduced to Index Erasure via the swap test, increasing the number of oracle queries by at most a constant factor; therefore, when Midrijānis presented an $\Omega((n/\log n)^{1/5})$ lower bound on the quantum query complexity of Set Equality [16], the same lower bound automatically applied to Index Erasure, ruling out the existence of poly-logarithmic query algorithms for these two problems.

Quantum state generation comes in two forms: the *coherent state generation*, where all memory aside from the target state must return to its initial state, $|\mathbf{0}\rangle := |0 \cdots 0\rangle$, and the *non-coherent state generation*, where there is no such a requirement, namely, where the ancillary memory can remain in some function-dependent state $|t_f\rangle$. Ambainis, Magnin, Roetteler, and Roland devised the *hybrid adversary method* [2], which they used to prove a tight $\Omega(\sqrt{n})$ lower bound for Index Erasure in the *coherent* regime, and left the non-coherent case as an open question. Later, the lower bound for Set Equality was improved to a tight $\Omega(n^{1/3})$ [26, 4], which in turn led to an improved query lower bound for the non-coherent Index Erasure.

In this paper, we close the gap for the non-coherent Index Erasure problem, by proving a tight lower bound on its quantum query complexity under the condition that the range of the black-box function f is sufficiently large. More formally, we show the following.

▶ **Theorem 1** (Main Result). *The bounded-error quantum query complexity of* Index Erasure *is* $\Theta(\sqrt{n})$ *in the non-coherent state generation regime, provided that* $m \geq n^{3\sqrt{n}}$.

To the best of our knowledge, the proof of Theorem 1 is the first application of the general adversary method of Lee, Mittal, Reichardt, Špalek, and Szegedy [15] for a non-coherent state generation problem. We outline the proof below.

1.1 Outline of the Proof of the Main Result

The symmetries of Index Erasure are paramount in our proof. The product $S_n \times S_m$ of two symmetric groups act on a function $f\colon [n] \to [m]$ as $(\pi, \rho)\colon f \mapsto \rho * f * \pi^{-1}$, where $(\pi, \rho) \in S_n \times S_m$ and $*$ denotes the composition of functions. This group action on injective functions defines a representation of $S_n \times S_m$. This representation is multiplicity-free, meaning that it contains no more than one instance of any irrep (irreducible representation) of $S_n \times S_m$. Moreover, it consists of those and only those irreps $\lambda \otimes \lambda'$ where the Young diagram $\lambda \vdash n$ is contained in the Young diagram $\lambda \vdash m$ and the skew shape λ'/λ has no more than one cell per column. Throughout the paper, we often abuse the terminology and we interchangeably use the terms partition λ of n, denoted $\lambda \vdash n$, the Specht module corresponding to λ (which is irreducible and distinct for every λ), and the n-cell Young diagram corresponding to λ.

Two types of irreps are of particular interest to us. Given $\lambda \vdash n$, we call the irrep $\lambda \otimes \bar{\lambda}$ where $\bar{\lambda} \vdash m$ is obtained from λ by adding $m-n$ cells to the first row of λ a *minimal* irrep and the irrep $\lambda \otimes (m-n, \lambda)$ where $(m-n, \lambda) \vdash m$ is obtained from λ by adding one cell to each of the first $m-n$ columns of λ (alternatively, by adding the part $m-n$ to λ) a *maximal* irrep, where we assume $m \geq 2n$. In other words, if $\theta \vdash k$ and $\lambda := (n-k, \theta) \vdash n$, then $(n-k, \theta) \otimes (m-k, \theta)$ is a minimal irrep and $(n-k, \theta) \otimes (m-n, n-k, \theta)$ is a maximal irrep. In particular, to lower bound the quantum query complexity of the non-coherent Index Erasure, we use essentially the same adversary matrix Γ as [2] used for the coherent Index Erasure, which is specified through minimal irreps.

An adversary matrix is a symmetric real matrix whose rows and columns are labeled by all the functions in the domain of the problem, and it is the central object of most adversary methods. In our case, the adversary matrix acts on the same $m^{\underline{n}}$-dimensional space as the representation matrices of $S_n \times S_m$ mentioned above, where $m^{\underline{n}} := m!/(m-n)!$ is the total number of functions. Similarly to [2], we choose

$$\Gamma := \sum\nolimits_{k=0}^{\sqrt{n}} \left(\sqrt{n} - k\right) \sum_{\theta \vdash k} E_{(n-k,\theta)\otimes(m-k,\theta)},$$

where $E_{\lambda\otimes\lambda'}$ is the orthogonal projector on the irrep $\lambda \otimes \lambda'$ (note that we have only used projectors on certain minimal irreps to construct Γ). We also note that the Gram matrices $T_{\lambda\otimes\lambda'} = m^{\underline{n}} E_{\lambda\otimes\lambda'}/d_{\lambda\otimes\lambda'}$, where $d_{\lambda\otimes\lambda'} := \operatorname{tr}[E_{\lambda\otimes\lambda'}]$ is the dimension of $\lambda \otimes \lambda'$, play an important role in our proof.

In order to take advantage of the inherent symmetries of the Index Erasure problem, we first extend the *automorphism principle* of Høyer, Lee, and Špalek [13] to the *general adversary method* for state generation and conversion problems [15] (see Corollary 3 and Theorem 4). This extension leads us to consider the Gram matrix corresponding to the final state $|\psi_f, t_f\rangle$ of an algorithm run with oracle O_f (assuming no error). The Gram matrix corresponding to $|\psi_f\rangle$ is

$$\frac{n}{m} T_{(n)\otimes(m)} + \left(1 - \frac{n}{m}\right) T_{(n)\otimes(m-1,1)} =: T^{\odot},$$

therefore the Gram matrix corresponding to $|\psi_f, t_f\rangle$ is $T^{\odot} \circ T$, where $T_{f,f'} := \langle t_f | t_{f'} \rangle$ and $\circ$ denotes the Schur (i.e., entrywise) matrix product. For the coherent regime lower bound, $\langle \mathbf{0} | \mathbf{0} \rangle = 1$ and $T = J = T_{(n)\otimes(m)}$ is the all-ones matrix. For the non-coherent regime, one of the consequences of the generalization of the automorphism principle is that it suffices to consider T such that $T_{f,f'} = T_{\sigma(f),\sigma(f')}$ for all functions f, f' and all $\sigma \in S_n \times S_m$.

To prove the $\Omega(\sqrt{n})$ lower bound, we must show, for all such Gram matrices T, that

$$\operatorname{tr}\left[\Pi_\Gamma \frac{T^{\odot} \circ T}{m^{\underline{n}}}\right] = o(1), \tag{1.1}$$

where Π_Γ is the orthogonal projector on the image of Γ, and that $\|\Gamma \circ \Delta_x\| = \mathrm{O}(1)$ for all $x \in [n]$, where Δ_x is the binary matrix with $(\Delta_x)_{f,f'} := 1$ if and only if $f(x) \neq f'(x)$.[1] Here we only need to prove the former condition because we use essentially the same adversary matrix as [2], and the latter condition is shown in their work. On the other hand, showing condition (1.1) was a triviality in [2] because $T = J$ in the coherent regime and thus the trace evaluates to n/m.

We now present the three main simplifying steps used to narrow the scope of condition (1.1). First, we use linearity to show that it suffices to prove

$$\operatorname{tr}\left[\Pi_\Gamma \frac{T_{(n)\otimes(m-1,1)} \circ T_{\lambda\otimes\lambda'}}{m^{\underline{n}}}\right] = o(1)$$

for all irreps $\lambda \otimes \lambda'$. That is, we can restrict our attention from a continuum of choices for T to a finite set $\{T_{\lambda\otimes\lambda'}\}_{\lambda\otimes\lambda'}$ of choices, where we have also used that the term $T_{(n)\otimes(m-1,1)}$ "dominates" $T_{(n)\otimes(m)}$ in $T^{\odot}$.

Second, we use the connection between $T_{(n)\otimes(m-1,1)}$ and a specific primitive idempotent of the Johnson (association) scheme to obtain

$$\operatorname{tr}\left[E_{\lambda\otimes\bar{\lambda}} \frac{T_{(n)\otimes(m-1,1)} \circ T_{\lambda\otimes\lambda'}}{m^{\underline{n}}}\right] = o(1)$$

as a sufficient condition, where we have to consider only Young diagrams $\lambda \vdash n$ that have less than $\sqrt{n}$ cells below the first row.

Third, for such λ, we show that the dimension of $\lambda\otimes\bar{\lambda}$ is much smaller than the dimension of any other $\lambda \otimes \lambda'$ (thus the nomenclature "minimal irrep"); therefore, we show it suffices to prove

$$\operatorname{tr}\left[E_{\lambda\otimes\bar{\lambda}} \frac{T_{(n)\otimes(m-1,1)} \circ T_{\lambda\otimes\bar{\lambda}}}{m^{\underline{n}}}\right] = o(1). \tag{1.2}$$

It is convenient to think of (1.1) and its simplifications in terms of the following association scheme. For a pair of functions (f, f'), consider the orbit $\mathcal{O}_\mu := \{(\sigma(f), \sigma(f')) \colon \sigma \in S_n \times S_m\}$, and let A_μ be the binary matrix with $(A_\mu)_{h,h'} = 1$ if and only if $(h, h') \in \mathcal{O}_\mu$. Here we use μ to label distinct orbits and let $\mathcal{C}_n$ be the set of all of them. The set of matrices $\{A_\mu \colon \mu \in \mathcal{C}_n\}$ forms a symmetric association scheme, denoted $\mathcal{A}_{n,m}$, which we call the *partial permutation scheme* due to the bijection $f \leftrightarrow (f(1), f(2), \cdots, f(n))$ between injective functions $f \colon [n] \to [m]$ and n-partial permutations of $[m]$.

In the terminology of (commutative) association schemes, the projectors $E_{\lambda\otimes\lambda'}$ are called the *primitive idempotents*, and their entries corresponding to the orbit $\mathcal{O}_\mu$ multiplied by $m^{\underline{n}}$ are called *dual eigenvalues* of the association scheme, which we denote as $q_{\lambda\otimes\lambda'}(\mu)$. The *valency* $v_\mu^{(m)}$ is the size of $\mathcal{O}_\mu$ divided by $m^{\underline{n}}$, thus, in terms of dual eigenvalues, the left hand side of condition (1.2) can be written as

$$\frac{\sum_{\mu\in\mathcal{C}_n} v_\mu^{(m)} \cdot q_{(n)\otimes(m-1,1)}(\mu) \cdot q^2_{\lambda\otimes\bar{\lambda}}(\mu)}{m^{\underline{n}} d_{(n)\otimes(m-1,1)} d_{\lambda\otimes\bar{\lambda}}}. \tag{1.3}$$

[1] The terms in condition (1.1) and similar expressions are written in such a way to emphasize that $\frac{T^{\odot}\circ T}{m^{\underline{n}}}$ is a density operator.

One of the crucial results of our paper relates the dual eigenvalues corresponding to a minimal irrep $(n-k,\theta)\otimes(m-k,\theta)$ in the $\mathcal{A}_{n,m}$ scheme to the dual eigenvalues corresponding to the maximal irrep $\theta\otimes(n-k,\theta)$ in the $\mathcal{A}_{k,n}$ scheme, where $\theta\vdash k$. This relation allows us to express the terms under the sum in (1.3) as polynomials in m. The condition $m\geq n^{3\sqrt{n}}$ in Theorem 1 ensures that it suffices to restrict the sum to those $\mu\in\mathcal{C}_n$ whose corresponding polynomials are of maximum degree, and for such μ, we relate their valencies $v_\mu^{(m)}$ in the $\mathcal{A}_{n,m}$ scheme to valencies of the $\mathcal{A}_{k,n}$ scheme, which together with certain properties of dual eigenvalues allow us to obtain the desired bound.

1.2 Additional Results on the Partial Permutation Scheme

In the context of quantum query complexity, the partial permutation association scheme was already considered in [21], where a conjecture on its eigenvalues implied tight adversary bounds for the Collision and Set Equality problems. Along these lines, our work shows a connection between quantum query complexity and the *Krein parameters* $q_{i,j}(k)$ of association schemes (see Section 5 for a formal definition). Indeed, condition (1.2) is equivalent to the conditions

$$q_{\lambda\otimes\bar{\lambda},\,\lambda\otimes\bar{\lambda}}((n)\otimes(m-1,1))=o(d_{\lambda\otimes\bar{\lambda}})\quad\text{and}\quad q_{\lambda\otimes\bar{\lambda},\,(n)\otimes(m-1,1)}(\lambda\otimes\bar{\lambda})=o(m)$$

on the Krein parameters of $\mathcal{A}_{n,m}$, and (1.3) gives an expression of these parameters in terms of dual eigenvalues.

The Krein parameters of an association scheme are important because they are the *dual structure constants* of its corresponding *Bose-Mesner algebra*. While the structure constants of Bose-Mesner algebras admit an obvious combinatorial meaning, its dual structure constants do not (e.g., they can be irrational) and are difficult to interpret. Indeed, the question of whether or not there exists a "good" interpretation of these constants has been asked often in algebraic combinatorics, so it seems interesting that we are able to relate these constants to the quantum query complexity of suitably symmetric state generations problems in the non-coherent regime. We are unaware of any previously known connection between quantum computing and the Krein parameters of association schemes.

Aside from computer science, we believe that the partial permutation scheme is of independent interest in combinatorics, as it generalizes both the Johnson scheme and the conjugacy class scheme of S_n. We include other new results on the partial permutation association scheme in the appendix, including a procedure that facilitates the calculation of dual eigenvalues corresponding to maximal irreps when $m\gg n$. We are unaware of any previously known results on the dual eigenvalues or Krein parameters of the partial permutation association scheme.

1.3 Organization of the paper

The paper is organized as follows. In Section 2, we present preliminaries on the quantum query model, with emphasis on state generation problems, including Index Erasure, the general adversary method, and the automorphism principle. In Section 3, we present preliminaries on the representation theory, particularly focusing on the symmetric group and its action on partial permutations. The automorphism principle of the general adversary method requires us to analyze highly symmetric matrices, which are elements of the Bose–Mesner algebra corresponding to the partial permutation scheme. In Section 4, we formally define this association scheme, establishing the labeling of various its parameters and computing some of them, as well as addressing its connection to the Johnson scheme. With this formalism at

our disposal, in Section 5, we show that the proof for $\Omega(\sqrt{n})$ lower bound on the quantum query complexity of the non-coherent INDEX ERASURE can be reduced to upper bounds on certain Krein parameter of the partial permutation scheme. Finally, we place the required bounds on these Krein parameters in Section 6. We defer some proofs and additional results on the partial permutation scheme to the appendix.

2 Quantum state generation

In this paper, we address limitations of quantum query algorithms for solving the INDEX ERASURE problem. We assume that the reader is familiar with foundations of quantum computing (see [18] for an introductory reference), some of which we review here. The basic memory unit of a quantum computer is a qubit, which is a two-dimensional complex Euclidean space $\mathbb{C}[\{0,1\}]$ having *computational* orthonormal basis $\{|0\rangle, |1\rangle\}$. Similarly, a k-qubit system corresponds to Euclidean space $\mathbb{C}[\{0,1\}^k]$ and it has computational basis $\{|b\rangle : b \in \{0,1\}^n\}$. Unit vectors $|\Psi\rangle \in \mathbb{C}[\{0,1\}^k]$ are called (pure) *quantum states* and they represent superpositions over various computations basis states.

Quantum bits are often grouped together in *registers* for the ease of algorithm design and analysis. If $|\psi\rangle, |\phi\rangle$ are states of two registers, then the state of the joint system is $|\psi\rangle \otimes |\phi\rangle$. We often shorten the notation $|\psi\rangle \otimes |\phi\rangle$ to $|\psi\rangle |\phi\rangle$ or $|\psi, \phi\rangle$. Due to *entanglement*, not always the state of the joint system can be written as a tensor product of states of the individual registers.

Quantum information is processed by unitary transformations, which correspond to square matrices U such that $UU^* = U^*U = I$, and they map quantum states to quantum states. This unitary processing of quantum information implies that any (noiseless) quantum computation is reversible.

2.1 Quantum query model

In the oracle model, we are given an access to a black-box oracle O_f that evaluates some unknown function $f\colon [n] \to [m]$. The goal of a query algorithm is to perform some computational task that depends on f, for example, to compute some function of f, such as $\textsc{Parity}(f) := f(1) \oplus f(2) \oplus \cdots \oplus f(n)$ when $m = 2$. In quantum computing, one can query the oracle in superposition. On the other hand, due to the requirement for reversibility, the oracle is typically designed so that it preserves the input query x. Namely, given $|x, y\rangle$ as an input, the oracle O_f outputs $|k, y \oplus f(x)\rangle$ (see Figure 1). Here and below we may assume $x, y, f(x)$ to be represented in binary. Even if f is injective – as it is for INDEX ERASURE – unless one knows how to compute the inverse of f, implementing $|x\rangle \mapsto |f(x)\rangle$ in practice might be much harder than $|x, y\rangle \mapsto |k, y \oplus f(x)\rangle$.

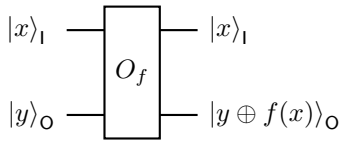


Figure 1 A schematic of a quantum oracle O_f. We assume that y and $f(x)$ are encoded in binary, and thus O_f is its own inverse.

A quantum query algorithm with oracle O_f consists of

- four registers: the input and output registers I and O for accessing the black-box function f, the target register T for storing the result of the computation, and an additional workspace register W;
- an indexed sequence of unitary transformations $U_0, U_1, \cdots, U_Q$ acting on those four registers.

The quantum query algorithm starts its computation in state $|\mathbf{0}\rangle := |00\cdots0\rangle$, and then performs $2Q+1$ unitary operations, alternating between U_i, which acts on all the registers, and O_f, which acts on registers IO. Thus the final state of the computation is

$$|\Psi_f\rangle := U_Q(O_f \otimes I_{\mathsf{TW}})U_{Q-1}(O_f \otimes I_{\mathsf{TW}})\cdots U_1(O_f \otimes I_{\mathsf{TW}})U_0\,|\mathbf{0}\rangle\,,$$

where I_{TW} is the identity operator on registers TW. Figure 2 gives a schematic of a quantum query algorithm. Note that Q is the number of oracle queries performed by the algorithm, and we also refer to it as the *query complexity of the algorithm*.

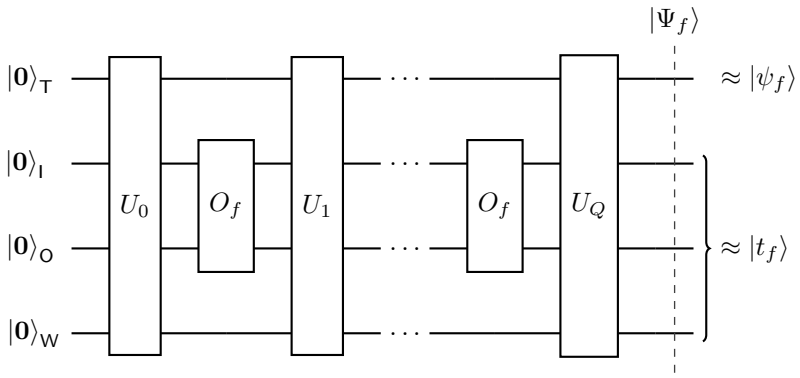


Figure 2 A schematic of a quantum algorithm that uses an oracle O_f. The registers labeled $\mathsf{T}, \mathsf{I}, \mathsf{O}, \mathsf{W}$ are, respectively, the target, input, output, and workspace registers of the algorithm. The target register of the final state $|\Psi_f\rangle$ of the algorithm should be in a state close to the target state $|\psi_f\rangle$.

In this paper we are interested in quantum query algorithms whose goal is to generate a specific f-dependent state $|\psi_f\rangle$ by accessing f via O_f. We note that this generalizes classical function evaluation by a quantum algorithm, where each $|\psi_f\rangle$ is asked to be a computational basis vector. In the next section we describe two distinct regimes of quantum state generation, as well as why they are exactly the same for classical function evaluation.

2.2 Coherent vs. Non-coherent State Generation

When we talk about quantum state generation with oracle O_f, we implicitly assume the domain $[n]$ and the range $[m]$ of f to be fixed. A *quantum state generation* problem is thus specified by a subset $\mathcal{F}$ of functions in form $f\colon [n] \to [m]$, which we call the *domain of the problem*, a complex Euclidean space called the *target space*, and, for every $f \in \mathcal{F}$, a quantum state $|\psi_f\rangle$ in the target space called the *target state*.

One may consider quantum state generation in two regimes: coherent and non-coherent. In the *coherent* state generation regime, all the computational memory other than the target register (i.e., registers IOW) must be returned to its initial state $|\mathbf{0}\rangle$. Therefore, if one was running an algorithm for a superposition of oracles, the final quantum state would be a

superposition of the target states. In contrast, for *non-coherent* state generation, one does not place any requirements on the ancillary memory. More precisely, in the coherent case, for every input $f \in \mathcal{F}$ we require that the final state $|\Psi_f\rangle$ satisfies

$$\Re\langle\psi_f, \mathbf{0}|\Psi_f\rangle \geq \sqrt{1-\epsilon},$$

where $|\mathbf{0}\rangle$ is the initial state of the ancillary registers and a constant $\epsilon \geq 0$ is the desired precision [15] and $\Re$ denotes the real part of the number it precedes. We call the minimum among quantum query complexities among quantum query algorithms that achieve this task the *(ϵ-error) quantum query complexity of the coherent version of the problem*. On the other hand, in the non-coherent case, the final state $|\Psi_f\rangle$ has to satisfy

$$\|(\langle\psi_f| \otimes I)|\Psi_f\rangle\| = \max_{|t_f\rangle} \Re\langle\psi_f, t_f|\Psi_f\rangle \geq \sqrt{1-\epsilon},$$

where the maximum is over unit vectors $|t_f\rangle$ on the system of registers IOW [15], and we analogously define the quantum query complexity of the non-coherent version of the problem.

It is worth noting that evaluation of classical functions can be considered as a special case of quantum state generation, where one is asked to prepare the computational basis state $|\psi_f\rangle$. Since quantum mechanics permits cloning of orthogonal states (computational basis states, in this case), there is no difference between coherent and non-coherent function evaluation, if one is willing to tolerate a two-fold increase in query complexity: at the end of a non-coherent computation, one can copy the target register into an additional register, and then run the whole computation in reverse, restoring all but this additional register to their initial state.

Also, note that an algorithm for the coherent case of a problem solves its non-coherent case as well. Looking on it from the other side: a lower bound on the non-coherent version of the problem is a lower bound on the coherent version as well.

2.3 Index Erasure

The domain of Index Erasure is the set of all injective functions $f\colon [n] \to [m]$. These functions are in one-to-one correspondence with n-partial permutations of $[m]$ and thus $|\mathcal{F}| = m^{\underline{n}} := m!/(m-n)!$. Index Erasure is the task of preparing the quantum state that is the uniform superposition

$$|\psi_f\rangle := \frac{1}{\sqrt{n}} \sum_{x=1}^{n} |f(x)\rangle$$

over the image of f. Note that the state

$$\frac{1}{\sqrt{n}} \sum_{x=1}^{n} |x\rangle|f(x)\rangle$$

can be prepared using a single query to O_f. This would give us the superposition that we seek if we could only ignore or *erase* the first register that records the *index* x, which gives the problem its namesake.

The question of the complexity of Index Erasure was first raised by Shi [23]. As for the upper bound, there is a simple quantum query algorithm for coherent Index Erasure given access to O_f. Thinking of the injective function f as a database with entries in $[m]$, for any $y \in [m]$ we may use Grover's algorithm with O_f to find the unique index x of f such that $f(x) = y$. In other words, there is a circuit that sends the superposition

$$\frac{1}{\sqrt{n}}\sum_{x=1}^{n}|f(x)\rangle \quad \text{to} \quad \frac{1}{\sqrt{n}}\sum_{x=1}^{n}|x\rangle|f(x)\rangle.$$

Inverting this circuit effectively "erases" the index register, which implies that the quantum query complexity of Index Erasure is $\mathrm{O}(\sqrt{n})$.

The first non-trivial lower bounds on the quantum query complexity of Index Erasure were obtained via the Set Equality problem, which asks to decide whether two injective functions f, f' given via black-box oracles $O_f, O_{f'}$ have the same image or have disjoint images, given a promise that either is the case. Set Equality can be easily reduced to non-coherent (and, thus, coherent too) Index Erasure via the swap test, increasing the number of oracle queries by at most a constant factor. Thus, when Midrijānis presented an $\Omega((n/\log n)^{1/5})$ lower bound on the quantum query complexity of Set Equality [16], the same lower bound automatically applied to Index Erasure. Ambainis, Magnin, Roetteler, and Roland devised the *hybrid adversary method* [2], which they used to prove a tight $\Omega(\sqrt{n})$ lower bound for Index Erasure in the *coherent* regime, and left the non-coherent case as an open question. Later, the lower bound for Set Equality was improved to a tight $\Omega(n^{1/3})$ [26, 4], which in turn led to an improved query lower bound for the non-coherent Index Erasure.

The focus of this work is to prove a tight lower bound on the quantum query complexity of Index Erasure in the *non-coherent* case. To show this, we use the so-called *general adversary method* [15] which we review in Section 2.4.

Finally, we note that if one were able to solve Index Erasure using poly-logarithmic number of queries, one would obtain a time-efficient algorithm for Graph Isomorphism. In Appendix A we refer to two similar tests for Graph Isomorphism based on Index Erasure, one on the coherent version, one on the non-coherent version of the problem. However, Midrijānis' lower bound on Set Equality ruled out efficiency of such tests.

2.4 General Adversary Method

The general adversary method places optimal lower bounds on the quantum query complexity of any state conversion problem [15]. State conversion problems generalize state generation problems, yet in this paper it will suffice to introduce the adversary bound only for the latter.

The general adversary bound is stated via the γ_2 and filtered γ_2 norms, which are defined as follows. Let M be any matrix and let $\Delta = \{\Delta_x \colon x \in [n]\}$ be a family of matrices of the same dimensions as M. Define

$$\begin{aligned}\gamma_2(M) &:= \max_{\Gamma'}\{\|M \circ \Gamma'\| \colon \|\Gamma'\| \le 1\},\\ \gamma_2(M|\Delta) &:= \max_{\Gamma}\left\{\|M \circ \Gamma\| \colon \max_{x\in[n]}\|\Delta_x \circ \Gamma\| \le 1\right\},\end{aligned}$$

where $\circ$ denotes the Schur (i.e., entrywise) product of two matrices and, thus, Γ and Γ' are required to have the same dimensions as M. One can show that $\gamma_2(\cdot)$ is a norm over the set of all matrices and $\gamma_2(\cdot|\Delta)$ is a norm over the set of matrices M that has $M_{f,f'} = 0$ whenever $(\Delta_x)_{f,f'} = 0$ for all $x \in [n]$ (see [15] for details). The two norms are called the γ_2 *norm* and the *filtered* γ_2 *norm*, respectively.

The general adversary bound employs various real symmetric matrices whose rows and columns are labeled by black-box functions $f \in \mathcal{F}$ in the same order. The family of *difference matrices* Δ is defined as follows. For each $x \in [n]$, the Δ_x is a binary matrix such that

$(\Delta_x)_{f,f'} := 1$ if and only if $f(x) \neq f'(x)$. A *state matrix* is any positive-semidefinite matrix T such that $T \circ I = I$. In other words, it is a Gram matrix corresponding to some family of unit vectors. Note that $\gamma_2(\cdot|\Delta)$ is a norm on the set of matrices whose diagonals are all-zeros, and a difference of any two state matrices belongs to this set.

Let $\mathcal{T}$ be the set of all state matrices. Note that $\mathcal{T}$ is a compact set and it is closed under the Schur product. Two particular state matrices of our interest are the all ones matrix J, which corresponds to the family $\{|\mathbf{0}\rangle : f \in \mathcal{F}\}$, and the *target matrix* $T^{\odot}$ defined as $(T^{\odot})_{f,f'} := \langle\psi_f|\psi_{f'}\rangle$.

Theorem 2 is a special case of [15, Theorem 4.9].

▶ **Theorem 2.** *The ϵ-error quantum query complexity of a non-coherent state generation problem with the target matrix $T^{\odot}$ and the family of difference matrices Δ is both*

$$\Omega\big(\mathsf{Adv}_{2\sqrt{2\epsilon}}\big) \quad \textit{and} \quad \mathrm{O}\big(\mathsf{Adv}_{\epsilon^4/16}\,\epsilon^{-2}\log\epsilon^{-1}\big),$$

where

$$\mathsf{Adv}_{\delta} := \min_{R,T\in\mathcal{T}}\{\gamma_2(J-R|\Delta)\colon \gamma_2(R-T^{\odot}\circ T)\leq\delta\}. \tag{2.1}$$

In the case of coherent state generation, one imposes $T = J$ in the expression for Adv_{δ}.

In the expression for Adv_{δ}, the state matrix T essentially corresponds to the ancillary states that are prepared in addition to the target states. Thus, assuming there were no error, $T^{\odot} \circ T$ would be the Gram matrix corresponding to the final states of the whole system. However, since one allows some error – determined by the parameter δ – it suffices that the state matrix R corresponding *exactly* to the final states of the algorithm is close to $T^{\odot} \circ T$.

When applying the adversary bound, it is convenient to actually apply it to the zero-error case therefore eliminating the matrix R from the consideration. In particular, this leads to the following corollary of Theorem 2.

A symmetric matrix Γ that satisfies $\|\Delta_x \circ \Gamma\| \leq 1$ for all x is called an *adversary matrix*. Let Π_{Γ} denote the orthogonal projector on the image of Γ.

▶ **Corollary 3.** *Let Γ be an adversary matrix for a non-coherent state generation problem with the target matrix $T^{\odot}$ and the family of difference matrices Δ, let ω be a principal eigenvector of Γ of norm 1, and let*

$$\eta' := \max_{T\in\mathcal{T}} \omega^{\top}(T^{\odot}\circ T\circ\Gamma/\|\Gamma\|)\,\omega.$$

The ϵ-error quantum query complexity of the problem is

$$\Omega\big((1-\eta'-2\sqrt{2\epsilon})\,\|\Gamma\|\,\big).$$

If ω is a uniform superposition over $\mathcal{F}$, then $\eta' \leq \eta$ for

$$\eta := \max_{T\in\mathcal{T}} \mathrm{tr}\left[\Pi_{\Gamma}(T^{\odot}\circ T)/|\mathcal{F}|\right].$$

Proof. For the first part of the corollary, suppose $R, T \in \mathcal{T}$ satisfy $\gamma_2(R - T^{\odot} \circ T) \leq 2\sqrt{2\epsilon}$ and are thus a feasible solution to the minimization in $\mathsf{Adv}_{2\sqrt{2\epsilon}}$. We have

$$\begin{aligned}
\gamma_2(J-R|\Delta) \geq& \|(J-R)\circ\Gamma\| \\
\geq& \|(J-T^{\odot}\circ T)\circ\Gamma\| - \|\Gamma\|\big\|(R-T^{\odot}\circ T)\circ\Gamma/\|\Gamma\|\big\| \\
\geq& \,\omega^{\top}\Gamma\,\omega - \omega^{\top}(T^{\odot}\circ T\circ\Gamma)\,\omega - 2\sqrt{2\epsilon}\|\Gamma\| \\
\geq& (1-\eta'-2\sqrt{2\epsilon})\|\Gamma\|.
\end{aligned}$$

For the second part, note that, if ω is a uniform superposition over $\mathcal{F}$, then, for any two symmetric $|\mathcal{F}| \times |\mathcal{F}|$ matrices M, M', we have $\omega^\top (M \circ M')\, \omega = \operatorname{tr}[MM']/|\mathcal{F}|$. The inequality $\eta' \leq \eta$ results from both $T^\odot \circ T$ and $\Pi_\Gamma - \Gamma/\|\Gamma\|$ being positive-semidefinite. ◀

2.5 Automorphism Principle for State Generation

The *automorphism principle* of [13] addresses the adversary bound for function evaluation problems and states that, without loss of generality, the optimal adversary matrix can be required to respect symmetries of the problem. Here we generalize the automorphism principle to state generation problems.[2]

The wreath product $S_m \wr S_n$ of groups S_m and S_n is the group whose elements are $(\pi, \boldsymbol{\sigma}) \in S_n \times S_m^n$ and whose group operation is

$$\big(\pi', (\sigma'_1, \cdots, \sigma'_n)\big)\big(\pi, (\sigma_1, \cdots, \sigma_n)\big) = \big(\pi'\pi, (\sigma'_1 \sigma_{(\pi')^{-1}(1)}, \cdots, \sigma'_n \sigma_{(\pi')^{-1}(n)})\big)$$

(see [14, Ch. 4]). Similarly to (3.1) below, the action of $S_m \wr S_n$ on $f\colon [n] \to [m]$ is given by

$$\big((\pi, \boldsymbol{\sigma}) f\big)(x) = \sigma_x(f(\pi^{-1}(x)) \qquad \text{for all } x \in [n]. \tag{2.2}$$

The action of a subgroup $G \leq S_m \wr S_n$ on the set of black-box functions $\mathcal{F}$ is *closed* if $g(f) \in \mathcal{F}$ for all $f \in \mathcal{F}$ and $g \in G$.

Suppose M is a symmetric $|\mathcal{F}| \times |\mathcal{F}|$ matrix whose rows and columns are labeled by $f \in \mathcal{F}$ in the same order and suppose the action of a subgroup $G \leq S_m \wr S_n$ on $\mathcal{F}$ is closed. We say that M is *G-invariant* if $M_{g(f),g(f')} = M_{f,f'}$ for all $f, f' \in \mathcal{F}$ and $g \in G$. Similarly, a vector $\omega \in \mathbb{C}[\mathcal{F}]$ is G-invariant if $\omega_{g(f)} = \omega_f$ for all $f \in \mathcal{F}$ and $g \in G$. A subgroup G is an *automorphism group* for a state generation problem with a target matrix $T^\odot$ if G's action on $\mathcal{F}$ is closed and $T^\odot$ is G-invariant.[3]

Note that the free product of two automorphism groups is an automorphism group, so one can consider the maximum automorphism group of a problem. For example, the maximum automorphism group of Parity is the whole wreath product $S_2 \wr S_n$ while the maximum automorphism groups of Or and Index Erasure are, respectively,

$$\{(\pi, (\varepsilon, \cdots, \varepsilon))\colon \pi \in S_n\} \cong S_n,$$
$$\{(\pi, (\sigma, \cdots, \sigma))\colon \pi \in S_n \,\&\, \sigma \in S_m\} \cong S_n \times S_m,$$

where ε is the identity permutation in S_2.

▶ **Theorem 4.** *Let G be an automorphism group for a non-coherent state generation problem. The value of Adv_δ remains the same if one restricts the minimization in the expression defining Adv_δ and the maximization in the expressions defining the γ_2 and filtered γ_2 norms to R, T, Γ, Γ' that are all G-invariant and imposes that $(J - R) \circ \Gamma$ has an G-invariant principal eigenvector.*

The proof of Theorem 4 considers two type of symmetrizations of matrices R, T, Γ, Γ', depending on whether they are arguments in a minimization or a maximization. We defer the proof to Appendix D.

[2] One can easily see that the automorphism principle generalizes even further to state conversion problems.
[3] The G-invariance of $T^\odot$ is equivalent to the existence of a unitary representation U_g of G acting on the target space such that $U_g|\psi_f\rangle = |\psi_{g(f)}\rangle$ for all $f \in \mathcal{F}$ and $g \in G$.

Note that the ability to restrict T and Γ to be G-invariant carries over from Theorem 2 to Corollary 3. The ability to restrict T will be paramount in our proof (see Section 5). On the other hand, the ability to restrict Γ is optional. Namely, Corollary 3 provides an adversary bound regardless of what restrictions one imposes on Γ, yet for too strict restrictions this bound would not be optimal.

For Index Erasure, it is convenient to think of every black-box function $f \in \mathcal{F}$ as a n-partial permutation over the set of symbols $[m]$. As observed in [2], the set of $|\mathcal{F}| \times |\mathcal{F}|$ matrices indexed by $\mathcal{F}$ that are $(S_n \times S_m)$-invariant under the aforementioned action (2.2) afford a *commutative matrix algebra*. In particular, it is the Bose–Mesner algebra of a symmetric association scheme defined over partial permutations, which we formally define in Section 4.

3 Representation Theory Preliminaries

Our main result builds upon finite group representation theory, especially that of the symmetric group. We refer the reader to [6] for a more thorough introduction to group representation theory and [22] for more details on the representation theory of the symmetric group. Throughout this section, let $H, K \leq G$ be subgroups of a finite group G, let $\mathcal{V}$ be a finite-dimensional vector space over $\mathbb{C}$, and for any set X, let $\mathbb{C}[X]$ denote the vector space of dimension $|X|$ of complex-valued functions over X.

3.1 The Representation Theory of the Symmetric Group

A *representation* $(\phi, \mathcal{V})$ of a finite group G is a homomorphism $\phi : G \to GL(\mathcal{V})$ where $GL(\mathcal{V})$ is the *general linear group*, that is, the group of $(\dim \mathcal{V}) \times (\dim \mathcal{V})$ invertible matrices. It is customary to be less formal and denote the representation $(\phi, \mathcal{V})$ simply as ϕ when $\mathcal{V}$ is understood, or as $\mathcal{V}$ when ϕ is understood. For any representation ϕ, we define its *dimension* to be $d_\phi := \dim \phi := \dim \mathcal{V}$. When working concretely with a representation ϕ, we abuse terminology and let $\phi(g)$ refer to a $(\dim \phi) \times (\dim \phi)$ matrix realization of ϕ. Two representations ρ, ϕ are *equivalent* if there exists a matrix P such that $\rho(g) = P^{-1}\phi(g)P$ for all $g \in G$.

Let $(\phi, \mathcal{V})$ be a representation of G, and let $\mathcal{W} \leq \mathcal{V}$ be a *G-invariant* subspace, that is, $\phi(g)w \in \mathcal{W}$ for all $w \in \mathcal{W}$ and for all $g \in G$. We say that $(\phi|_{\mathcal{W}}, \mathcal{W})$ is a *subrepresentation* of ϕ where $\phi|_{\mathcal{W}}$ is the restriction of ϕ to the subspace $\mathcal{W}$. A representation $(\phi, \mathcal{V})$ of G is an *irreducible representation* (or simply, a *G-irrep*) if it has no proper subrepresentations. The *trivial representation* $(1, \mathbb{C})$ defined such that $1 : g \to 1$ for all $g \in G$ is clearly an irrep of dimension one for any group G.

It is well-known that there is a bijection between the set of inequivalent G-irreps and its conjugacy classes $\mathcal{C}$, and that any representation $\mathcal{V}$ of G decomposes uniquely as a direct sum of inequivalent G-irreps

$$\mathcal{V} \cong \bigoplus_{i=1}^{|\mathcal{C}|} m_i \mathcal{V}_i$$

where m_i is the number of occurrences of the G-irrep $\mathcal{V}_i$ in the decomposition. We call the representation $m_i\mathcal{V}_i$ the *ith isotypic component* of $\mathcal{V}$.

A natural way to find representations of groups is to let them act on sets. In particular, for any group G acting on a set X, let $(\phi, \mathbb{C}[X])$ be the *permutation representation* of G on X defined such that

$$(\phi(g)[\zeta])(x) := \zeta(g^{-1}x)$$

for all $g \in G$, $\zeta \in \mathbb{C}[X]$, and $x \in X$. If we let G act on itself, then we obtain the *left regular representation*, which admits the following decomposition into G-irreps

$$\mathbb{C}[G] \cong \bigoplus_{i=1}^{|\mathcal{C}|} d_{\mathcal{V}_i} \mathcal{V}_i.$$

We denote the *Fourier transform* of $\gamma \in \mathbb{C}[G]$ with respect to the representation ϕ as

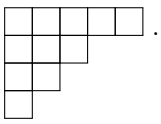
$$\phi(\gamma) := \sum_{g \in G} \gamma(g)\phi(g),$$

which is a linear operator on $\mathcal{V}$.

Let $\mathrm{Sym}(X)$ denote the *symmetric group* on the symbol set X. If $X = [m] := \{1, 2, \cdots, m\}$, then we define $S_m := \mathrm{Sym}(X)$. It is well-known that the conjugacy classes of S_m are given by the cycle-types of permutations of S_m, which in turn are in one-to-one correspondence with *integer partitions* $\lambda \vdash m$, i.e.,

$$\lambda := (\lambda_1, \lambda_2, \cdots, \lambda_k) \vdash m \text{ such that } \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_k \geq 0 \text{ and } \sum_{i=1}^{k} \lambda_i = m.$$

We may visualize λ as a *Young diagram*, a left-justified table of cells that contains λ_i cells in the ith row. When referencing a Young diagram, we alias λ as the *shape*. For example, the Young diagram below has shape $(5, 3, 2, 1) \vdash 11$:



A *standard Young tableau* of shape λ is a Young diagram with unique entries from $[n]$ that are strictly increasing along rows and strictly increasing along columns, e.g.,

1	2	5	8	9
3	6	7		
4	10			
11				

We may write the left regular representation of S_m as

$$\mathbb{C}[S_m] \cong \bigoplus_{\lambda \vdash m} d_\lambda \mathcal{V}_\lambda.$$

where d_λ is the number of standard Young tableau of shape λ, which can be counted elegantly via the *hook rule* (see [22] for a proof).

▶ **Theorem 5** (The Hook Rule). *Let $\lambda \vdash m$, and for any cell $c \in \lambda$ of the Young diagram of λ define the* hook-length $\mathsf{h}_\lambda(c)$ *to be the total number of cells below c in the same column and to the right of c in the same row, plus 1. Then we have*

$$d_\lambda = \frac{m!}{\mathsf{H}(\lambda)} \quad \textit{where} \quad \mathsf{H}(\lambda) := \prod_{c \in \lambda} \mathsf{h}_\lambda(c).$$

Another well-known result is the *branching rule*, which describes how an S_m-irrep decomposes into (S_{m-1})-irreps (see [22] for a proof). We say that a cell of a Young diagram is an *inner corner* if it has no cells to its right and no cells below it.

▶ **Theorem 6** (The Branching Rule). *If $\mathcal{V}_\lambda$ is a S_m-irrep, then*

$$\mathcal{V}_\lambda \cong \bigoplus_{\lambda^-} \mathcal{V}_{\lambda^-}$$

where λ^- ranges over all shapes obtainable by removing an inner corner from λ and $\mathcal{V}_{\lambda^-}$ is the corresponding (S_{m-1})-irrep.

The hook rule and the branching rule can be used to prove the following theorem. We defer its proof to Appendix D.

▶ **Theorem 7.** *Let $\theta \vdash k$ and $\theta^+ \vdash (k+1)$ be any shape obtained by adding an inner corner to θ. For all $m \geq 2(k+1)$, we have*

$$\frac{d_{(m-k-1,\theta^+)}}{d_{(m-k,\theta)}} \geq \frac{m}{k} \cdot \left(1 - \frac{2k+1}{m}\right).$$

Note that the above fraction is greater than 1 when $m > 3k+1$. For any $\lambda = (\lambda_1, \lambda_2, \cdots, \lambda_\ell) \vdash n$, henceforth, let $\bar{\lambda} := (\lambda_1 + (m-n), \lambda_2, \cdots, \lambda_\ell) \vdash m$. Theorem 7 has the following corollary.

▶ **Corollary 8.** *Let $\lambda \vdash n$ be a shape such that $\lambda_1 \geq n - \sqrt{n}$ and let $\lambda' \vdash m$ be any shape that covers λ such that λ'/λ is a horizontal strip. Then $d_{\lambda'}/d_{\bar{\lambda}} \in \Omega(m/\sqrt{n})$.*

3.2 The Representation Theory of Partial Permutations

Let $S_{n,m}$ denote the collection of *n-partial permutations of* $[m]$, that is, n-tuples

$$f := (f(1), f(2), \cdots, f(n))$$

with no repeated elements such that $f(x) \in [m]$ for all $x \in [n]$. Injective functions from $[n]$ to $[m]$ are in one-to-one correspondence with $S_{n,m}$, and it is not hard to see that $|S_{n,m}| = m^{\underline{n}}$, and when $m = n$ we recover the symmetric group S_n on n symbols. To understand the representation theory of $S_{n,m}$ we must first broaden our Young tableau vocabulary.

For any $\lambda \vdash m$, let $l(\lambda)$ denote the *length* of λ, that is, the number of parts in the partition. We say that a shape λ *covers* a shape μ if $\mu_i \leq \lambda_i$ for each i. If λ and μ are two shapes such that λ covers μ, then we obtain the *skew shape* λ/μ by removing the cells corresponding to μ from λ. For instance, the shape $(5,3,2,1)$ covers $(2,2,1)$, so we may consider the skew shape $(5,3,2,1)/(2,2,1)$:

A skew shape is a *horizontal strip* if each column has no more than one cell. For example, the skew shape $(5,3,2,1)/(3,3,1)$ is a horizontal strip, but the skew shape above is not.

Henceforth, we let $S_n \times S_m$ act on $S_{n,m}$ as follows:

$$(\tau, \sigma) \cdot (f_1, \cdots, f_n) = (\sigma^{-1}(f_{\tau^{-1}(1)}), \cdots, \sigma^{-1}(f_{\tau^{-1}(n)})) \text{ for all } (\tau, \sigma) \in S_n \times S_m. \tag{3.1}$$

The stabilizer of the *identity n-partial permutaton* $f_{\text{id}} := (1, 2, \cdots, n) \in S_{n,m}$ in $S_n \times S_m$ is isomorphic to the group

$$\text{diag}(S_n) \times S_{m-n} = \{(\tau, \tau, \pi) : \tau \in \text{Sym}([n]), \pi \in \text{Sym}(\{n+1, \cdots, m\}).$$

Using Pieri's rule (see [24]), one can show that the permutation representation of $(S_n \times S_m)$ acting on $S_{n,m} \cong (S_n \times S_m)/(\text{diag}(S_n) \times S_{m-n})$ is *multiplicity-free*, that is, its decomposition has at most one copy of any $(S_n \times S_m)$-irrep, as shown in Theorem 9.

▶ **Theorem 9** ([5]). *The complex-valued functions over n-partial permutations $\mathbb{C}[S_{n,m}]$ admits the following decomposition into $(S_n \times S_m)$-irreps:*

$$\mathbb{C}[S_{n,m}] \cong \bigoplus_{\mu,\lambda} \mathcal{V}_\mu \otimes \mathcal{V}_\lambda$$

where μ, λ ranges over all pairs $\mu \vdash n, \lambda \vdash m$ such that λ/μ is a horizontal strip.

Let $\mathrm{Irr}(S_{n,m})$ denote the set of $(S_n \times S_m)$-irreps that appear in Theorem 9. Every multiplicity-free permutation representation gives rise to a commutative association scheme (see [3]), so a consequence of Theorem 9 is the existence a symmetric association scheme $\mathcal{A}_{n,m}$ over $S_{n,m}$ that we call *the partial permutation association scheme*. In Section 4, we discuss this association scheme in more detail.

A coarser decomposition of $\mathbb{C}[S_{n,m}]$ into irreducibles of S_m can be obtained by identifying $S_{n,m}$ with the set of *tabloids* (i.e., Young tableaux with unordered rows) of shape $(m-n, 1^n)$ and applying Young's rule (see [22]). This representation is known as the $(m-n, 1^n)$-*permutation representation*, which we denote as $\mathcal{M}^{(m-n,1^n)}$. Its isotypic components can be determined combinatorially via the *Kostka numbers* $K_{\lambda,\mu}$ (see [22]).

▶ **Theorem 10.** *The complex-valued functions over n-partial permutations $\mathbb{C}[S_{n,m}]$ admits the following decomposition into S_m-irreps:*

$$\mathbb{C}[S_{n,m}] \cong \mathcal{M}^{(m-n,1^n)} \cong \bigoplus_{\lambda \vdash m} K_{\lambda,(m-n,1^n)}\ \mathcal{V}_\lambda.$$

Note that Theorem 9 gives a multiplicity-free orthogonal decomposition of the λ-isotypic component $K_{\lambda,(m-n,1^n)}\mathcal{V}_\lambda$ of $\mathcal{M}^{(m-n,1^n)}$ into $(S_n \times S_m)$-irreps

$$K_{\lambda,(m-n,1^n)}\mathcal{V}_\lambda \cong \bigoplus_{\mu} \mathcal{V}_\mu \otimes \mathcal{V}_\lambda$$

where the sum ranges over all $\mu \vdash n$ such that λ/μ is a horizontal strip.

The following two types of irreps in $\mathrm{Irr}(S_{n,m})$ will be of particular importance.

▶ **Definition 11** (Minimal and Maximal Irreps). *For any $\lambda \vdash n$, the* minimal irrep *and* maximal irrep *(w.r.t. λ) is $\lambda \otimes \bar{\lambda} \in Irr(S_{n,m})$ and $\lambda \otimes (m-n, \lambda) \in Irr(S_{n,m})$ respectively.*

If $\lambda_1 \geq n - \sqrt{n}$, Theorem 7 implies that the minimal and maximial irreps w.r.t. λ are indeed of the least and largest dimension over all irreps of the form $\lambda \otimes \mu \in \mathrm{Irr}(S_{n,m})$ for sufficiently large m.

For $\lambda \vdash m$, let E_λ be the orthogonal projector onto the λ-isotypic component $K_{\lambda,(m-n,1^n)}\mathcal{V}_\lambda$ of $\mathcal{M}^{(m-n,1^n)}$, which we may write as

$$E_\lambda = \frac{d_\lambda}{m!}\mathcal{M}^{(m-n,1^n)}(\overline{\chi_\lambda})$$

where $\mathcal{M}^{(m-n,1^n)}(\overline{\chi}_\lambda) = \sum_{\sigma \in S_m} \chi_\lambda(\sigma^{-1})\mathcal{M}^{(m-n,1^n)}(\sigma)$ is the Fourier transform of the function $\overline{\chi}_\lambda \in \mathbb{C}[S_m]$ with respect to the permutation representation of S_m acting on $S_{n,m}$. In particular, for any $\zeta \in \mathbb{C}[S_{n,m}]$ and $f \in S_{n,m}$, we have

$$[E_\lambda \zeta](f) = \frac{d_\lambda}{m!} \sum_{\sigma \in S_m} \chi_\lambda(\sigma)\zeta(\sigma^{-1} f)$$

using the well-known fact that $\chi_\lambda(\sigma^{-1}) = \chi_\lambda(\sigma)$ for any $\sigma \in S_m$, $\lambda \vdash m$. From our foregoing discussion, we also have that

$$E_\lambda E_{\mu\otimes\lambda} = E_{\mu\otimes\lambda}$$

where $E_{\mu\otimes\lambda}$ is the orthogonal projector onto $(\mathcal{V}_\mu \otimes \mathcal{V}_\lambda)$ for all $(\mu \otimes \lambda) \in \mathrm{Irr}(S_{n,m})$.

For any integer $k \geq 0$, one may think of the following as a "low-frequency" subspace of $\mathbb{C}[S_{n,m}]$ parameterized by k:

$$\mathcal{U}_k := \bigoplus_{\substack{(\mu\otimes\lambda)\in\mathrm{Irr}(S_{n,m}) \\ m-\lambda_1\leq k}} \mathcal{V}_\mu \otimes \mathcal{V}_\lambda \cong \bigoplus_{\substack{\lambda\vdash m \\ m-\lambda_1\leq k}} K_{\lambda,(m-n,1^n)}\mathcal{V}_\lambda.$$

Equivalently, we have

$$\mathcal{U}_k \cong \{\zeta \in \mathbb{C}[S_{n,m}] : \lambda(\zeta) = 0 \text{ for all } \lambda \vdash m \text{ such that } m - \lambda_1 > k\}$$

where we have identified $\mathbb{C}[S_{n,m}]$ with the space of functions $\zeta \in \mathbb{C}[S_m]$ that are constant on the cosets S_m/S_{m-n}. Let $\alpha = \{(x_1, \alpha(x_1)), (x_2, \alpha(x_2)), \cdots, (x_k, \alpha(x_k))\}$, be an injective function from $[n]$ to $[m]$, which we represent as a set of k ordered pairs, and define

$$S_\alpha := \{f \in S_{n,m} : f(x_j) = \alpha(x_j) \text{ for all } 1 \leq j \leq k\}.$$

Theorem 12 shows that the characteristic functions of S_α have "low Fourier-complexity", namely, they are supported on the "low" Fourier levels of $\mathbb{C}[S_{n,m}]$ (i.e., $\mathrm{Irr}(S_{n,m})$), which are in reverse-lexicographic order on the partitions $\lambda' \vdash m$ corresponding to their S_m-irrep. One can compare these functions to the so-called *k-juntas* in area of Boolean functions, as their output is determined by examining no more than k "coordinates" of its input [19]. Such junta generalizations have been fundamental to some recent developments in extremal combinatorics (see [8, 7] for example). The proof of Theorem 12 resembles [8, Theorem 7], which we defer to Appendix D.

▶ **Theorem 12.** *Let $1_\alpha \in \mathbb{C}[S_{n,m}]$ be the characteristic function of the family S_α. Then $1_\alpha \in \mathcal{U}_k$.*

An immediate corollary of this theorem is the following.

▶ **Corollary 13.** *Let $1_\alpha \in \mathbb{C}[S_{n,m}]$ be the characteristic function of the family S_α. Then $E_{\mu\otimes\lambda}1_\alpha = 0$ for all λ with more than k cells below the first row.*

4 The Partial Permutation Association Scheme

The theory of association schemes will be a convenient language for describing the algebraic and combinatorial components of our work. We refer the reader to Bannai and Ito's reference [3] for a more thorough treatment.

▶ **Definition 14** (Association Schemes). *A* symmetric association scheme *is a collection of $d+1$ binary $|X| \times |X|$ matrices $\mathcal{A} = \{A_0, A_1, \cdots, A_d\}$ over a set X that satisfy the following axioms:*

1. *A_i is symmetric for all $0 \leq i \leq d$,*
2. *$A_0 = I$ where I is the identity matrix,*
3. *$\sum_{i=0}^{d} A_i = J$ where J is the all-ones matrix, and*
4. *$A_iA_j = A_jA_i \in Span\{A_0, A_1, \cdots, A_d\} =: \mathfrak{A}$ for all $0 \leq i, j \leq d$.*

The matrices $A_1, A_2, \cdots, A_d$ are called the associates*, and the algebra $\mathfrak{A}$ is called the* Bose–Mesner *algebra of the association scheme.*

Since the all-ones matrix commutes with every associate, the matrices of an association scheme have constant row sum and constant column sum. Also, since the matrices of a symmetric association scheme are symmetric and commute with each other, they are simultaneously diagonalizable (equivalently, they share a system of orthonormal eigenvectors), which implies that $\mathfrak{A}$ admits a unique basis of *primitive idempotents* $E_0, E_1, \cdots, E_d$, i.e., $E_i^2 = E_i$ for all $0 \leq i \leq d$ and $\sum_{i=0}^{d} E_i = I$.

Since the permutation representation of $S_n \times S_m$ acting on $S_{n,m}$ is multiplicity-free (see Theorem 9), the orbits $A_0, A_1, \cdots, A_d$ (so-called *orbitals*) of the action of $S_n \times S_m$ on ordered pairs $S_{n,m} \times S_{n,m}$ forms a *symmetric association scheme* (see [3] for a proof). We abuse the notation, and also use A_i to denote the binary matrix with entries 1 corresponding to exactly those pairs that are in the orbit A_i. Let $\mathcal{A}_{n,m} := \{I, A_1, \cdots, A_d\}$ denote the *n-partial permutation association scheme of* $[m]$.

Although it is well-known that permutation representation of $S_n \times S_m$ acting on $S_{n,m}$ is multiplicity-free (see [2, 5, 12] for example), the parameters of its corresponding association scheme $\mathcal{A}_{n,m}$ have not been worked out (to the best of our knowledge). We now give a more in-depth treatment of the partial permutation association scheme.

4.1 The Associates

The following is a more combinatorial definition of the associates of $\mathcal{A}_{n,m}$ that gives a combinatorial bijection between the associates of $\mathcal{A}_{n,m}$ and $\mathrm{Irr}(S_{n,m})$, which are the eigenspaces of the association scheme. The bijection is readily observed by thinking of each element of $S_{n,m}$ graphically as a maximum matching of the complete bipartite graph $K_{n,m}$ (see Figure 3).

Let $f_{\mathrm{id}} = (1, 2, \cdots, n)$ denote the *identity n-partial permutation*, which we can view as the maximum matching of $K_{n,m}$ that pairs 1 with 1, 2 with 2, and so on (e.g., the red matching in Figure 3). For any two maximum matchings f, f' of $K_{n,m}$, let $G(f, f')$ be the multigraph whose edge multiset is the multiset union $f \cup f'$. Clearly $G(f, f') = G(f', f)$ and this graph is composed of disjoint even cycles and disjoint even paths. Let c denote the number of disjoint cycles and let $2\lambda_i$ denote the length of an even cycle. Let p denote the number of disjoint paths and let $2\rho_i$ denote the length of an even path. If we order the cycles and paths respectively from longest to shortest and divide each of their lengths by two, assuming $m \geq 2n$, we see that the graphs $G(f, f')$ are in bijection (up to graph isomorphism) with pairs $(\lambda|\rho)$ of integer partitions $\lambda = (\lambda_1, \lambda_2, \cdots, \lambda_c), \rho = (\rho_1, \rho_2, \cdots, \rho_p)$ such that $(\lambda_1, \cdots, \lambda_c, \rho_1, \cdots, \rho_p) \vdash n$. Let $\mathrm{d}(f, f') := (\lambda|\rho)$ denote this bijection, which we refer to as the *cycle-path type of f' with respect to f*. Note that $\mathrm{d}(\sigma(f), \sigma(f')) = \mathrm{d}(f, f')$ for all n-partial permutations f, f' and all $\sigma \in S_n \times S_m$. If one of the arguments is the identity matching, then we say $\mathrm{d}(f) := \mathrm{d}(f_{\mathrm{id}}, f)$ is *the cycle-path type of f*. Illustrations of the graphs $G_{(\varnothing|n)}$ and $G_{(n-1|1)}$, and $G_{(\varnothing|1^n)}$ are provided in Figure 3 where $n = 3$ and $m = 6$.

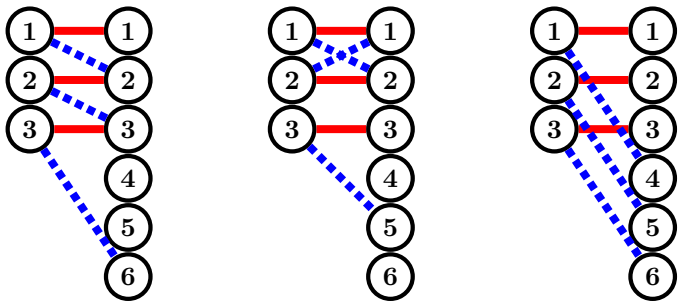
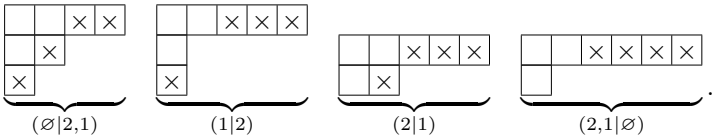


Figure 3 $(2, 3, 6)$ on the left has type $(\varnothing|3)$, $(2, 1, 5)$ has type $(2|1)$, and $(4, 5, 6)$ has type $(\varnothing|1^3)$.

Let $\mathcal{C}_n := \{(\lambda|\rho)\colon |\lambda| + |\rho| = n\}$ where λ and ρ are partitions. When $m \geq 2n$, $\mathcal{C}_n$ is the set of all cycle-path types. Note that $(\varnothing|1^n)$ is not a cycle-path type when $m < 2n$, and for $m = n$, all cycle-path types are of form $(\lambda|\varnothing)$, where $\lambda \vdash n$. We can decompose $\mathcal{C}_n$ as a disjoint union $\mathcal{C}_n = \bigcup_{k=0}^{n} \mathcal{C}_{n,k}$, where $\mathcal{C}_{n,k}$ consists of all $(\lambda|\rho) \in \mathcal{C}_n$ such that $l(\rho) = n - k$.

Recall that any irrep in $\mathrm{Irr}(S_{n,m})$ is of the form $\lambda \otimes \lambda'$ where λ'/λ is a horizontal strip of size $m - n$. To see that cycle-path types $(\tau|\rho)$ have a natural correspondence with these irreducibles, consider a Young diagram of λ' such that the cells of λ'/λ are marked. Every columns of λ in λ' with a marked cell below it corresponds to a part in ρ whereas an unmarked column correspond to a part in τ. For instance, taking $\lambda = (2, 1)$ and $m = 7$, we have



Note that marked singleton columns correspond to paths of length zero (i.e., isolated nodes). For each cycle-path type $(\tau|\rho)$, the $(\tau|\rho)$-*associate* of $\mathcal{A}_{n,m}$ is the following $m^{\underline{n}} \times m^{\underline{n}}$ binary matrix:

$$(A_{(\tau|\rho)})_{i,j} = \begin{cases} 1, & \text{if } \mathrm{d}(i,j) = (\tau|\rho) \\ 0, & \text{otherwise} \end{cases}$$

where $i, j \in S_{m,n}$.

4.2 The Valencies and Multiplicities

For each $0 \leq i \leq d$, let $d_i := \operatorname{tr} E_i$ denote the *multiplicity* of the ith eigenspace of an association scheme, that is, the dimension of its ith eigenspace. For each $0 \leq i \leq d$, define the *valency* v_i to be the row sum of an arbitrary row of A_i (equivalently, the largest eigenvalue of A_i). We now give formulas for the valencies $v^{(m)}_{(\lambda|\rho)}$ and multiplicities $d_{(\lambda|\rho)}$ of $\mathcal{A}_{n,m}$.[4]

For each $(\lambda|\rho)$, define the $(\lambda|\rho)$-*sphere* to be the following set:

$$\Omega_{(\lambda|\rho)} := \{f \in S_{n,m} : \mathrm{d}(f) = (\lambda|\rho)\}.$$

The spheres partition $S_{n,m}$ and it useful to think of them as conjugacy classes. Indeed, when $n = m$, these spheres are the conjugacy classes of S_m. Note that $v^{(m)}_{(\lambda|\rho)} = |\Omega_{(\lambda|\rho)}|$, and basic combinatorial reasoning reveals the following.

▶ **Proposition 15.** *For any cycle-path type $(\lambda|\rho)$, the size of the $(\lambda|\rho)$-sphere is*

$$v^{(m)}_{(\lambda|\rho)} = |\Omega_{(\lambda|\rho)}| = \frac{n!}{\prod_{i=1}^{n} i^{\ell_i} \ell_i! r_i!} (m-n)^{\underline{l(\rho)}}$$

where $\lambda = (1^{\ell_1}, \cdots, n^{\ell_n})$, $\rho = (1^{r_1}, \cdots, n^{r_n})$, and $l(\rho) = r_1 + \cdots + r_n$.

We omit the superscript (m) of the valency when m is clear from the context.

The multiplicities $d_{(\tau|\rho)}$ are easy to deduce due to the fact that each eigenspace of the scheme is isomorphic to an irrep $\mu \otimes \lambda$ of $S_n \times S_m$, and that $\dim \mu \otimes \lambda = \dim \mu \cdot \dim \lambda$. As we have seen, these dimensions are counted by the hook rule. In particular, for a cycle-path type $(\tau|\rho)$, let $\tau \cup \rho$ be the union of the set of parts of the two partitions. Then we have $d_{(\tau|\rho)} = d_{\lambda \otimes \lambda'}$ such that $\lambda = (\tau \cup \rho)^\top \vdash n$, $\lambda' = (\tau \cup (m - n, \rho^\top)^\top)^\top$, and '$\top$' denotes the transpose partition.

[4] The "m" in the superscript (m) of the valency simply indicates the size of the domain of any $f \in S_{n,m}$. It is a notational convenience that will make some of our proofs easier to follow later in the paper.

4.3 The Johnson Ordering of $\mathcal{A}_{n,m}$

The *Johnson scheme* $\mathcal{J}(m,n)$ is a symmetric association scheme defined over the n-subsets of $[m]$. The ith associate $A_i \in \mathcal{J}(m,n)$ of the Johnson scheme is defined such that $(A_i)_{X,Y} = 1$ if $n - |X \cap Y| = i$, and is 0 otherwise for any two n-subsets X, Y. It is well-known that the ith eigenspace of $\mathcal{J}(m,n)$ is isomorphic to the S_m-irrep associated to the partition $(m-i,i) \vdash m$. For proofs of these facts and more, see [10]. Henceforth, let E_i be the primitive idempotent of the Johnson scheme that projects onto $\mathcal{V}_{(m-i,i)}$.

There exists a natural ordering of the $S_{n,m}$ that we call *the Johnson ordering* that shows the Johnson scheme is a "quotient" of $\mathcal{A}_{n,m}$. First, we order $S_{n,m}$ first by the corresponding n-subsets (the particular order does not matter). Next, we lexicographically order all $n!$ n-partial permutations that correspond to the same n-subset (i.e., share the same image). For example, for $n = 3$, we could have:

$$(1,2,3),(1,3,2),(2,1,3),(2,3,1),(3,1,2),(3,2,1),$$
$$(1,2,4),(1,4,2),(2,1,4),(2,4,1),(4,1,2),(4,2,1),$$
$$(1,3,4),(1,4,3),\cdots$$

Both S_n and S_m act on the indices and entries of an n-partial permutation respectively, and so each of their actions correspond to some collection of $m^{\underline{n}} \times m^{\underline{n}}$ permutation matrices (i.e., their corresponding permutation representations). The action of S_m on $S_{m,n}$ is transitive, but S_n's action has $\binom{m}{n}$ orbits, one for each n-subset. Note that on all $n!$ permutations of S_n corresponding to any given n-subset, the action of S_n corresponds to the regular representation of S_n.

Given $\lambda \vdash n$ and $\lambda' \vdash m$, let E_λ and $E_{\lambda'}$ be the orthogonal projectors on the λ-isotypic and λ'-isotypic subspaces, respectively. Since the actions of S_n and S_m on $S_{n,m}$ commute, E_λ and $E_{\lambda'}$ also commute, and we have $E_{\lambda \otimes \lambda'} = E_\lambda E_{\lambda'}$. From the specific way we ordered $S_{n,m}$ in the previous paragraph, for any $\lambda \vdash n$ we have

$$E_\lambda = I_{\binom{m}{n}} \otimes F_\lambda = \underbrace{F_\lambda \oplus F_\lambda \oplus \cdots \oplus F_\lambda}_{\binom{m}{n} \text{ times}},$$

where F_λ is the $n! \times n!$ orthogonal projector on the λ-isotypic subspace of the regular representation of S_n. Hence, we can write $E_{\lambda \otimes \lambda'}$ as a product of two block matrices:

$$E_{\lambda \otimes \lambda'} = \begin{pmatrix} B^{\lambda'}_{1,1} & B^{\lambda'}_{1,2} & \cdots \\ B^{\lambda'}_{2,1} & B^{\lambda'}_{2,2} & \\ \vdots & & \ddots \end{pmatrix} \begin{pmatrix} F_\lambda & 0 & \cdots \\ 0 & F_\lambda & \\ \vdots & & \ddots \end{pmatrix} = \begin{pmatrix} B^{\lambda'}_{1,1} F_\lambda & B^{\lambda'}_{1,2} F_\lambda & \cdots \\ B^{\lambda'}_{2,1} F_\lambda & B^{\lambda'}_{2,2} F_\lambda & \\ \vdots & & \ddots \end{pmatrix},$$

where the first matrix is $E_{\lambda'}$, in which each block $B^{\lambda'}_{i,j}$ is some $n! \times n!$ matrix.

Note that $F_{(n)} = J/n!$, where J is the $n! \times n!$ all-ones matrix. Thus, from the expression above, we have

$$E_{(n)\otimes(m-1,1)} = J/n! \otimes E_1 = \begin{pmatrix} b_{1,1} J & b_{1,2} J & \cdots \\ b_{2,1} J & b_{2,2} J & \\ \vdots & & \ddots \end{pmatrix},$$

where $b_{i,j}$ are scalars, $1/m^{\underline{n}}$ times the dual eigenvalues described above. Thus we have

$$E_{(n)\otimes(m-1,1)} \circ E_{\lambda\otimes\lambda'} = \begin{pmatrix} b_{1,1}B^{\lambda'}_{1,1}F_\lambda & b_{1,2}B^{\lambda'}_{1,2}F_\lambda & \cdots \\ b_{2,1}B^{\lambda'}_{2,1}F_\lambda & b_{2,2}B^{\lambda'}_{2,2}F_\lambda & \\ \vdots & & \ddots \end{pmatrix}, \tag{4.1}$$

which is orthogonal to E_μ (and thus $E_{\mu\otimes\mu'}$) for all $\mu \vdash n$ such that $\mu \neq \lambda$.

4.4 The Dual Eigenvalues of $\mathcal{A}_{n,m}$

A classic result in the theory of association schemes is that the primitive idempotents can be written as a linear combination of associates weighted by the dual eigenvalues of the scheme (see [9, Ch. 2.1]). In our case, we have

$$E_{\lambda\otimes\lambda'} = \frac{1}{m^{\underline{n}}} \sum_{(\mu|\rho)\in\mathcal{C}_n} q_{\lambda\otimes\lambda'}(\mu|\rho) A_{(\mu|\rho)},$$

and these coefficients $q_{\lambda\otimes\lambda'}(\mu|\rho)$ are the *dual eigenvalues* of $\mathcal{A}_{m,n}$.

The dual eigenvalues corresponding to minimal and maximal irreps play a central role in the proof of our main result. Let us consider matrices in the Bose–Mesner algebra $\mathfrak{A}_{n,m}$ of the partial permutation association scheme. By symmetry, every such matrix can be specified by a row or column corresponding to a single n-partial permutation of $[m]$.

Note that

$$(E_{\lambda\otimes\lambda'})_{f,h} = \frac{q_{\lambda\otimes\lambda'}(\mathrm{d}(f,h))}{m^{\underline{n}}}$$

and that

$$\sum_{f\in S_{n,m}} q_{\lambda\otimes\lambda'}(\mathrm{d}(f,h))\ 1_f = m^{\underline{n}}\,(E_{\lambda\otimes\lambda'})_h \in \lambda\otimes\lambda'$$

for all $\lambda\otimes\lambda'$ and $f,h \in S_{n,m}$, where $1_f \in \mathbb{C}[S_{n,m}]$ denotes the binary unit vector with the unique 1 in position f. The projector $E_{\lambda'}$ onto the λ'-isotypic component can be written as

$$(E_{\lambda'})_{f,h} = \frac{d_{\lambda'}}{m!} \sum_{\sigma\in S_m} \chi_{\lambda'}(\sigma^{-1})(V_\sigma)_{f,h}.$$

where $V_\sigma : 1_f \mapsto 1_{\sigma * f}$ for all $f,h \in S_{n,m}$. The foregoing, and the fact that $E_{\lambda'}E_{\lambda\otimes\lambda'} = E_{\lambda\otimes\lambda'}$ implies the following proposition.

▶ **Proposition 16.** *For any $f,h \in S_{n,m}$, $\lambda \vdash n$, and $\lambda' \vdash m$, we have*

$$q_{\lambda\otimes\lambda'}\big(\mathrm{d}(f,h)\big) = \frac{d_{\lambda'}}{m!} \sum_{\sigma\in S_m} \chi_{\lambda'}(\sigma)\, q_{\lambda\otimes\lambda'}\big(\mathrm{d}(\sigma^{-1} * f, h)\big).$$

▶ **Lemma 17.** *Let $q_i(j)$ be a dual eigenvalue of the Johnson scheme $\mathcal{J}(m,n)$. Then we have*

$$q_1(j) = \frac{\binom{n}{m}}{\binom{m-2}{n-1}} \left(n - j - \frac{n^2}{m}\right).$$

Moreover, if $m \geq n^2$, then $q_i(j) \geq 0$ for all $j \neq n$.

Proof. Let $p_i(j)$ denote the j-th eigenvalue of the i-th associate of the Johnson scheme $\mathcal{J}(m,n)$. It is well-known (see [10] for example) that

$$p_i(j) = \sum_{r=i}^{n} (-1)^{(r-i+j)} \binom{r}{i} \binom{m-2r}{n-r} \binom{m-r-j}{r-j}.$$

Let P be the $(n+1) \times (n+1)$ matrix defined such that $P_{j,i} = p_i(j)$. As their name suggests, the dual eigenvalues $Q_{j,i} := q_i(j)$ are formally dual to the eigenvalues of the association scheme, that is, $Q = \binom{m}{n} P^{-1}$. Inverting P reveals that

$$Q_{j,1} = q_1(j) = \frac{\binom{m}{n}}{\binom{m-2}{n-1}} \left(n - j - \frac{n^2}{m} \right),$$

which is non-negative for all $m \geq n^2$ and $j \neq n$, which completes the proof. ◀

▶ **Lemma 18.** *For all $\mu \in \mathcal{C}_{n,k}$, we have*

$$q_{(n)\otimes(m-1,1)}(\mu) = \frac{(km - n^2)(m-1)}{n(m-n)}.$$

Proof. Recall that the ith eigenspace of the Johnson scheme $\mathcal{J}(m,n)$ is isomorphic to the S_m-irrep associated to the partition $(m-i,i) \vdash m$, and that E_i denotes the primitive idempotent of the Johnson scheme that projects onto its $(m-i,i)$ eigenspace. Since

$$E_{(n)\otimes(m-1,1)} = \frac{J}{n!} \otimes E_1,$$

Lemma 17 implies that

$$\begin{aligned} E_1 &= \frac{1}{\binom{n}{m}} \sum_{j=0}^{n} \left[\frac{\binom{n}{m}}{\binom{m-2}{n-1}} \left(n - j - \frac{n^2}{m} \right) \right] A_j \\ &= \frac{1}{\binom{m-2}{n-1}} \sum_{j=0}^{n} \left(n - j - \frac{n^2}{m} \right) A_j. \end{aligned}$$

Since $(A_j)_{X,Y} = 1$ only if $|X \cap Y| = n - j$, we have the dual eigenvalue

$$\left(m^n E_{(n)\otimes(m-1,1)}\right)_{f,h} = m^n \frac{|\mathrm{im}\ f \cap \mathrm{im}\ h| - n^2/m}{n!\binom{m-2}{n-1}} = \frac{(m-1)(|\mathrm{im}\ f \cap \mathrm{im}\ h| m - n^2)}{n(m-n)}.$$

It follows that

$$q_{(n)\otimes(m-1,1)}(\mu) = \frac{(km - n^2)(m-1)}{n(m-n)}$$

for all $\mu \in \mathcal{C}_{n,k}$, which completes the proof. ◀

5 A sufficient condition on Krein parameters

Recall that $\circ$ denotes the Schur (entrywise) product of two matrices.

▶ **Definition 19** (Krein Parameters). *Let $\mathcal{A}$ be an association scheme on v vertices with d associates. For any $0 \leq i,j \leq d$, there exist constants $q_{i,j}(k)$ such that*

$$E_i \circ E_j = \frac{1}{v} \sum_{k=0}^{d} q_{i,j}(k) E_k,$$

which are called the Krein parameters *of $\mathcal{A}$. More explicitly, we have*

$$q_{i,j}(k) = v\frac{\operatorname{tr}\left[E_k(E_i \circ E_j)\right]}{d_k}.$$

For more details on the Krein parameters of an association scheme, see [3]. The Krein parameters can alternatively be written as

$$q_{i,j}(k) = \frac{1}{vd_k}\sum_{\ell=0}^{d}\frac{q_i(\ell)q_j(\ell)\overline{q_k(\ell)}}{v_\ell} = \frac{d_i d_j}{v}\sum_{\ell=0}^{d}\frac{\overline{p_i(\ell)p_j(\ell)}p_k(\ell)}{v_\ell^2}, \tag{5.1}$$

where $p_i(j)$ denotes the j-th eigenvalue of A_i and $q_i(j)$ denotes the jth *dual eigenvalue* of E_i (see [9, Chap. 2.4] for a proof).

To prove the lower bound on non-coherent Index Erasure, we use the same adversary matrix Γ as [2] used for the coherent case.[5] For simplifying the equations, without loss of generality let us assume that n is a square. As in [2], we choose

$$\Gamma := \sum_{k=0}^{\sqrt{n}}(\sqrt{n}-k)\sum_{\lambda\vdash k}E_{(n-k,\lambda)\otimes(m-k,\lambda)},$$

and thus the orthogonal projection onto its image is

$$\Pi_\Gamma := \sum_{\lambda\colon|\lambda|<\sqrt{n}}E_{(n-|\lambda|,\lambda)\otimes(m-|\lambda|,\lambda)}.$$

Note that the sole principal eigenvector ω of Γ is the uniform superposition over $\mathcal{F}$ (i.e., $\omega_f = 1/\sqrt{m^{\underline{n}}}$ for all $f\in\mathcal{F}$). Thus, as per Corollary 3, we are interested in the quantity

$$\eta = \max_{T\in\mathcal{T}}\operatorname{tr}\left[\Pi_\Gamma\frac{(T\circ T^{\odot})}{m^{\underline{n}}}\right].$$

For any primitive idempotent $E_{\lambda\otimes\lambda'}$, let

$$T_{\lambda\otimes\lambda'} := \left(\frac{m^{\underline{n}}}{\operatorname{tr}E_{\lambda\otimes\lambda'}}\right)E_{\lambda\otimes\lambda'} = \left(\frac{m^{\underline{n}}}{d_{\lambda\otimes\lambda'}}\right)E_{\lambda\otimes\lambda'}$$

be its corresponding state matrix. In [2] it is shown that the target matrix can be written as

$$T^{\odot} = \frac{n}{m}T_{(n)\otimes(m)} + \left(1-\frac{n}{m}\right)T_{(n)\otimes(m-1,1)}.$$

In the coherent case, recall that $T = J$, and therefore

$$\eta = \operatorname{tr}\left[\Pi_\Gamma\frac{T^{\odot}}{m^{\underline{n}}}\right] = \frac{n}{m}\operatorname{tr}\left[\frac{T_{(n)\otimes(m)}}{m^{\underline{n}}}\right] = \frac{n}{m}.$$

The most technically involved part of the proof of the lower bound by [2] is proving that $\|\Delta_x\circ\Gamma\| = \mathrm{O}(1)$. Since we are using the same adversary matrix Γ, we already have the above bound on $\|\Delta_x\circ\Gamma\|$. Our goal is to show that $\operatorname{tr}\left[\Pi_\Gamma(T\circ T^{\odot})/m^{\underline{n}}\right]$ is small for *all* state matrices T.

[5] Technically, the adversary matrix used here is $\sqrt{n}$ times that of [2] as the adversary method they use places slightly different conditions on the adversary matrix.

By dividing the elements in the set $\mathcal{T}$ by $m^{\underline{n}}$ we obtain the set of all density matrices (positive-semidefinite Hermitian matrices with trace 1) of the Bose–Mesner algebra $\mathfrak{A}_{n,m}$. Note that $(T \circ T')/m^{\underline{n}}$ is a density matrix for all $T, T' \in \mathcal{T}$.

For any $T \in \mathcal{T}$, we have

$$\operatorname{tr}\left[\Pi_\Gamma \frac{\big(T \circ T_{(n)\otimes(m)}\big)}{m^{\underline{n}}}\right] = \operatorname{tr}\left[\Pi_\Gamma \frac{T}{m^{\underline{n}}}\right] \leq 1,$$

therefore

$$\operatorname{tr}\left[\Pi_\Gamma \frac{\big(T \circ T^{\odot}\big)}{m^{\underline{n}}}\right] \leq \frac{n}{m} + \left(1 - \frac{n}{m}\right) \operatorname{tr}\left[\Pi_\Gamma \frac{\big(T \circ T_{(n)\otimes(m-1,1)}\big)}{m^{\underline{n}}}\right].$$

Our goal is to bound the latter term:

$$\left(1 - \frac{n}{m}\right) \operatorname{tr}\left[\Pi_\Gamma \frac{\big(T \circ T_{(n)\otimes(m-1,1)}\big)}{m^{\underline{n}}}\right]. \tag{5.2}$$

First note that

$$\mathcal{T} = \left\{\sum_\chi c_\lambda T_\chi \colon \sum_\chi c_\chi = 1 \text{ and } c_\chi \geq 0\right\},$$

where the sums range over $\chi \in \operatorname{Irr}(S_{n,m})$. Hence,

$$\begin{aligned}\max_{T\in\mathcal{T}} \operatorname{tr}\left[\Pi_\Gamma \frac{\big(T \circ T_{(n)\otimes(m-1,1)}\big)}{m^{\underline{n}}}\right] &= \max_{\substack{\{c_\chi \geq 0\}_\chi \\ \sum_\chi c_\chi = 1}} c_\chi \operatorname{tr}\left[\Pi_\Gamma \frac{\big(T_\chi \circ T_{(n)\otimes(m-1,1)}\big)}{m^{\underline{n}}}\right] \\ &= \max_\chi \operatorname{tr}\left[\Pi_\Gamma \frac{\big(T_\chi \circ T_{(n)\otimes(m-1,1)}\big)}{m^{\underline{n}}}\right].\end{aligned}$$

The following proposition simplifies (5.2).

▶ **Proposition 20.** *For any $\lambda = (n - |\nu|, \nu)$ and $\bar{\lambda} = (m - |\nu|, \nu)$, we have*

$$\operatorname{tr}\left[\Pi_\Gamma \frac{\big(T_{\lambda\otimes\lambda'} \circ T_{(n)\otimes(m-1,1)}\big)}{m^{\underline{n}}}\right] = \operatorname{tr}\left[E_{\lambda\otimes\bar{\lambda}} \frac{\big(T_{\lambda\otimes\lambda'} \circ T_{(n)\otimes(m-1,1)}\big)}{m^{\underline{n}}}\right].$$

Proof. By Equation (4.1), if $|\nu| < \sqrt{n}$, then we have

$$\begin{aligned}\Pi_\Gamma \frac{\big(T_{\lambda\otimes\lambda'} \circ T_{(n)\otimes(m-1,1)}\big)}{m^{\underline{n}}} &= \sum_{\mu\colon |\mu|<\sqrt{n}} E_{(n-|\mu|,\mu)\otimes(m-|\mu|,\mu)} \frac{\big(T_{\lambda\otimes\lambda'} \circ T_{(n)\otimes(m-1,1)}\big)}{m^{\underline{n}}} \\ &= E_{(n-|\nu|,\nu)\otimes(m-|\nu|,\nu)} \frac{\big(T_{\lambda\otimes\lambda'} \circ T_{(n)\otimes(m-1,1)}\big)}{m^{\underline{n}}} \\ &= E_{\lambda\otimes\bar{\lambda}} \frac{\big(T_{\lambda\otimes\lambda'} \circ T_{(n)\otimes(m-1,1)}\big)}{m^{\underline{n}}};\end{aligned}$$

otherwise, the left-hand and right-hand side are both 0, which completes the proof. ◀

The proposition above now allows us to bound (5.2) for all $\lambda \vdash n$ having no more than $\sqrt{n}$ cells under its first row.

▶ **Corollary 21.** *Suppose $\lambda \vdash n$ has no more than $\sqrt{n}$ cells below the first row. Then*

$$\operatorname{tr}\left[E_{\lambda\otimes\bar{\lambda}}\frac{\left(T_{(n)\otimes(m-1,1)}\circ T_{\lambda\otimes\lambda'}\right)}{m^{\underline{n}}}\right] \in \mathrm{O}(\sqrt{n}/m)$$

for all $\lambda' \neq \bar{\lambda}$.

Proof. Let $\operatorname{sum}[\cdot]$ denote the sum of the entries of the matrix. We have

$$\begin{aligned}\operatorname{tr}\left[E_{\lambda\otimes\bar{\lambda}}\frac{\left(T_{(n)\otimes(m-1,1)}\circ T_{\lambda\otimes\lambda'}\right)}{m^{\underline{n}}}\right] &= \frac{1}{d_{\lambda\otimes\lambda'}}\operatorname{sum}\left[E_{\lambda\otimes\bar{\lambda}}\circ T_{(n)\otimes(m-1,1)}\circ E_{\lambda\otimes\lambda'}\right]\\ &= \frac{d_{\lambda\otimes\bar{\lambda}}}{d_{\lambda\otimes\lambda'}}\operatorname{tr}\left[E_{\lambda\otimes\lambda'}\frac{\left(T_{(n)\otimes(m-1,1)}\circ T_{\lambda\otimes\bar{\lambda}}\right)}{m^{\underline{n}}}\right]\end{aligned}$$

Since $\left(T_{(n)\otimes(m-1,1)}\circ T_{\lambda\otimes\bar{\lambda}}\right)/m^{\underline{n}}$ is a density matrix, the trace on the right-hand side is at most 1. We now have

$$\leq \frac{d_{\bar{\lambda}}}{d_{\lambda'}} \in \mathrm{O}(\sqrt{n}/m),$$

where the asymptotic bound follows from Corollary 8, completing the proof. ◀

We therefore have

$$\eta = \mathrm{O}\left(\frac{n}{m} + \operatorname{tr}\left[E_{\lambda\otimes\bar{\lambda}}\frac{\left(T_{\lambda\otimes\bar{\lambda}}\circ T_{(n)\otimes(m-1,1)}\right)}{m^{\underline{n}}}\right]\right),$$

and it remains to bound the value

$$\begin{aligned}\operatorname{tr}\left[E_{\lambda\otimes\bar{\lambda}}\frac{\left(T_{\lambda\otimes\bar{\lambda}}\circ T_{(n)\otimes(m-1,1)}\right)}{m^{\underline{n}}}\right] &= \frac{m^{\underline{n}}}{(m-1)d_{\lambda\otimes\bar{\lambda}}}\operatorname{sum}\left[E_{(n)\otimes(m-1,1)}\circ E_{\lambda\otimes\bar{\lambda}}\circ E_{\lambda\otimes\bar{\lambda}}\right]\\ &= \frac{\sum_{\mu\in\mathcal{C}_n} v_\mu^{(m)}\cdot q_{(n)\otimes(m-1,1)}(\mu)\cdot q^2_{\lambda\otimes\bar{\lambda}}(\mu)}{m^{\underline{n}}(m-1)d_{\lambda\otimes\bar{\lambda}}}. \qquad (5.3)\end{aligned}$$

In the next section we show that, under the assumption that $m \geq n^{3\sqrt{n}}$, the value of (5.3), and thus η, is in $\mathrm{O}(1/\sqrt{n})$.

Note that, according to the expression (5.1) for Krein parameters, (5.3) is equal to

$$\frac{q_{\lambda\otimes\bar{\lambda},\,\lambda\otimes\bar{\lambda}}((n)\otimes(m-1,1))}{d_{\lambda\otimes\bar{\lambda}}} = \frac{q_{\lambda\otimes\bar{\lambda},\,(n)\otimes(m-1,1)}(\lambda\otimes\bar{\lambda})}{m-1},$$

and therefore the task of bounding (5.3) is the task of bounding Krein parameters

$$q_{\lambda\otimes\bar{\lambda},\,\lambda\otimes\bar{\lambda}}((n)\otimes(m-1,1)) \quad\text{and}\quad q_{\lambda\otimes\bar{\lambda},\,(n)\otimes(m-1,1)}(\lambda\otimes\bar{\lambda}).$$

6 Bounding relevant Krein parameters

Let $(\mathfrak{A}_{n,m})_{f_{\mathrm{id}}}$ denote the space of the columns of matrices in the Bose–Mesner algebra $\mathfrak{A}_{n,m}$ corresponding to f_{id}. For brevity, we call such columns *characteristic columns*. Note that, due to symmetry, $\phi \in (\mathfrak{A}_{n,m})_{f_{\mathrm{id}}}$ is the characteristic column of exactly one matrix in

$\mathfrak{A}_{n,m}$, in effect defining one-to-one correspondence between $(\mathfrak{A}_{n,m})_{f_{\mathrm{id}}}$ and $\mathfrak{A}_{n,m}$. Since the characteristic column of any primitive idempotent $E_{\lambda\otimes\lambda'}$ is an eigenvector of the $(\lambda\otimes\lambda')$-eigenspace of $\mathcal{A}_{n,m}$, we have $\dim[(\mathfrak{A}_{n,m})_{f_{\mathrm{id}}} \cap (\lambda\otimes\lambda')] \geq 1$. Since the characteristic columns of $\mathcal{A}_{n,m}$ form an orthogonal basis for $(\mathfrak{A}_{n,m})_{f_{\mathrm{id}}}$, we have that $\dim(\mathfrak{A}_{n,m})_{f_{\mathrm{id}}} = |\mathcal{C}_n|$. These facts imply the following proposition.

▶ **Proposition 22.** *For any $\lambda\otimes\lambda' \in \mathrm{Irr}(S_{n,m})$, we have* $\dim[(\mathfrak{A}_{n,m})_{f_{\mathrm{id}}} \cap (\lambda\otimes\lambda')] = 1$.

We let $1_f \in \mathbb{C}[S_{n,m}]$ denote the binary unit vector with the unique 1 in position f. For $\mu \in \mathcal{C}_n$, let

$$1_\mu := \sum_{f\colon \mathrm{d}(f)=\mu} 1_f.$$

6.1 Assignments

We call an injective function $\alpha\colon D \to [n]$ with $D = \mathrm{dom}(\alpha) \subseteq [n]$ an *assignment*. Let $|\alpha| := |\mathrm{dom}(\alpha)| = |\mathrm{im}(\alpha)|$, which we call the *weight* of α. We say that $f \in S_{n,m}$ *agrees* with α if $f(x) = \alpha(x)$ for all $x \in \mathrm{dom}(\alpha)$, and we write $\alpha \rightsquigarrow f$; in other words, $f \in S_\alpha$ In particular, there are $(m-|\alpha|)^{\underline{n-|\alpha|}}$ partial permutations in $S_{n,m}$ that agree with α. Recall that

$$1_\alpha = \sum_{f\colon \alpha\rightsquigarrow f} 1_f.$$

For any assignment α and any permutation $\pi \in S_n$, define assignments α^{-1} and $\alpha * \pi^{-1}$ with domains $\mathrm{im}(\alpha)$ and $\pi(\mathrm{dom}(\alpha))$ respectively in the natural way. Note that the image of both α and $\alpha*\pi^{-1}$ is the same. Since the action of $\pi \in S_n$ maps $f \in S_{n,m}$ to $f*\pi^{-1}$, we have $\alpha \rightsquigarrow h$ if and only if $\pi(\alpha) \rightsquigarrow \pi(h)$. Hence, $V_\pi 1_\alpha = 1_{\pi(\alpha)}$, where $\pi \mapsto V_\pi$ is the representation of S_n defined as $V_\pi : 1_f \mapsto 1_{f*\pi^{-1}}$.

Consider $X := \{x_1, \cdots, x_k\} \subseteq [n]$ with $x_i < x_{i+1}$ for all i, and define $\alpha_X\colon [k] \to [n]$ such that $i \mapsto x_i$. Note that the set of all $\alpha^{-1} * \alpha_X$ such that $\mathrm{im}(\alpha) = X$ equals $S_{k,n}$, where we think of $S_{k,n}$ as the set of injective functions from $[k]$ to $[n]$. We define the cycle-path type of α to be

$$\mathrm{d}(\alpha) := \mathrm{d}(\alpha^{-1} * \alpha_X, \alpha_X) \in \mathcal{C}_k.$$

See Figure 4 for an illustration.

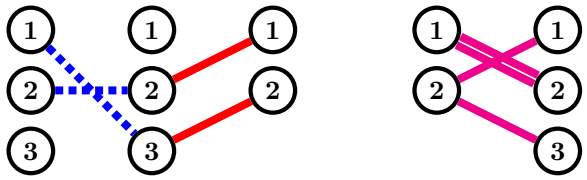


Figure 4 As an example, consider $n = 3$ and an assignment $\alpha\colon D \to [n]$ with $D = \{1,2\}$ defined as $\alpha(1) := 3$ and $\alpha(2) := 2$. The image of α is $X = \{2,3\}$, and thus α_X maps 2 to 1 and 3 to 2. The left picture depicts α with blue dashed lines and α_X^{-1} with red solid lines. The right picture depicts the multigraph resulting from the union of edges corresponding to $\alpha^{-1} * \alpha_X$ and α_X, from which we can see that $\mathrm{d}(\alpha) = (1|1)$.

Corollary 13 immediately implies the following proposition.

▶ **Proposition 23.** *For any assignment α and $\lambda' \vdash m$ such that $m - \lambda'_1 > |\alpha|$, we have $E_{\lambda'} 1_\alpha = 0$.*

6.2 Dual eigenvalues of minimal irreps

Suppose $\theta \vdash k$, and let us analyze the idempotent corresponding to the irrep $(n-k,\theta) \otimes (m-k,\theta)$. For a cycle-path type $\nu \in \mathcal{C}_k$, define

$$\phi_\nu := \sum_{\alpha\colon \mathrm{d}(\alpha)=\nu} 1_\alpha.$$

Note that $\phi_\nu \in (\mathfrak{A}_{n,m})_{f_{\mathrm{id}}}$. Consider the irrep

$$\xi_\theta := \theta \otimes (n-k,\theta) \in \mathrm{Irr}(S_k \times S_n).$$

We will be interested in its dual eigenvalues $q_{\xi_\theta}(\nu)$ for the $\mathcal{A}_{k,n}$ scheme.

▶ **Lemma 24.** *The characteristic column of $m^{\underline{n}} E_{(n-k,\theta)\otimes(m-k,\theta)}$ equals*

$$\phi_{\xi_\theta} := \frac{d_{(m-k,\theta)}}{\binom{n}{k} d_\theta} \sum_{\nu \in \mathcal{C}_k} q_{\xi_\theta}(\nu)\phi_\nu, \tag{6.1}$$

i.e., for any $\mu \in \mathcal{C}_n$ and $f \in S_{n,m}$ with $\mathrm{d}(f) = \mu$, we have $q_{(n-k,\theta)\otimes(m-k,\theta)}(\mu) = (\phi_{\xi_\theta})_f$.

Proof. First, note that $\phi_{\xi_\theta} \in (\mathfrak{A}_{n,m})_{f_{\mathrm{id}}}$ as $\phi_\nu \in (\mathfrak{A}_{n,m})_{f_{\mathrm{id}}}$ for all $\nu \in \mathcal{C}_k$. Recall that Proposition 22 implies that the intersection of $(\mathfrak{A}_{n,m})_{f_{\mathrm{id}}}$ and the $((n-k,\theta)\otimes(m-k,\theta))$-isotypic subspace is one-dimensional. Using this fact and the expression of the primitive idempotents in the $\mathcal{A}_{n,m}$ basis, to prove the first statement of the lemma, it suffices to show that

1. ϕ_{ξ_θ} belongs to the $(n-k,\theta)$-isotypic,
2. ϕ_{ξ_θ} belongs to the $(m-k,\theta)$-isotypic, and
3. that ϕ_{ξ_θ} has the right scaling $d_{(m-k,\theta)}/\binom{n}{k}d_\theta$.

Let us first prove statement (3). Any assignment α with $\mathrm{d}(\alpha) \neq (1^k|\varnothing)$ is incompatible with f_{id}, that is, $\alpha \not\rightsquigarrow f_{\mathrm{id}}$. By linearity, it follows that $(\phi_\nu)_{f_{\mathrm{id}}} = 0$ for $\nu \neq (1^k|\varnothing)$. On the other hand, there are $\binom{n}{k}$ assignments α with $\mathrm{d}(\alpha) = (1^k|\varnothing)$, and they all agree with f_{id}. We deduce that $(\phi_{(1^k|\varnothing)})_{f_{\mathrm{id}}} = \binom{n}{m}$, and

$$\begin{aligned}(\phi_{\xi_\theta})_{f_{\mathrm{id}}} = \frac{d_{(m-k,\theta)}}{d_\theta} q_{\xi_\theta}(1^k|\varnothing) &= d_{(n-k,\theta)} \cdot d_{(m-k,\theta)} \\ &= q_{(n-k,\theta)\otimes(m-k,\theta)}(1^n|\varnothing) \\ &= m^{\underline{n}} \left(E_{(n-k,\theta)\otimes(m-k,\theta)}\right)_{f_{\mathrm{id}},f_{\mathrm{id}}},\end{aligned}$$

where we have used the fact that $q_{\lambda\otimes\lambda'}(1^k|\varnothing)$ is the dimension of $(\lambda \otimes \lambda') \in \mathrm{Irr}(S_{k,n})$. This completes the proof of statement (3).

We now prove statement (1). Equation (6.1) can be written as

$$\phi_{\xi_\theta} = \frac{d_{(m-k,\theta)}}{\binom{n}{k} d_\theta} \sum_{\substack{X\subset[n]\\|X|=k}} \phi_{\xi_\theta,X},$$

where we define

$$\phi_{\xi_\theta,X} := \sum_{\nu\in\mathcal{C}_k} q_{\xi_\theta}(\nu) \sum_{\substack{\alpha\\ \mathrm{d}(\alpha)=\nu\\ \mathrm{im}(\alpha)=X}} 1_\alpha = \sum_{\substack{\alpha\\ \mathrm{im}(\alpha)=X}} q_{\xi_\theta}(\mathrm{d}(\alpha^{-1} * \alpha_X, \alpha_X)) 1_\alpha.$$

Hence,

$$\begin{aligned}
\left[E_{(n-k,\theta)}\right]\phi_{\xi_\theta,X} &= \sum_{\pi\in S_n}\sum_{\substack{\alpha\\ \mathrm{im}(\alpha)=X}} \frac{d_{(n-k,\theta)}\chi_{(n-k,\theta)}(\pi)}{n!} q_{\xi_\theta}(\mathrm{d}(\alpha^{-1}*\alpha_X,\alpha_X))1_{\alpha*\pi^{-1}}\\
&= \sum_{\pi\in S_n}\sum_{\substack{\tilde\alpha*\pi\\ \mathrm{im}(\tilde\alpha*\pi)=X}} \frac{d_{(n-k,\theta)}\chi_{(n-k,\theta)}(\pi)}{n!} q_{\xi_\theta}(\mathrm{d}((\tilde\alpha*\pi)^{-1}*\alpha_X,\alpha_X))1_{\tilde\alpha*\pi*\pi^{-1}}\\
&= \sum_{\substack{\tilde\alpha\\ \mathrm{im}(\tilde\alpha)=X}} \left(\sum_{\pi\in S_n} \frac{d_{(n-k,\theta)}\chi_{(n-k,\theta)}(\pi)}{n!} q_{\xi_\theta}(\mathrm{d}(\pi^{-1}*\tilde\alpha^{-1}*\alpha_X,\alpha_X))\right)1_{\tilde\alpha}\\
&= \sum_{\substack{\tilde\alpha\\ \mathrm{im}(\tilde\alpha)=X}} q_{\xi_\theta}(\mathrm{d}(\tilde\alpha^{-1}*\alpha_X,\alpha_X)1_{\tilde\alpha}\\
&= \phi_{\xi_\theta,X},
\end{aligned}$$

where the second to last equality follows from Proposition 16. By linearity, we deduce that $E_{(n-k,\theta)}\phi_{\xi_\theta} = \phi_{\xi_\theta}$, and thus ϕ_{ξ_θ} belongs to the $(n-k,\theta)$-isotypic subspace, completing the proof of statement (1).

Finally, we prove statement (2). Consider the irreps $\lambda\otimes\lambda' \in \mathrm{Irr}(S_{n,m})$ in the $(n-k,\theta)$-isotypic, which are of the form $(n-k,\theta)\otimes\lambda'$. If $\lambda' \neq (m-k,\theta)$, then λ' has more than k cells below its first row. But then Proposition 23 implies that ϕ_{ξ_θ} is orthogonal to all such $(n-k,\theta)\otimes\lambda'$ irreps, which finishes the proof of the first part of the lemma.

The second part of the lemma is a restatement of the first that is seen by expressing the primitive idempotents in the $\mathcal{A}_{n,m}$ basis, which completes the proof of the lemma. ◀

Recall that, for every $1 \le k \le \sqrt{n}$ and every $\theta \vdash k$, to upper bound

$$\frac{\sum_{\mu\in\mathcal{C}_n} v_\mu^{(m)} \cdot q_{(n)\otimes(m-1,1)}(\mu)\cdot q^2_{(n-k,\theta)\otimes(m-k,\theta)}(\mu)}{m^{\underline{n}}\ (m-1)\ d_{(n-k,\theta)}d_{(m-k,\theta)}} \tag{6.2}$$

$$= \frac{\sum_{\ell=k}^{n}\left((\ell m-n^2)\sum_{\mu\in\mathcal{C}_{n,\ell}} v_\mu^{(m)}\cdot q^2_{(n-k,\theta)\otimes(m-k,\theta)}(\mu)\right)}{m^{\underline{n}}\ n(m-n)\ d_{(n-k,\theta)}d_{(m-k,\theta)}} \tag{6.3}$$

where the second equality follows from Lemma 18. We break the latter sum into two parts, $\ell = k$ and $\ell > k$, then bound these parts individually.

6.3 Case $\ell = k$

Given a partition $\rho = (1^{r_1}, 2^{r_2}, \cdots, n^{r_n})$, define

$$\breve\rho := (1^{r_2}, 2^{r_3}, \cdots, (n-1)^{r_n}) \vdash |\rho| - l(\rho).$$

In terms of Young diagrams, the shape $\breve\rho$ is obtained from ρ by removing the first column. Similarly, for $\mu = (\lambda|\rho) \in \mathcal{C}_n$, define $\breve\mu := (\lambda|\breve\rho)$. In particular, for $\mu \in \mathcal{C}_{n,k}$, we have $\breve\mu \in \mathcal{C}_k$. Also note that, as long as $k \le n/2$, the operation $\rho \mapsto \breve\rho$ defines a one-to-one correspondence between $\mathcal{C}_{n,k}$ and $\mathcal{C}_k$.

Consider $f \in \Omega_\mu$ such that $\mu \in \mathcal{C}_{n,k}$. There exists exactly one assignment α of weight k such that $\alpha \rightsquigarrow f$. Moreover, this assignment satisfies $\mathrm{d}(\alpha) = \breve\mu$. Hence $(\phi_{\breve\mu})_f = 1$ and $(\phi_\nu)_f = 0$ for $\nu \in \mathcal{C}_k \setminus \{\breve\mu\}$. Lemma 24 then implies

$$q_{(n-k,\theta)\otimes(m-k,\theta)}(\mu) = \frac{d_{(m-k,\theta)}}{\binom{n}{k} d_\theta} q_{\xi_\theta}(\breve\mu). \tag{6.4}$$

We also relate the valencies of μ and $\breve\mu$ as follows.

▶ **Proposition 25.** *For $\mu \in \mathcal{C}_{n,k}$, we have $v_\mu^{(m)} = \binom{n}{k}(m-n)^{\underline{n-k}} v_{\breve{\mu}}^{(n)}$.*

Proof. Let $(\lambda|\rho) := \mu$ and $\rho = (1^{r_1}, \cdots, n^{r_n})$, so that $\breve{\mu} = (\lambda|\breve{\rho})$ and $\breve{\rho} = (2^{r_1}, \cdots, (n-1)^{r_n})$. We also have $l(\rho) = n-k$ and $l(\breve{\rho}) = n-k-r_1$. Using Proposition 15, we get

$$\frac{v_{(\lambda|\rho)}^{(m)}}{v_{(\lambda|\breve{\rho})}^{(n)}} = \frac{n!(m-n)^{\underline{n-k}}/r_1!}{k!(n-k)^{\underline{n-k-r_1}}} = \frac{n!(m-n)^{\underline{n-k}}}{k!(n-k)!} = \binom{n}{k}(m-n)^{\underline{n-k}}.$$

Rearranging gives the desired result. ◀

Finally, we need

$$\sum_{\nu \in \mathcal{C}_k} v_\nu^{(n)} q_{\xi_\theta}^2(\nu) = n^{\underline{k}} d_\theta d_{(n-k,\theta)}, \tag{6.5}$$

which holds because, for the primitive idempotent E_{ξ_θ} of $\mathcal{A}_{k,n}$, we have

$$d_{\xi_\theta} = \operatorname{Tr}\big[E_{\xi_\theta}\big] = \operatorname{Tr}\big[E_{\xi_\theta}^2\big] = n^{\underline{k}} \sum_{\nu \in \mathcal{C}_k} v_\nu^{(n)} \left(\frac{q_{\xi_\theta}(\nu)}{n^{\underline{k}}}\right)^2.$$

Putting everything together, we get

$$\begin{aligned}
&\frac{(km-n^2)\sum_{\mu \in \mathcal{C}_{n,k}} v_\mu^{(m)} q^2_{(n-k,\theta)\otimes(m-k,\theta)}(\mu)}{m^{\underline{n}} n(m-n) d_{(n-k,\theta)} d_{(m-k,\theta)}} \\
&= \frac{(km-n^2)\sum_{\nu \in \mathcal{C}_k} \binom{n}{k}(m-n)^{\underline{n-k}} v_\nu^{(n)} \left(\frac{d_{(m-k,\theta)}}{\binom{n}{k} d_\theta} q_{\xi_\theta}(\nu)\right)^2}{m^{\underline{n}} n(m-n) d_{(n-k,\theta)} d_{(m-k,\theta)}} \\
&= \frac{km-n^2}{n(m-n)} \cdot \frac{(m-n)^{\underline{n-k}}}{m^{\underline{n}}} \cdot \frac{d_{(m-k,\theta)} \sum_{\nu \in \mathcal{C}_k} v_\nu^{(n)} q_{\xi_\theta}^2(\nu)}{\binom{n}{k} d_{(n-k,\theta)} (d_\theta)^2} \\
&= \frac{km-n^2}{n(m-n)} \cdot \frac{(m-n)^{\underline{n-k}}}{m^{\underline{n}}} \cdot \frac{k! d_{(m-k,\theta)}}{d_\theta} \\
&\le \frac{km-kn}{n(m-n)} \cdot \frac{(m-n)^{\underline{n-k}}}{m^{\underline{n}}} \cdot \frac{k! m^{\underline{k}}/\mathsf{H}(\theta)}{k!/\mathsf{H}(\theta)} \\
&= \frac{k}{n} \cdot \frac{(m-n)^{\underline{n-k}}}{(m-k)^{\underline{n-k}}} \\
&\le k/n,
\end{aligned}$$

where the first equality is from (6.4) and Proposition 25, the third equality is from (6.5), and for the first inequality we have used

$$d_{(m-k,\theta)} = m!/\mathsf{H}((m-k,\theta)) \le m^{\underline{k}}/\mathsf{H}(\theta).$$

6.4 Case $\ell > k$

Now consider $f \in \Omega_\mu$ such that $\mu \in \mathcal{C}_{n,\ell}$ for $\ell > k$. There are exactly $\binom{\ell}{k}$ assignments α of weight k that agree with f. We therefore have

$$\left|\left(\sum_{\nu \in \mathcal{C}_k} q_{\xi_\theta}(\nu)\phi_\nu\right)_f\right| \le \max_{\nu \in \mathcal{C}_k} |q_{\xi_\theta}(\nu)| \cdot \left(\sum_{\nu \in \mathcal{C}_k} \phi_\nu\right)_f \qquad = q_{\xi_\theta}\big((1^k|\varnothing)\big)\binom{\ell}{k} = d_{\xi_\theta}\binom{\ell}{k},$$

and Lemma 24 implies

$$|q_{(n-k,\theta)\otimes(m-k,\theta)}(\mu)| = |(\phi_{\xi_\theta})_f| \leq d_{(m-k,\theta)} d_{(n-k,\theta)} \ell^{\underline{k}} / n^{\underline{k}}. \tag{6.6}$$

We can also see that

$$\sum_{\mu \in \mathcal{C}_{n,\ell}} v_\mu^{(m)} = \ell! \binom{n}{\ell}^2 (m-n)^{\underline{n-\ell}}, \tag{6.7}$$

as that is the number of elements of $S_{n,m}$ whose image overlaps $[n]$ in ℓ points. Putting everything together, we have

$$\begin{aligned}
&\frac{\sum_{\ell=k+1}^{n} \left((\ell m - n^2) \sum_{\mu \in \mathcal{C}_{n,\ell}} v_\mu^{(m)} \cdot q^2_{(n-k,\theta)\otimes(m-k,\theta)}(\mu) \right)}{m^{\underline{n}} n (m-n) d_{(n-k,\theta)} d_{(m-k,\theta)}} \\
&\leq \frac{\sum_{\ell=k+1}^{n} (\ell m - n^2) \ell! \binom{n}{\ell}^2 (m-n)^{\underline{n-\ell}} \left(d_{(m-k,\theta)} d_{(n-k,\theta)} \ell^{\underline{k}} / n^{\underline{k}} \right)^2}{m^{\underline{n}} n (m-n) d_{(n-k,\theta)} d_{(m-k,\theta)}} \\
&= d_{(m-k,\theta)} d_{(n-k,\theta)} \sum_{\ell=k+1}^{n} \frac{(m-n)^{\underline{n-\ell}}}{m^{\underline{n}}} \frac{\ell m - n^2}{n(m-n)} \binom{n-k}{\ell-k}^2 \ell! \\
&\leq m^k n^k \sum_{\ell=k+1}^{n} \frac{m^{\underline{n-\ell}}}{m^{\underline{n}}} n^{2(\ell-k)} n^\ell \\
&\leq \frac{m^k}{n^k} \sum_{\ell=k+1}^{\infty} \left(\frac{n^3}{m-n} \right)^\ell \\
&= \frac{m^k}{n^k} \cdot \frac{n^{3k+3}}{(m-n)^k} \cdot \frac{1}{m-n-n^3} \\
&\leq 2 \frac{n^{2k+3}}{m},
\end{aligned}$$

where the first inequality follows from (6.6) and (6.7), and the last from $(1-n/m)^k \geq 1 - kn/m$. This completes the proof of the main result.

7 Concluding Remarks and Open Questions

While we have proven a tight $\Omega(\sqrt{n})$ lower bound on the bounded-error quantum query complexity of Index Erasure, the requirement $m \geq n^{3\sqrt{n}}$ on the range of injective functions seems unreasonably strict. We suspect that the same lower bound holds whenever $m \geq c \cdot n$ for any constant $c > \frac{1}{1-\epsilon}$, and we leave proving such a lower bound as an open problem.

When $n = m$, the n-partial permutation association scheme is simply the conjugacy-class association scheme of S_n (see [10]), whose eigenvalues are a normalization of the irreducible characters of S_n. While there is no known closed-formula for computing these normalized characters, they do admit elegant determinantal and combinatorial expressions (i.e., the Jacobi-Trudi and Murnaghan-Nakayama identities). Strahov [25] gave a generalization of the Murnaghan-Nakayama rule for $n = m - 1$, which gives a combinatorial expression for the eigenvalues of $S_{m-1,m}$ association scheme, but for arbitrary $n < m - 1$, there is no known Murnaghan-Nakayama-type rule for expressing the eigenvalues of the $S_{n,m}$ association scheme. Such an expression would give a deeper understanding of the dual eigenvalues of this scheme, and may allow one to extend our lower bound to smaller m.

Finally, our proof suggests a general strategy for deriving lower bounds on the quantum query complexity of sufficiently symmetric state conversion problems in the non-coherent regime. It would be interesting to use our approach to find such lower bounds for other such problems in the non-coherent regime.

References

1. A. Ambainis. Understanding quantum algorithms via query complexity. In *Proceedings of the 2018 International Congress of Mathematicians*, volume 3, pages 3249–3270, 2018.
2. A. Ambainis, L. Magnin, M. Roetteler, and J. Roland. Symmetry-Assisted Adversaries for Quantum State Generation. In *2011 IEEE 26th Annual Conference on Computational Complexity*, pages 167–177, June 2011. doi:10.1109/CCC.2011.24.
3. E. Bannai and T. Ito. *Algebraic Combinatorics I: Association Schemes*. Mathematics lecture note series. Benjamin/Cummings Pub. Co., 1984.
4. A. Belovs and A. Rosmanis. Adversary lower bounds for the collision and the set equality problems. *Quantum Information & Computation*, 18:200–224, 2018.
5. T. Ceccherini-Silberstein, F. Scarabotti, and F. Tolli. *Harmonic Analysis on Finite Groups: Representation Theory, Gelfand Pairs and Markov Chains*. Cambridge Studies in Advanced Mathematics. Cambridge University Press, 2008.
6. P. Diaconis. *Group Representations in Probability and Statistics*. Institute of Mathematical Statistics Lecture Notes—Monograph Series, 11. Institute of Mathematical Statistics, Hayward, CA, 1988.
7. D. Ellis, Y. Filmus, and E. Friedgut. LOW-DEGREE BOOLEAN FUNCTIONS ON S_n , WITH AN APPLICATION TO ISOPERIMETRY. *Forum of Mathematics, Sigma*, 5:e23, 2017. doi:10.1017/fms.2017.24.
8. D. Ellis, E. Friedgut, and H. Pilpel. Intersecting families of permutations. *J. Amer. Math. Soc.*, 24:649–682, 2011.
9. C. Godsil. Notes on Association Schemes, June 2010.
10. C. Godsil and K. Meagher. *Erdos-Ko-Rado Theorems: Algebraic Approaches*. Cambridge Studies in Advanced Mathematics. Cambridge University Press, 2015.
11. C. Godsil and G. Royle. *Algebraic Graph Theory*, volume 207 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 2001. doi:10.1007/978-1-4613-0163-9.
12. A.S. Greenhalgh. Random Walks on Groups with Subgroup Invariance Properties. Technical report, Stanford University, Department of Statistics, April 1989.
13. P. Høyer, T. Lee, and R. Špalek. Negative Weights Make Adversaries Stronger. In *Proceedings of the Thirty-ninth Annual ACM Symposium on Theory of Computing*, STOC '07, pages 526–535, New York, NY, USA, 2007. ACM. doi:10.1145/1250790.1250867.
14. G.D. James and A. Kerber. *The Representation Theory of the Symmetric Group*. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 1984.
15. T. Lee, R. Mittal, B.W. Reichardt, R. Špalek, and M. Szegedy. Quantum Query Complexity of State Conversion. In *Proceedings of the 2011 IEEE 52Nd Annual Symposium on Foundations of Computer Science*, FOCS '11, pages 344–353, Washington, DC, USA, 2011. IEEE Computer Society. doi:10.1109/FOCS.2011.75.
16. G. Midrijānis. A Polynomial Quantum Query Lower Bound for the Set Equality Problem. In *Automata, Languages and Programming*, pages 996–1005, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.
17. A. Montanaro. Quantum algorithms: An overview. *npj Quantum Information*, 2, November 2015. doi:10.1038/npjqi.2015.23.
18. M.A. Nielsen and I.L. Chuang. *Quantum Computation and Quantum Information*. Cambridge Series on Information and the Natural Sciences. Cambridge University Press, 2000.
19. R. O'Donnell. *Analysis of Boolean Functions*. Cambridge University Press, 2014.
20. A. Rosmanis. Quantum Adversary Lower Bound for Element Distinctness with Small Range. *Chicago Journal of Theoretical Computer Science*, 2014(4), July 2014.
21. A. Rosmanis and A. Belovs. On Adversary Lower Bounds for the Collision and the Set Equality Problems, 2013. Available at arXiv:1310.5185v1 [quant-ph].
22. B. Sagan. *The Symmetric Group: Representations, Combinatorial Algorithms, and Symmetric Functions*. Graduate Texts in Mathematics. Springer New York, 2001.

23 Y. Shi. Quantum Lower Bounds for the Collision and the Element Distinctness Problems. In *Proceedings of the 43rd Symposium on Foundations of Computer Science*, FOCS '02, pages 513–519, Washington, DC, USA, 2002. IEEE Computer Society.

24 R.P. Stanley. *Enumerative Combinatorics: Volume 2*. Cambridge Studies in Advanced Mathematics. Cambridge University Press, 2001.

25 E. Strahov. Generalized characters of the symmetric group. *Advances in Mathematics*, 212(1):109–142, 2007. doi:10.1016/j.aim.2006.09.017.

26 M. Zhandry. A Note on the Quantum Collision and Set Equality Problems. *Quantum Info. Comput.*, 15(7-8):557–567, May 2015.

A Connection to the graph isomorphism

Suppose we are given two rigid graphs G_0 and G_1 on k vertices (e.g., as $k \times k$ adjacency matrices), and we are asked to decide whether there exists a permutation of vertices $\pi \in S_k$ such that $\pi(G_0) = G_1$. Let $S_k(G) := \{\pi(G) \colon \pi \in S_k\}$ and

$$|S_k(G)\rangle := \frac{1}{\sqrt{k!}} \sum_{\pi \in S_k} |\pi(G)\rangle.$$

For a rigid graph G, the function $\pi \mapsto \pi(G)$ is injective and $|S_k(G)\rangle$ is the uniform superposition over the image of this function. Note that $S_k(G_0) = S_k(G_1)$ and $\langle S_k(G_0)|S_k(G_1)\rangle = 1$ if $G_0 \cong G_1$ and $S_k(G_0) \cap S_k(G_1) = \emptyset$ and $\langle S_k(G_0)|S_k(G_1)\rangle = 0$ if $G_0 \ncong G_1$.

Here we present two algorithms for Graph Isomorphism based on ability to generate the state $|S_k(G)\rangle|t_G\rangle$, where $|t_G\rangle$ is the final state of the ancillary memory. Both algorithms always return 0 when the graphs are isomorphic and return 1 with probability 1/2 when the graphs are non-isomorphic.

A.1 Coherent test

Suppose we prepare a quantum state

$$\frac{1}{\sqrt{2}} \sum_{b \in \{0,1\}} |b\rangle|S_k(G_b)\rangle|t_{G_b}\rangle,$$

where the first register specifies whether we create the superposition over permutations of G_0 or G_1 in the second register. Then we apply the Hadamard gate on the first register and measure. This procedure returns 0 with probability

$$\frac{1}{2} + \frac{1}{2}\Re\big[\langle S_k(G_0)|S_k(G_1)\rangle \langle t_{G_0}|t_{G_1}\rangle\big].$$

If $G_0 \ncong G_1$, then $|S_k(G_0)\rangle$ and $|S_k(G_1)\rangle$ are orthogonal states, and the measurement returns 1 with probability 1/2. On the other hand, if $G_0 \cong G_1$, then the probability of outputting 0 completely depends on $\Re[\langle t_{G_0}|t_{G_1}\rangle]$, on which we do not place any restrictions in the non-coherent case. However, in the coherent case, $\Re[\langle t_{G_0}|t_{G_1}\rangle] = 1$ and 0 would be always output whenever $G_0 \cong G_1$.

A.2 Non-coherent test

Now suppose we first prepare a quantum state

$$\frac{1}{\sqrt{2}}\Big(|0\rangle + |1\rangle\Big)|S_k(G_0)\rangle|S_k(G_1)\rangle|t_{G_0}\rangle|t_{G_1}\rangle,$$

and then use the content of the first register to decide whether to swap the next two registers, obtaining

$$\frac{1}{\sqrt{2}}\Big(|0\rangle|S_k(G_0)\rangle|S_k(G_1)\rangle + |1\rangle|S_k(G_1)\rangle|S_k(G_0)\rangle\Big)|t_{G_0}\rangle|t_{G_1}\rangle.$$

Then we apply the Hadamard gate on the first register and measure. This procedure returns 0 with probability

$$\frac{1}{2} + \frac{1}{2}\big\|\langle S_k(G_0)|S_k(G_1)\rangle\big\|^2.$$

Note that the ancillary memory states $|t_{G_0}\rangle$ play no role in this test.

We note that our $\Omega(\sqrt{n})$ lower bound for non-coherent Index Erasure does not apply here because of the condition $m = \Omega(n^{\sqrt{n}})$. In this context, n is the number of vertex permutations, $k!$, and m is the number of rigid graphs, which is of order $2^{\binom{k}{2}}$ [11, Corollary 2.3.3]. Nonetheless, the $\Omega(n^{1/3})$ lower bound (via Set Equality) still applies, which tells us that both Graph Isomorphism tests based on Index Erasure are inefficient. However, this dose not rule out an efficient preparation of $|S_k(G)\rangle$ by exploiting some inner workings of the oracle O_G (that is, by not treating it as a black-box).

B Dual Eigenvalues of Maximal Irreps

In Lemma 24 we essentially expressed dual eigenvalues for irreps of form $(n-k,\theta)\otimes(m-k,\theta)$ via dual eigenvalues for irreps of form $\lambda\otimes(m-n,\lambda)$, where $\theta\vdash k$ and $\lambda\vdash n$. Here we address obtaining the dual eigenvalues for irreps of form $\lambda\otimes(m-n,\lambda)$, in particular, obtaining the column vector $(m^{\underline{n}}E_{\lambda\otimes(m-n,\lambda)})_{f_{\mathrm{id}}}$, whose entries are the dual eigenvalues.

Of course, since we have $E_{\lambda\otimes(m-n,\lambda)} = E_{(m-n,\lambda)}$, we have

$$(m^{\underline{n}}E_{\lambda\otimes(m-n,\lambda)})_{f_{\mathrm{id}}} = m^{\underline{n}}E_{(m-n,\lambda)}1_{f_{\mathrm{id}}},$$

where

$$E_{(m-n,\lambda)} = \frac{d_{(m-n,\lambda)}}{m!}\sum_{\pi\in S_m}\chi_{(m-n,\lambda)}(\pi)V_\pi$$

and $V_\pi\colon 1_f \mapsto 1_{\pi * f}$. Below, however, we present an expression for $(m^{\underline{n}}E_{\lambda\otimes(m-n,\lambda)})_{f_{\mathrm{id}}}$ that might be more usefull when $n \ll m$, especially when n is constant.

Suppose $m \geq 2n$ and consider a $2n$-tuple

$$\boldsymbol{a} := (a_1^{(0)}, a_2^{(0)}, \cdots, a_n^{(0)}, a_1^{(1)}, a_2^{(1)}, \cdots, a_n^{(1)}) \in [m]^{2n}$$

of distinct elements. Consider the idempotent

$$E_{\boldsymbol{a}} := \frac{I - V_{(a_1^{(0)},a_1^{(1)})}}{2}\cdot\frac{I - V_{(a_2^{(0)},a_2^{(1)})}}{2}\cdot\cdots\cdot\frac{I - V_{(a_n^{(0)},a_n^{(1)})}}{2}.$$

The image of $E_{\boldsymbol{a}}$ is orthogonal to all λ'-isotypic subspaces for all $\lambda'\vdash m$ with $\lambda'_1 > m-n$ [4, Lemma 8]. For the $\mathcal{A}_{n,m}$ scheme, this means that the image of $E_{\boldsymbol{a}}$ is orthogonal to all irreps $\lambda\otimes\lambda'$ such that $\lambda' \neq (m-n,\lambda)$. On the other hand, for the vector

$$\phi_{\boldsymbol{a}} := \sum_{b\in\{0,1\}^n}(-1)^{|b|}1_{\boldsymbol{a}^b},$$

where $\boldsymbol{a}^b\colon [n]\to[m]\colon x\mapsto a_x^{(b_x)}$, we have $E_{\boldsymbol{a}}\phi_{\boldsymbol{a}} = \phi_{\boldsymbol{a}}$.

Now consider $a_x^{(0)} = x$ for all $x \in [n]$ and $(m-n)^{\underline{n}}$ choices of distinct $a_1^{(1)}, a_2^{(1)}, \cdots, a_n^{(1)}$ from $\{n+1, n+2, \cdots, m\}$, with their corresponding vectors $\phi_{\boldsymbol{a}}$. By adding all these vectors and then dividing the result by $(m-n)^{\underline{n}}$, we obtain

$$\phi_{\max} := \sum_{\ell=0}^{n} \frac{(-1)^\ell}{(m-n)^{\underline{\ell}}} 1_{(1^{n-\ell}|1^\ell)} \in (\mathfrak{A}_{n,m})_{f_{\mathrm{id}}},$$

where, for $\mu \in \mathcal{C}_n$, 1_μ is the characteristic column of A_μ, namely, $1_\mu = \sum_{f \in \Omega_\mu} 1_f$.

Now consider $\lambda \vdash n$ and the projector

$$E_\lambda = \frac{d_\lambda}{n!} \sum_{\pi \in S_n} \chi_\lambda(\pi) V_\pi$$

on the λ-isotypic subspace. By the above discussion, we have

$$E_\lambda \phi_{\max} = E_{\lambda \otimes (m-n,\lambda)} \phi_{\max},$$

which is proportional to the characteristic column of $E_{\lambda\otimes(m-n,\lambda)} 1_{f_{\mathrm{id}}}$, and the coefficient of proportionality is

$$\frac{1_{f_{\mathrm{id}}}^\top E_\lambda \phi_{\max}}{1_{f_{\mathrm{id}}}^\top E_{\lambda\otimes(m-n,\lambda)} 1_{f_{\mathrm{id}}}} = \frac{(d_\lambda/n!) d_\lambda}{d_\lambda d_{\lambda'}/m^{\underline{n}}} = \frac{d_\lambda m^{\underline{n}}}{d_{\lambda'} n!},$$

where we have used that $1_{f_{\mathrm{id}}}^\top V_\pi 1_{(1^{n-\ell}|1^\ell)} = 0$ whenever π is not the identity permutation ε or $\ell \neq 0$ (or both), $1_{(1^n|\varnothing)} = 1_{f_{\mathrm{id}}}$, and $\chi_\lambda(\varepsilon) = d_\lambda$. Hence,

$$(m^{\underline{n}} E_{\lambda\otimes(m-n,\lambda)})_{f_{\mathrm{id}}} = \frac{d_{\lambda'} n!}{d_\lambda} E_\lambda \phi_{\max} = d_{\lambda'} \sum_{\pi \in S_n} \chi_\lambda(\pi) V_\pi \phi_{\max}.$$

C Partial Permutations and RSK Correspondence

A well-known fact is that S_m admits the following representation-theoretic count

$$|S_m| = \sum_{\lambda \vdash m} (d_\lambda)^2 \tag{C.1}$$

where d_λ is the number of standard Young tableau of shape $\lambda \vdash m$ (see [22]). An elegant combinatorial proof of this fact follows from *Robinson-Schensted Correspondence*, a well-known combinatorial procedure that associates to each permutation $\sigma \in S_m$ a unique pair of standard Young tableaux of the same shape, and vice versa (see [22]).

Knuth generalized this correspondence to a wider class of combinatorial objects called *generalized permutations*, which are $2 \times m$ arrays of integers

$$\begin{pmatrix} i_1 & i_2 & \cdots & i_m \\ j_1 & j_2 & \cdots & j_m \end{pmatrix} \text{ such that } i_1 \leq \cdots \leq i_m \text{ and if } i_r = i_{r+1}, \text{then } j_r \leq j_{r+1}.$$

Robinson-Schensted-Knuth Correspondence (RSK) associates a pair of semistandard Young tableau of the same shape to each generalized permutation, and vice versa (see [22]). We may encode each n-partial permutation $(j_1, j_2, \cdots, j_n) \in S_{n,m}$ as a generalized permutation as follows:

$$(j_1, j_2, \cdots, j_n) \longleftrightarrow \begin{pmatrix} 1 & 2 & \cdots & n & n+1 & \cdots & n+1 \\ j_1 & j_2 & \ldots & j_n & j_{n+1} & \cdots & j_m \end{pmatrix},$$

where $j_{n+1}, \cdots, j_m \in [m] \setminus \{j_1, \cdots, j_n\}$ are ordered from least to greatest. Applying RSK to n-partial permutations associates to each $(j_1, j_2, \cdots, j_n) \in S_{n,m}$ a standard Young tableau P and a semistandard Young tableau Q of the same shape $\lambda \vdash m$. The subtableau of cells labeled $n+1$ in Q form a horizontal strip on $m-n$ cells. Ignoring this horizontal strip results in a standard Young tableau of shape $\mu \vdash n$ such that λ/μ is a horizontal strip, and so we arrive at the following theorem.

▶ **Theorem 26.** *RSK gives an explicit bijection between $S_{n,m}$ and pairs (P, Q) where P is a standard Young tableau of shape $\lambda \vdash m$ and Q is a standard Young tableau of shape $\mu \vdash n$ such that λ/μ is a horizontal strip.*

As a corollary, we get a combinatorial proof of a natural generalization of Equation (C.1).

▶ **Corollary 27.** *The number of n-partial permutations of $[m]$ can be counted as follows:*

$$|S_{n,m}| = \sum_{\mu,\lambda} d_\mu d_\lambda$$

where the sum runs over pairs $\mu \vdash n, \lambda \vdash m$ such that λ/μ is a horizontal strip.

D Proofs

Theorem 7 (restated). *Let $\theta \vdash k$ and $\theta^+ \vdash (k+1)$ be any shape obtained by adding an inner corner to θ. For all $m \geq 2(k+1)$, we have*

$$\frac{d_{(m-k-1,\theta^+)}}{d_{(m-k,\theta)}} \geq \frac{m}{k} \cdot \left(1 - \frac{2k+1}{m}\right).$$

Proof. Recall that, by the hook rule, $\mathsf{H}(\theta) d_\theta = |\theta|!$. First, [20, Claim 6.3] states that

$$\frac{d_{(m-|\theta^+|,\theta)}}{d_{(m-|\theta|,\theta)}} \geq 1 - \frac{2k}{m}.$$

We reprove this claim here for completeness. Note that when we add a cell to the end of the top row of $(m-|\theta^+|, \theta)$ to obtain $(m-|\theta|, \theta)$, this increases the hook-lengths of the cells in the top row by 1, and the rest of the hook-lengths are unchanged. If we just consider the "overhang" and ignore everything else in the first row, then the product of the hook-lengths with respect to $(m-|\theta|, \theta)$ is $(m-2k)!$ whereas it is $(m-2k-1)!$ with respect to $(m-|\theta^+|, \theta)$. This gives us

$$\frac{d_{(m-|\theta^+|,\theta)}}{d_{(m-|\theta|,\theta)}} \geq \frac{m-2k}{m} = 1 - \frac{2k}{m},$$

which proves the claim.

Since $(m-|\theta^+|, \theta)$ is a partition of $(m-1)$, it corresponds to an (S_{m-1})-irrep. When we added one cell to $(m-|\theta^+|, \theta)$ to obtain $(m-|\theta^+|, \theta^+)$, only one hook-length in the first row increased, and before the increment it was at least $m-2k-1$. Thus, in the following derivation, all the other hook-lengths of the first rows of $(m-|\theta^+|, \theta)$ and $(m-|\theta^+|, \theta^+)$ have cancelled out.

$$\begin{aligned}\frac{d_{(m-|\theta^+|,\theta^+)}}{d_{(m-|\theta^+|,\theta)}} &\geq \frac{m!}{(m-1)!} \cdot \frac{m-2k-1}{m-2k} \cdot \frac{\mathsf{H}(\theta)}{\mathsf{H}(\theta^+)} \\ &= m\left(1 - \frac{1}{m-2k}\right) \frac{k! d_{\theta^+}}{(k+1)! d_\theta} \geq \frac{m}{k+1}\left(1 - \frac{1}{m-2k}\right),\end{aligned}$$

where in the middle equality we have used hook-length formula once more, and the last inequality follows from the branching rule (namely, that $d_{\theta^+} \geq d_\theta$). Combining these two inequalities gives the result. ◀

Theorem 12 (restated). *Let $1_\alpha \in \mathbb{C}[S_{n,m}]$ be the characteristic function of the family S_α. Then $1_\alpha \in \mathcal{U}_k$.*

Proof. Let $\mathrm{Stab}_{S_m}(S_\alpha)$ be the stabilizer of S_α, which consists of all the permutations of $[m] \setminus \mathrm{im}(\alpha)$ and, thus, $\mathrm{Stab}_{S_m}(S_\alpha) \cong S_{m-k}$. Let $\lambda \vdash m$ be any irrep of S_m such that $m - \lambda_1 > k$ and let $\lambda(\pi)$ be a matrix that represents $\pi \in S_m$. Finally, let $1_{\mathrm{Stab}(S_\alpha)} \in \mathbb{C}[S_m]$ be the characteristic function of $\mathrm{Stab}_{S_m}(S_\alpha)$, and recall from Section 3.1 that its Fourier transform is

$$\lambda(1_{\mathrm{Stab}(S_\alpha)}) = \sum_{\pi \in S_m} 1_{\mathrm{Stab}(S_\alpha)}(\pi)\, \lambda(\pi) = \sum_{\pi \in S_{m-k}} \left(\lambda \downarrow^{S_m}_{S_{m-k}}\right)(\pi).$$

Since λ has more than k cells below its first row, by the branching rule, no irrep μ of $\lambda \downarrow^{S_m}_{S_{m-k}}$ is isomorphic to the trivial representation $(m-k)$. This, and the fact that $\langle \chi^{(m-k)}, \chi^\mu \rangle = 0$ for any $\mu \vdash m-k$ with $\mu \neq (m-k)$, implies that, for each irrep μ of the multiset of irreps $\lambda \downarrow^{S_m}_{S_{m-k}}$, we have $\sum_{\pi \in S_{m-k}} \mu(\pi) = 0$. This gives us

$$\lambda(1_{\mathrm{Stab}(S_\alpha)}) = \bigoplus_{\mu \in \lambda \downarrow^{S_m}_{S_{m-k}}} \sum_{\pi \in S_{m-k}} \mu(\pi) = 0,$$

which completes the proof. ◀

Theorem 4 (restated). *Let G be an automorphism group for a non-coherent state generation problem. The value of Adv_δ remains the same if one restricts the minimization in the expression defining Adv_δ and the maximization in the expressions defining the γ_2 and filtered γ_2 norms to R, T, Γ, Γ' that are all G-invariant and imposes that $(J - R) \circ \Gamma$ has an G-invariant principal eigenvector.*

Proof. In this proof, let M denote a generic symmetric matrix whose rows and columns are labeled by black-box functions $f \in \mathcal{F}$ in the same order. Let $g(M)$ be obtained by permuting the rows and the columns of M according to the action of $g \in G$ of $\mathcal{F}$ (see (2.2)). Namely, entrywise we define $g(M)$ as

$$(g(M))_{f,f'} := M_{g^{-1}(f), g^{-1}(f')}.$$

Similarly, for a vector $\omega \in \mathbb{C}[\mathcal{F}]$, define $g(\omega)$ entrywise as $(g(\omega))_{f'} := \omega_{g^{-1}(f)}$. For the sake of conciseness, we also occasionally write M^g and ω^g instead of $g(M)$ and $g(\omega)$, respectively. Note that M is G-invariant if $M^g = M$ for all $g \in G$, and $T^\odot, I, J$ are G-invariant. Also note that $(M \circ M')^g = M^g \circ M'^g$.

Let $\Delta = \{\Delta_1, \cdots, \Delta_n\}$ be the family of difference matrices. This family is closed under the action of G is the following sense.

▷ Claim 28. We have $(\pi, \boldsymbol{\sigma})(\Delta_x) = \Delta_{\pi(x)}$ for all $(\pi, \boldsymbol{\sigma}) \in G$.

Proof. Fix $(\pi, \boldsymbol{\sigma}) \in G$ and let $g := (\pi, \boldsymbol{\sigma})^{-1}$. Note that $g = (\pi^{-1}, \boldsymbol{\sigma}')$ for some $\boldsymbol{\sigma}' \in S_m^n$. From (2.2), we have $(g(f))(x) = (g(f'))(x)$ if and only if $f(\pi(x)) = f'(\pi(x))$. As a result, we have

$$((\pi, \boldsymbol{\sigma})(\Delta_x))_{f,f'} = (\Delta_x)_{g(f), g(f')} = 1$$

if and only if $f(\pi(x)) = f'(\pi(x))$. ◀

Note that M^g equals M with its rows and columns permuted. Since permuting rows and columns do not affect the γ_2 norm, we have $\gamma_2(M^g) = \gamma_2(M)$ for all $g \in G$. And, if the diagonal of M is all-zeros, then Claim 28 also implies that $\gamma_2(M^g|\Delta) = \gamma_2(M|\Delta)$ for all $g \in G$.

▶ **Lemma 29.** *Restricting $R, T \in \mathcal{T}$ to be G-invariant does not change the optimal value of the minimization problem defining Adv_δ.*

Proof. Let R, T be an optimal solution of the minimization in (2.1). We define their respective G-symmetrizations as

$$\overline{R} := \frac{1}{|G|}\sum_{g\in G} g(R) \qquad \text{and} \qquad \overline{T} := \frac{1}{|G|}\sum_{g\in G} g(T),$$

which are both clearly in $\mathcal{T}$. Since $g(T^\odot) = T^\odot$ for all $g \in G$, the triangle inequality yields

$$\begin{aligned}\gamma_2(\overline{R} - T^\odot \circ \overline{T}) = \gamma_2\left(\frac{1}{|G|}\sum_{g\in G} g(R - T^\odot \circ T)\right) &\le \frac{1}{|G|}\sum_{g\in G}\gamma_2\big(g(R - T^\odot \circ T)\big)\\ &= \frac{1}{|G|}\sum_{g\in G}\gamma_2(R - T^\odot \circ T) = \gamma_2(R - T^\odot \circ T) \le \delta.\end{aligned}$$

Hence we have show that the pair $\overline{R}, \overline{T}$ is a feasible solution to the minimization in (2.1), and it remains to show that it is also optimal. And, again by the triangle inequality,

$$\begin{aligned}\gamma_2(J - \overline{R}|\Delta) = \gamma_2\left(\frac{1}{|G|}\sum_{g\in G} g(J - R)\middle|\Delta\right) &\le \frac{1}{|G|}\sum_{g\in G}\gamma_2\big(g(J - R)|\Delta\big)\\ &= \frac{1}{|G|}\sum_{g\in G}\gamma_2(J - R|\Delta) = \gamma_2(J - R|\Delta) = \mathsf{Adv}_\delta.\end{aligned}$$

◀

Now, fix G-invariant $R, T \in \mathcal{T}$ and let $M := J - R$, which is also G-invariant. Let us now show that the maximization in

$$\gamma_2(M|\Delta) = \max_{\Gamma}\left\{\|M \circ \Gamma\| \colon \forall x\ \|\Delta_x \circ \Gamma\| \le 1\right\}$$

can be restricted to G-invariant Γ.

The proof is very similar to that of the automorphism principle in [15]. Fix an optimal solution Γ, and without loss of generality assume that the largest eigenvalue of $M \circ \Gamma$ is positive and let it correspond to an eigenvector $\omega \in \mathbb{C}[\mathcal{F}]$ of norm 1. Namely,

$$\|M \circ \Gamma\| = \omega^\top (M \circ \Gamma)\omega.$$

Define the G-symmetrization $\overline{\omega}$ of ω entrywise as

$$\overline{\omega}_f := \sqrt{\frac{1}{|G|}\sum_{g\in G} |(\omega^g)_f|^2},$$

and note that $\overline{\omega}$ also has norm 1. Without loss of generality, all the entries of $\overline{\omega}$ are strictly positive (the rows and columns corresponding to f such that $\overline{\omega}_f = 0$ can be removed from the consideration), and thus we can entrywise define a vector μ as $\mu_f := 1/\overline{\omega}_f$. Let us define

$$\overline{\Gamma} := \mu\mu^\top \circ \frac{1}{|G|} \sum_{g \in G} \Gamma^g \circ \omega^g \omega^{g\top},$$

which is clearly G-invariant.

Let us start by showing that $\|\overline{\Gamma} \circ \Delta_x\| \leq 1$ for all x. Note that $\|\Gamma^g \circ \Delta_x\| \leq 1$ for all x and all $g \in G$ due to Claim 28, $\|\Gamma \circ \Delta_x\| \leq 1$ if and only if $I \pm \Gamma \circ \Delta_x$ is positive-semidefinite, and

$$I \circ \mu\mu^\top \circ \frac{1}{|G|} \sum_{g \in G} \omega^g \omega^{g\top} = I.$$

We thus have that

$$I \pm \overline{\Gamma} \circ \Delta_x = \mu\mu^\top \circ \frac{1}{|G|} \Big(\sum_{g \in G} \omega^g \omega^{g\top} \circ (I \pm \Gamma^g \circ \Delta_x) \Big)$$

is positive-semidefinite as the sum and the entrywise product of positive-semidefinite matrices are positive-semidefinite. Thus, indeed, $\|\overline{\Gamma} \circ \Delta_x\| \leq 1$ for all x.

Now let us use the fact that ω is a principal eigenvector of $M \circ \Gamma$, and, therefore, ω^g is a principal eigenvector of $M \circ \Gamma^g$ for all $g \in G$ (recall that M is G-invariant). We have

$$\begin{aligned} \|M \circ \overline{\Gamma}\| \geq \overline{\omega}\, \overline{\Gamma}\, \overline{\omega}^\top &= \sum_{f,f' \in \mathcal{F}} \Big(\frac{1}{|G|} \sum_{g \in G} (M \circ \Gamma^g \circ \omega^g \omega^{g\top}) \Big)_{f,f'} \\ &= \frac{1}{|G|} \sum_{g \in G} \omega^{g\top} (M \circ \Gamma^g) \omega^g = \|M \circ \Gamma\|. \end{aligned}$$

Thus $\overline{\Gamma}$ is also an optimal solution of the maximization above. Also note that $\overline{\omega}$ is the principal eigenvector of $M \circ \overline{\Gamma}$.

A similar argument shows that, for G-invariant $M' := R - T^{\odot} \circ T$, one can restrict the maximization in

$$\gamma_2(M') = \max_{\Gamma'} \big\{ \|M' \circ \Gamma'\| \colon \|\Gamma'\| \leq 1 \big\}$$

to G-invariant Γ'. ◀

Optimal Single-Choice Prophet Inequalities from Samples

Aviad Rubinstein
Stanford University, Stanford, CA, USA
aviad@cs.stanford.edu

Jack Z. Wang
Cornell University, Ithaca, NY, USA
jackzwang@cs.cornell.edu

S. Matthew Weinberg
Princeton University, Princeton, NJ, USA
smweinberg@princeton.edu

Abstract

We study the single-choice Prophet Inequality problem when the gambler is given access to samples. We show that the optimal competitive ratio of 1/2 can be achieved with a single sample from each distribution. When the distributions are identical, we show that for any constant $\varepsilon > 0$, $O(n)$ samples from the distribution suffice to achieve the optimal competitive ratio (≈ 0.745) within $(1+\varepsilon)$, resolving an open problem of [9].

2012 ACM Subject Classification Theory of computation → Online algorithms

Keywords and phrases Online algorithms, Probability, Optimization, Prophet inequalities, Samples, Auctions

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.60

Funding *S. Matthew Weinberg*: Supported by NSF CCF-1717899.

1 Introduction

Consider the classic single-choice Prophet Inequality problem. Offline, there are n distributions $\mathcal{D}_1, \ldots, \mathcal{D}_n$ presented to a gambler. For $i = 1$ to n, a random variable X_i is drawn independently from $\mathcal{D}_i$ and revealed online. The gambler must then decide immediately and irrevocably whether to accept X_i (and achieve reward X_i, ending the game), or reject X_i (continuing the game, but never revisiting X_i again). The goal of the gambler is to devise a stopping rule which maximizes their expected reward. The performance of potential stopping rules is typically measured by their *competitive ratio* in comparison to a prophet (who knows all X_i in advance and achieves expected reward $\mathbb{E}[\max_i\{X_i\}]$). Typically, prophet inequalities are designed assuming that the distributions presented offline are fully known. This paper focuses on the setting where the gambler is instead presented with offline samples from the $\mathcal{D}_i$, rather than complete knowledge.

In the classic setting, seminal work of Krengel and Sucheston provides a strategy guaranteeing a competitive ratio of 1/2, which is the best possible [22].[1] Samuel-Cahn later proved that simply setting a threshold equal to the *median* of $\max_i\{X_i\}$ (i.e. a value T such that $\Pr[\max_i\{X_i\} > v] = 1/2$) also achieves the optimal competitive ratio of 1/2 [25], and

[1] To see that no better than 1/2 is possible, consider an instance where X_1 is deterministically 1, and X_2 is $1/\varepsilon$ with probability ε, and 0 otherwise. The prophet achieves $2-\varepsilon$ (taking X_2 when it is large, and X_1 otherwise), while the gambler achieves only 1 (they must decide whether to take X_1 without knowing if X_2 is large).

it was later shown that a threshold of $\mathbb{E}[\max_i\{X_i\}]/2$ suffices as well [21]. These last two thresholds are remarkably simple, but certainly require a non-trivial number of samples to estimate well.

Our first result establishes that *a single sample* from each $\mathcal{D}_i$ suffices to achieve the optimal competitive ratio of $1/2$. The algorithm is also exceptionally simple: if $\tilde{X}_1, \ldots, \tilde{X}_n$ denote independent samples from $\mathcal{D}_1, \ldots, \mathcal{D}_n$, simply set $\max_i\{\tilde{X}_i\}$ as a threshold.

▶ **Definition 1** (Single Sample Algorithm)**.** *Given as input $\tilde{X}_1, \ldots, \tilde{X}_n$, set a threshold $T = \max_i\{\tilde{X}_i\}$ and accept the first random variable exceeding T.*

▶ **Theorem 2.** *The Single Sample Algorithm guarantees a competitive ratio of $1/2$.*

A subsequent line of works considers the special case where each $\mathcal{D}_i$ is identical (which we'll refer to as $\mathcal{D}$). Here, work of Hill and Kertz provided the first improved competitive ratio (of $1 - 1/e$) [19], and this was recently improved to the optimal competitive ratio of $\alpha \approx 0.745$ [10]. Our second result establishes that a *linear number of samples* from $\mathcal{D}$ suffices to achieve the optimal competitive ratio, up to ε. Since our algorithm simply replaces the quantile-based thresholds of [10] with samples, we call it Samples-CFHOV (the five authors of [10]). The algorithm and analysis are fairly simple and we provide a formal description in Section 4.

▶ **Theorem 3.** *With $O(n/\varepsilon^6)$ samples, Samples-CFHOV achieves a competitive ratio of $\alpha - O(\varepsilon)$.*

1.1 Related Work

Over the past decade, prophet inequalities have been studied from numerous angles within the TCS community [7, 3, 21, 17, 15, 23, 16, 24, 12, 14, 6, 2, 11, 4, 18, 13, 8]. All of these works assume explicit knowledge of the given distributions. The limited prior work most related to ours considers sample access to the underlying distributions. On this front, Azar et al. consider prophet inequalities subject to combinatorial constraints, and establish that limited samples suffice to obtain constant competitive ratios in many settings [5]. In comparison to this work, our paper considers only *optimal* competitive ratios, and the simple single-choice setting.

In the i.i.d. model (each value is drawn from the same $\mathcal{D}$), a $(1 - 1/e)$-approximation was first shown in [19], and recent work achieved the same guarantee with $n - 1$ samples from $\mathcal{D}$ [9]. This ratio was later improved to ≈ 0.738 [1], and then to $\alpha \approx 0.745$ [10], where α is the optimal achievable competitive ratio [19, 20]. The most related work in this sequence to ours is [9], who establish that a competitive ratio of $\alpha - \varepsilon$ is achievable with $O(n^2)$ samples (for any constant ε), and that $\Omega(n)$ samples are necessary. They also establish a formal barrier to achieving $\alpha - \varepsilon$ with $o(n^2)$ samples. In comparison, we circumvent their barrier to achieve a competitive ratio of $\alpha - \varepsilon$ with $O(n)$ samples, resolving one of their open problems.

Roadmap

In Section 2, we provide brief preliminaries. Section 3 contains our 2-approximation with a single sample. Section 4 contains our $(\alpha - \varepsilon)$-approximation with linearly many samples.

2 Preliminaries

There are n distributions, $\mathcal{D}_1, \ldots, \mathcal{D}_n$. Online, a gambler sees a random variable X_i drawn from $\mathcal{D}_i$ one at a time, and must immediately and irrevocably decide whether to accept (and get reward X_i) or reject (and see X_{i+1}, throwing away X_i forever). Strategies for a gambler are often termed *stopping rules*, and the competitive ratio of a stopping rule is the worst-case ratio (over all possible n, and $\mathcal{D}_1, \ldots, \mathcal{D}_n$) between its expected reward and $\mathbb{E}[\max_i\{X_i\}]$.

Our algorithms will not have knowledge of any $\mathcal{D}_i$, but instead will have access to samples. Our algorithms will treat these samples as the only offline input, and decide whether to accept or reject an element based only on the value of that element and the samples.[2] Here, we will count the number of samples from each distribution as our sample complexity.

We will also consider the *i.i.d. setting*, where each $\mathcal{D}_i = \mathcal{D}$. Here, we will count the total number of samples from $\mathcal{D}$ as our sample complexity. In this setting, we will let $\alpha \approx 0.745$ denote the optimal competitive ratio for an algorithm with knowledge of $\mathcal{D}$.

Continuous versus Discrete Random Variables

All of our algorithm definitions are straight-forward for continuous distributions. For distributions with point masses, the following "reduction" to continuous is needed. Instead of thinking of $\mathcal{D}$ as a single-variate distribution, we will (overload notation and) think of $\mathcal{D}$ as a bivariate distribution with the first coordinate drawn from $\mathcal{D}$, and the second "tie-breaker" coordinate drawn independently and uniformly from $[0, 1]$. Then $(X_1, t_1) > (X_2, t_2)$ if either $X_1 > X_2$, or $X_1 = X_2$ and $t_1 > t_2$. Observe that because the tie-breaker coordinate is continuous, the probability of having $(X_1, t_1) = (X_2, t_2)$ for any two values during a run of any algorithm is zero. Therefore, if we define $F_{\mathcal{D}}(X, t) := \Pr_{(Y,u)\leftarrow(\mathcal{D},U([0,1]))}[(Y, u) < (X, t)]$, we have that $F_{\mathcal{D}}(X, t) < F_{\mathcal{D}}(Y, u) \Leftrightarrow (X, t) < (Y, u)$. We will not explicitly reference this tie-breaker random variable in the definition of our algorithms, but simply refer to $X \leftarrow \mathcal{D}$ as the pair (X, t).

Adversaries

Prophet Inequalities are typically studied against an *offline adversary*. That is, the adversary simply picks the distributions $\mathcal{D}_1, \ldots, \mathcal{D}_n$ (and their indices), which is all presented to the gambler offline. Some prophet inequalities hold against the stronger *almighty adversary*, which selects the set of distributions $\{\mathcal{D}_1, \ldots, \mathcal{D}_n\}$ offline, then decides in which order to reveal the random variables $X_1, \ldots, X_n$ based on their realization. Note that previous competitive ratios of $1/2$ in the non-i.i.d. setting hold against an almighty adversary, and Theorem 2 does as well. Previous competitive ratios of α in the i.i.d. setting hold against the offline adversary (and are impossible to achieve against the almighty adversary), so Theorem 3 holds against the offline adversary as well.

3 The Non-I.I.D. Case: Optimal Ratio with a Single Sample From $\mathcal{D}_i$

The Single Sample Algorithm proceeds as follows. It takes as input $\tilde{X}_i$ drawn independently from each $\mathcal{D}_i$, sets a threshold $T = \max_i\{\tilde{X}_i\}$, and accepts the first element exceeding T. Our goal in this section is to prove Theorem 2 that this algorithm obtains $\frac{1}{2}$ the reward, in expectation, of the omniscient prophet that always selects the highest value.

[2] In principle, sample-based algorithms might also consider previously viewed elements, but our algorithms don't.

Our analysis will use the principle of deferred decisions: instead of first drawing the samples $\tilde{X}$, and then revealing the actual draws X, we will jointly draw $2n$ samples $Y_1, \ldots, Y_n, Z_1, \ldots, Z_n$, and then for each i randomly decide which of $\{Y_i, Z_i\}$ is equal to $\tilde{X}_i$ and which is equal to X_i. Formally, consider the following Deferred-Decisions procedure for drawing $X, \tilde{X}$:

1. Draw $Y_1, \ldots, Y_n$ and $Z_1, \ldots, Z_n$ independently each from $\mathcal{D}_1, \ldots, \mathcal{D}_n$.
2. For ease of notation later, for all i, relabel so that $Y_i > Z_i$.
3. Independently, flip n fair coins. If coin i is heads, set $X_i := Y_i$ and $\tilde{X}_i := Z_i$. Otherwise, set $X_i := Z_i$ and $\tilde{X}_i := Y_i$.

▶ **Observation 4.** *The output of the Deferred-Decisions procedure correctly generates $\tilde{X}_1, \ldots, \tilde{X}_n$ and $X_1, \ldots, X_n$ as independent draws from $\mathcal{D}_1, \ldots, \mathcal{D}_n$.*

Our analysis will proceed by directly comparing, for any fixed $Y_1, \ldots, Y_n, Z_1, \ldots, Z_n$, the expected reward of the gambler over the randomness in the coin flips of step three to the expected reward of the prophet over the randomness in the coin flips of step three. We note that this analysis is similar to that of the rehearsal algorithm of [5] for k-uniform matroids (whose competitive ratio is asymptotically optimal for large k), and that prior to this it was folklore knowledge that the Single Sample Algorithm achieves a competitive ratio of at least $1/4$. The novelty in our analysis is precisely nailing down the tight competitive ratio.

3.1 Analysis Setup

For a fixed $Y_1, \ldots, Y_n, Z_1, \ldots, Z_n$, sort the values into descending order, and relabel them as $W_1, \ldots, W_{2n}$. If W_j is equal to Y_i (or Z_i), we say that W_j *comes from* i, and denote this with $\text{index}(W_j) = i$. Call the *pivotal index* j^* the minimum j such that there exists an $\ell > j$ with $\text{index}(W_\ell) = \text{index}(W_j)$. That is, the pivotal index j^* is such that there are exactly $j^* - 1$ Y random variables exceeding the largest Z random variable.

Our analysis will make use of the following concept: for each $W_1, \ldots, W_{j^*-1}$, let C_j denote the outcome of the coinflip for $\text{index}(W_j)$ (which assigns either Y_i or Z_i to arrive as a sample and the other to arrive as a real value). Observe, importantly, that the random variables $C_1, \ldots, C_{j^*-1}$ are independent (because they are independent coin flips for different indices). Also importantly, observe that the random variable C_{j^*} is deterministic conditioned on $C_1, \ldots, C_{j^*-1}$ (because it is exactly the same coin flip as one of the earlier indices).

3.2 The Prophet's Expected Reward

▶ **Proposition 5.** *For fixed $W_1, \ldots, W_{2n}$ and pivotal index j^*, the prophet's expected reward, over the randomness in the coin flips of step three, is $\sum_{j=1}^{j^*-1} W_j/2^j + W_{j^*}/2^{j^*-1}$.*

Proof. Observe that the prophet achieves expected reward equal to $\max_i\{X_i\}$, so we just want to compute the probability that this is $W_1, \ldots, W_{2n}$. For each $j < j^*$, W_j is equal to $\max_i\{X_i\}$ if and only if C_j is heads, and C_ℓ is tails for all $\ell < j$ (recall that all W_j's are Y random variables for $j < j^*$). Because each of the coin flips are independent, this occurs with probability precisely $1/2^j$.

For j^*, W_{j^*} is equal to $\max_i\{X_i\}$ if and only if C_ℓ is tails for all $\ell < j^*$, and coin C_{j^*} is tails. Observe, however, that by definition of the pivotal index j^*, that when C_ℓ is tails for all $\ell < j^*$ we have C_{j^*} as tails as well (because it is the same coin as one of the first $j^* - 1$). Therefore, whenever all of the first $j^* - 1$ coins are tails, $\max_i\{X_i\} = W_{j^*}$ (and this happens with probability $1/2^{j^*-1}$).

As the first j^*-1 coins either contain some heads, or are all tails (and we have counted the prophet's reward in all such cases), we have now fully accounted for the prophet's expected reward over the randomness in the coin flips. ◀

3.3 The Single Sample Algorithm's Expected Reward

▶ **Proposition 6.** *For fixed $W_1, \ldots, W_{2n}$ and pivotal index j^*, the gambler's expected reward, over the randomness in the coin flips of step three, is at least $\sum_{j=1}^{j^*-2} W_j/2^{j+1} + W_{j^*-1}/2^{j^*-1}$.*

Proof. Consider the case where C_1 is tails. In this case, the gambler gets no reward because the threshold is higher than all revealed elements. For $j < j^* - 1$, consider next the case where $C_1, \ldots, C_j$ are heads, but C_{j+1} is tails. In this case, the gambler gets reward *at least* W_j (because the gambler will accept the first non-sample random variable exceeding W_{j+1}, and these random variables have values $W_1, \ldots, W_j$). The probability that this occurs is exactly $1/2^{j+1}$.

Consider also the case where $C_1, \ldots, C_{j^*-1}$ are all heads. Then the threshold is set at W_{j^*}, and the gambler will get at least W_{j^*-1}. This occurs with probability exactly $1/2^{j^*-1}$. ◀

3.4 Proof of Theorem 2

Proof. We can immediately see that:

$$\sum_{j=1}^{j^*-2} W_j/2^{j+1} + W_{j^*-1}/2^{j^*-1} \geq \sum_{j=1}^{j^*-1} W_j/2^{j+1} + W_{j^*}/2^{j^*}$$
$$= \frac{1}{2} \cdot \left(\sum_{j=1}^{j^*-1} W_j/2^j + W_{j^*}/2^{j^*-1} \right).$$

By Propositions 5 and 6, the right-hand side is exactly half the prophet's expected reward, conditioned on $W_1, \ldots, W_{2n}$ and j^*, and the left-hand side is exactly the gambler's expected reward (again conditioned on $W_1, \ldots, W_{2n}$ and j^*). As the gambler achieves half the prophet's expected reward for all $W_1, \ldots, W_{2n}$ and j^*, the guarantee holds in expectation as well. ◀

4 The I.I.D. Case: Optimal Ratio with Linear Samples From $\mathcal{D}$

We begin with a brief overview of the algorithm from [10] and its main features, followed by a formal specification of our algorithm.

4.1 Overview of [10] and Samples-CFHOV

The algorithm of [10] (with one slight modification due to [9]) proceeds as follows. We'll refer to this algorithm as Explicit-CFHOV.

1. As a function only of n, and independently of $\mathcal{D}$, define monotone increasing probabilities $0 \leq p_1 \leq \ldots p_n \leq 1$.
2. For all i such that $p_i \leq \delta = \varepsilon^2/n$, update $p_i := 0$ (this is the [9] modification).
3. Accept X_i if and only if $F_{\mathcal{D}}(X_i) > 1 - p_i$. Observe that this is identical to accepting X_i if and only if $X_i > \sigma_i = F_{\mathcal{D}}^{-1}(1-p_i)$. Also observe that X_i exceeds σ_i with probability p_i.

▶ **Theorem 7** ([10, 9]). *In the i.i.d. setting, Explicit-CFHOV has competitive ratio $\alpha - \varepsilon$.*[3]

[3] Without step two, the algorithm achieves a competitive ratio of α.

That is, Explicit-CFHOV sets, for each $i \in [n]$, a probability p_i independent of $\mathcal{D}$, and sets a threshold σ_i for accepting X_i which is exceeded with probability exactly p_i.

If instead of explicit access to $\mathcal{D}$, we're given m i.i.d. samples from $\mathcal{D}$, the challenge is simply that we can no longer compute $F_{\mathcal{D}}(X_i)$ exactly and run Explicit-CFHOV. The algorithm of [9] observes that $m = O(n^2)$ samples suffices to estimate the quantiles sufficiently well. Our algorithm observes that in fact $m = O(n)$ samples suffice (which is asymptotically tight, by a lower bound in [9]). Our algorithm proceeds as follows, which we call Samples-CFHOV.

1. **As a function only of n, and independently of $\mathcal{D}$**, define monotone increasing probabilities $0 \le p_1 \le \dots p_n \le 1$, exactly as in Explicit-CFHOV.
2. Round down each p_i to the nearest integer power of $(1+\varepsilon)$; we denote the rounded value by $\lfloor p_i \rfloor \in \{(1+\varepsilon)^{-1}, (1+\varepsilon)^{-2} \dots\}$.
3. Set $\tilde{p}_i := \lfloor p_i \rfloor / (1+\varepsilon)$ (that is, we have rounded down each p_i, then further divided by $(1+\varepsilon)$).
4. From our m samples, let τ_i denote the value of the $(\tilde{p}_i \cdot m)$-th highest sample.
5. Accept X_i if and only if $X_i > \tau_i$.

That is, Samples-CFHOV provides an estimate τ_i of σ_i via the m samples. Intuitively, we are trying to overestimate σ_i so that it is unlikely that Samples-CFHOV will ever choose to accept an element that Explicit-CFHOV would not. We'll prove Theorem 3 as a corollary of Theorem 8:

▶ **Theorem 8.** *For any distribution $\mathcal{D}$, with $m = O(n/\varepsilon^6)$ samples, the expected value achieved by Samples-CFHOV is at least a $(1 - O(\varepsilon))$-fraction of that of Explicit-CFHOV.*

We briefly remark that our proof of Theorem 8 actually holds for any choice of p_i's (all $\notin (0, \delta)$). That is, if Explicit-CFHOV achieves a competitive ratio of $\gamma(\vec{p})$ with a particular choice of $\vec{p}$, Samples-CFHOV achieves a competitive ratio of $\gamma(\vec{p}) - O(\varepsilon)$ (as long as each $p_i \notin (0, \delta)$).

4.2 Brief Comparison to [9]

The algorithm employed by [9] using $O(n^2)$ samples is conceptually similar in that they also wish to set thresholds τ_i such that $F_{\mathcal{D}}(\tau_i) \approx 1 - p_i$. The main difference is that we target a multiplicative $(1-\varepsilon)$-approximation to each, whereas they target an additive $1/n$-approximation for each threshold. That is, they aim to ensure that for each p_i, the threshold τ_i has $|F_{\mathcal{D}}(\tau_i) - p_i| \le 1/n$. They prove, using the Dvoretzky-Kiefer-Wolfowitz Inequality, that $O(n^2)$ samples suffice for this, then further argue that these small additive errors in the CDF don't cost much.

The same paper also establishes a barrier to moving beyond $\Omega(n^2)$ samples. Specifically, they establish that $\Omega(n^2)$ samples are necessarily just to guarantee for a *single* i with $p_i \approx 1/2$ that $|F_{\mathcal{D}}(\tau_i) - p_i| \le 1/n$. Our approach circumvents this bound by seeking a significantly weaker guarantee for such i (only that $|F_{\mathcal{D}}(\tau_i) - p_i| \le O(\varepsilon p_i)$ – see Equation (1)). So the two key differences in our approach is (a) we show that $O(n)$ samples suffice to learn the thresholds up to a multiplicative $(1+\varepsilon)$ error in the CDF and (b) establishing that this (significantly weaker) estimation suffices for a good approximation.

4.3 Proof of Theorem 8

Our proof breaks down into two simple claims. The first establishes that with high probability, our sample-based thresholds are "good." The second establishes that "good" thresholds yield a good approximation. Below, recall that $\delta = \varepsilon^2/n$.

▶ **Lemma 9.** *With $m = 12\ln(1/\varepsilon)/(\varepsilon^3\delta) = O(n/\varepsilon^6)$ samples, with probability at least $1-\varepsilon$, we have that for every i (simultaneously),*

$$\frac{p_i}{(1+\varepsilon)^3} \leq \Pr_{x\sim\mathcal{D}}[x > \tau_i] \leq p_i. \tag{1}$$

Note that Equation (1) does not reference the values of the actual elements $X_1, \ldots, X_n$ at all – it is just a claim about the thresholds $\vec{\tau}$ being set. That is, the probability $1-\varepsilon$ is taken only over the randomness in *drawing the samples* (and in particular independent of the values of the actual elements). We will call a set of thresholds "good" if they satisfy Equation (1).

Proof. Recall that τ_i is set by first rounding down p_i to $\lfloor p_i \rfloor$, then further dividing by $(1+\varepsilon)$ to $\tilde{p}_i$, then set equal to the $(\tilde{p}_i \cdot m)$-th highest of m samples. To proceed, let L_i be such that

$$\Pr_{x\sim\mathcal{D}}[x > L_i] = \lfloor p_i \rfloor.$$

Similarly, let H_i be such that

$$\Pr_{x\sim\mathcal{D}}[x > H_i] = (1+\varepsilon)^{-2}\lfloor p_i \rfloor.$$

Then (1) certainly holds whenever $L_i < \tau_i < H_i$. The remainder of the proof establishes that we are likely to have $L_i < \tau_i < H_i$ for all i.

Indeed, observe that we expect to see $\lfloor p_i \rfloor m$ samples greater than L_i. We say that $\lfloor p_i \rfloor$ is *bad* if the number of samples greater than L_i is *not* in the range $\Big[(1+\varepsilon)^{-1}(\lfloor p_i \rfloor m), (1+\varepsilon)(\lfloor p_i \rfloor m)\Big]$. Note that whenever neither $\lfloor p_i \rfloor$ nor $(1+\varepsilon)^{-2}\lfloor p_i \rfloor$ is bad, then we indeed have $L_i < \tau_i < H_i$.

Because the number of samples greater than p is an average of m independent $\{0,1\}$ random variables with expectation p, the multiplicative Chernoff bound implies that the the probability that a particular p is bad is upper bounded by:

$$\Pr[p \text{ is bad}] < e^{-\varepsilon^2 pm/3}.$$

If all $p \in \{(1+\varepsilon)^{-1}, \ldots, \delta\}$ are not bad, then our desired claim holds. Taking union bound over this $(1+\varepsilon)$-multiplicative net, we have that the probability that some $p \in \{(1+\varepsilon)^{-1}, (1+\varepsilon)^{-2} \ldots, \delta\}$ is bad is bounded by:

$$\sum_{i=0}^{O(\ln(1/\delta)/\varepsilon)} e^{-\varepsilon^2(1-\varepsilon)^{-i}\delta m/3} \leq \sum_{i=0}^{\infty} e^{-\varepsilon^2(1-\varepsilon)^{-i}\delta m/3} \leq \sum_{i=0}^{\infty} e^{-\varepsilon^3 i\delta m/6} \leq e^{-\varepsilon^3\delta m/12}$$

Above, the first term is simply a union bound over all p in this net. The second inequality follows as $(1-\varepsilon)^{-i} \geq \varepsilon i/2$ for all $\varepsilon \in (0,1)$ and $i \geq 0$. The final inequality holds (at least) when $m \geq 6/(\varepsilon^3\delta)$. Therefore, setting $m = 12\ln(1/\varepsilon)/(\varepsilon^3\delta)$ satisfies the desired claim. ◀

Next, we wish to show that whenever the thresholds are "good", the algorithm performs well in expectation. Below, let t_1 denote the stopping time of Explicit-CFHOV (i.e. the random variable denoting the element it chooses to accept), and let t_2 denote the stopping time of Samples-CFHOV.

▷ Claim 10. Conditioned on (1) holding for every i, $t_2 \geq t_1$. That is, Samples-CFHOV selects an element *later* than Explicit-CFHOV.

Proof. For every i, we have that by (1), the threshold used by Samples-CFHOV is at least the threshold used by Explicit-CFHOV. Therefore, the first time they deviate (if any) is when Explicit-CFHOV accepts an element but Samples-CFHOV does not. ◁

▶ **Lemma 11.** *Conditioned on* (1) *holding for every i, the following holds for every v:*

$$\Pr[X_{t_2} > v] \geq \frac{1}{(1+\varepsilon)^3} \Pr[X_{t_1} > v]. \tag{2}$$

Proof. We prove that (2) holds uniformly for every $i \in [n]$, i.e.

$$\Pr[(X_{t_2} > v) \wedge (t_2 = i)] \geq \frac{1}{(1+\varepsilon)^3} \Pr[(X_{t_1} > v) \wedge (t_1 = i)]. \tag{3}$$

The event $(X_{t_b} > v) \wedge (t_b = i)$ (for either $b \in \{1, 2\}$) occurs if and only if the corresponding algorithm *doesn't* stop before i, and X_i is larger than both v and the threshold set (by the corresponding algorithm). Of course, whether or not an algorithm stops before i is completely independent of X_i. We claim that the following holds on the probability that the two algorithms accept X_i (conditioned on making it to X_i). Intuitively, Claim 12 establishes that, even though the threshold τ_i is stricter than σ_i, we are still roughly as likely to accept an X_i exceeding v using τ_i versus σ_i, for all v.

▷ Claim 12. Conditioned on (1) holding for every i, then for every v and i such that $p_i \geq \delta$:

$$(1+\varepsilon)^3 \Pr[(X_i > v) \wedge (X_i > \tau_i)] \geq \Pr[(X_i > v) \wedge (X_i > \sigma_i)].$$

Proof. We consider the following three cases: perhaps $v > \tau_i$, or perhaps $v \in (\sigma_i, \tau_i)$, or perhaps $v < \sigma_i$. We claim that the following three inequalities hold:

$$v \geq \tau_i \Rightarrow \quad \Pr[(X_i > v) \wedge (X_i > \tau_i)] = \Pr[X_i > v] = \Pr[(X_i > v) \wedge (X_i > \sigma_i)].$$

$$v \in (\sigma_i, \tau_i) \Rightarrow \quad \Pr[(X_i > v) \wedge (X_i > \sigma_i)] \leq \Pr[X_i > \sigma_i] \leq \frac{\Pr[X_i > \tau_i]}{(1+\varepsilon)^3} = \frac{\Pr[(X_i > v) \wedge (X_i > \tau_i)]}{(1+\varepsilon)^3}.$$

$$v < \sigma_i \Rightarrow \quad \Pr[(X_i > v) \wedge (X_i > \sigma_i)] = \Pr[X_i > \sigma_i] \leq \frac{\Pr[X_i > \tau_i]}{(1+\varepsilon)^3} = \frac{\Pr[(X_i > \tau_i) \wedge (X_i > v)]}{(1+\varepsilon)^3}.$$

Indeed, the first implication follows as v exceeds both σ_i and τ_i. The second implication follows as $v > \sigma_i$, and then by condition (1). The third implication follows from condition (1). ◁

Claim 12 is the heart of the proof, and we can now wrap up. Observe that $\Pr[(X_{t_2} > v) \wedge (t_2 = i)] = \Pr[t_2 \geq i] \cdot \Pr[(X_i > v) \wedge (X_i > \tau_i)]$. Similarly, $\Pr[(X_{t_1} > v) \wedge (t_1 = i)] = \Pr[t_1 \geq i] \cdot \Pr[(X_i > v) \wedge (X_i > \sigma_i)]$. By the work above, $(1+\varepsilon)^3 \Pr[(X_i > v) \wedge (X_i > \tau_i)] \geq \Pr[(X_i > v) \wedge (X_i > \sigma_i)]$. By Claim 10, $\Pr[t_2 \geq i] \geq \Pr[t_1 \geq i]$. Therefore, we've proven the desired claim for every $i \in [n]$. As $\Pr[X_{t_b} > v] = \sum_i \Pr[(X_{t_b} > v) \wedge (t_b = i)]$, this completes the proof of Lemma 11. ◀

Proof of Theorem 8. The proof of Theorem 8 now follows as a direct corollary of Lemmas 9 and 11. Lemma 11 asserts that whenever the thresholds are "good", Samples-CFHOV achieves at least a $1/(1+\varepsilon)^3$ fraction of the expected reward of Explicit-CFHOV (this is because the expected reward of Samples-CFHOV is simply $\int_0^\infty \Pr[X_{t_2} > v]dv \geq \int_0^\infty \Pr[X_{t_1} > v]dv/(1+\varepsilon)^3$, and the expected reward of Explicit-CFHOV is precisely $\int_0^\infty \Pr[X_{t_2} > v]dv$). Lemma 9 asserts that the thresholds are good with probability at least $1-\varepsilon$. So together, Samples-CFHOV achieves at least a $\frac{1-\varepsilon}{(1+\varepsilon)^3}$ of the expected reward of Explicit-CFHOV. ◀

References

1 Melika Abolhassani, Soheil Ehsani, Hossein Esfandiari, MohammadTaghi Hajiaghayi, Robert D. Kleinberg, and Brendan Lucier. Beating 1-1/e for ordered prophets. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 61–71, 2017. doi:10.1145/3055399.3055479.

2 Marek Adamczyk and Michal Wlodarczyk. Random Order Contention Resolution Schemes. In *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 790–801, 2018. doi:10.1109/FOCS.2018.00080.

3 Saeed Alaei. Bayesian Combinatorial Auctions: Expanding Single Buyer Mechanisms to Many Buyers. In *the 52nd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 512–521, 2011. doi:10.1109/FOCS.2011.90.

4 Nima Anari, Rad Niazadeh, Amin Saberi, and Ali Shameli. Nearly Optimal Pricing Algorithms for Production Constrained and Laminar Bayesian Selection. In *Proceedings of the 2019 ACM Conference on Economics and Computation, EC 2019, Phoenix, AZ, USA, June 24-28, 2019.*, pages 91–92, 2019. doi:10.1145/3328526.3329652.

5 Pablo Daniel Azar, Robert Kleinberg, and S. Matthew Weinberg. Prophet Inequalities with Limited Information. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 1358–1377, 2014. doi:10.1137/1.9781611973402.100.

6 Yossi Azar, Ashish Chiplunkar, and Haim Kaplan. Prophet Secretary: Surpassing the 1-1/e Barrier. In *Proceedings of the 2018 ACM Conference on Economics and Computation, Ithaca, NY, USA, June 18-22, 2018*, pages 303–318, 2018. doi:10.1145/3219166.3219182.

7 Shuchi Chawla, Jason D. Hartline, David L. Malec, and Balasubramanian Sivan. Multi-Parameter Mechanism Design and Sequential Posted Pricing. In *the 42nd ACM Symposium on Theory of Computing (STOC)*, 2010.

8 Shuchi Chawla, J. Benjamin Miller, and Yifeng Teng. Pricing for Online Resource Allocation: Intervals and Paths. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 1962–1981, 2019. doi:10.1137/1.9781611975482.119.

9 José R. Correa, Paul Dütting, Felix A. Fischer, and Kevin Schewior. Prophet Inequalities for I.I.D. Random Variables from an Unknown Distribution. In *Proceedings of the 2019 ACM Conference on Economics and Computation, EC 2019, Phoenix, AZ, USA, June 24-28, 2019.*, pages 3–17, 2019. doi:10.1145/3328526.3329627.

10 José R. Correa, Patricio Foncea, Ruben Hoeksma, Tim Oosterwijk, and Tjark Vredeveld. Posted Price Mechanisms for a Random Stream of Customers. In Constantinos Daskalakis, Moshe Babaioff, and Hervé Moulin, editors, *Proceedings of the 2017 ACM Conference on Economics and Computation, EC '17, Cambridge, MA, USA, June 26-30, 2017*, pages 169–186. ACM, 2017. doi:10.1145/3033274.3085137.

11 José R. Correa, Raimundo Saona, and Bruno Ziliotto. Prophet Secretary Through Blind Strategies. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 1946–1961, 2019. doi:10.1137/1.9781611975482.118.

12 Paul Duetting, Michal Feldman, Thomas Kesselheim, and Brendan Lucier. Prophet Inequalities Made Easy: Stochastic Optimization by Pricing Non-Stochastic Inputs. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 540–551, 2017. doi:10.1109/FOCS.2017.56.

13 Paul Dütting and Thomas Kesselheim. Posted Pricing and Prophet Inequalities with Inaccurate Priors. In *Proceedings of the 2019 ACM Conference on Economics and Computation, EC 2019, Phoenix, AZ, USA, June 24-28, 2019.*, pages 111–129, 2019. doi:10.1145/3328526.3329576.

14 Soheil Ehsani, MohammadTaghi Hajiaghayi, Thomas Kesselheim, and Sahil Singla. Prophet Secretary for Combinatorial Auctions and Matroids. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 700–714, 2018. doi:10.1137/1.9781611975031.46.

15 Hossein Esfandiari, MohammadTaghi Hajiaghayi, Vahid Liaghat, and Morteza Monemizadeh. Prophet Secretary. In *Algorithms - ESA 2015 - 23rd Annual European Symposium, Patras, Greece, September 14-16, 2015, Proceedings*, pages 496–508, 2015. doi:10.1007/978-3-662-48350-3_42.

16 Moran Feldman, Ola Svensson, and Rico Zenklusen. Online Contention Resolution Schemes. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1014–1033, 2016. doi:10.1137/1.9781611974331.ch72.

17 Oliver Göbel, Martin Hoefer, Thomas Kesselheim, Thomas Schleiden, and Berthold Vöcking. Online Independent Set Beyond the Worst-Case: Secretaries, Prophets, and Periods. In *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part II*, pages 508–519, 2014. doi:10.1007/978-3-662-43951-7_43.

18 Nikolai Gravin and Hongao Wang. Prophet Inequality for Bipartite Matching: Merits of Being Simple and Non Adaptive. In *Proceedings of the 2019 ACM Conference on Economics and Computation, EC 2019, Phoenix, AZ, USA, June 24-28, 2019.*, pages 93–109, 2019. doi:10.1145/3328526.3329604.

19 Theodore P. Hill and Robert P. Kertz. Comparisons of stop rule and supremum expectations of i.i.d. random variables. *The Annals of Probability*, 10:336–345, 1982.

20 Robert P. Kertz. Stop rule and supremum expectations of i.i.d. random variables: a complete comparison by conjugate duality. *Journal of Multivariate Analysis*, 19:88–112, 1986.

21 Robert Kleinberg and S. Matthew Weinberg. Matroid prophet inequalities. In *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 - 22, 2012*, pages 123–136, 2012. doi:10.1145/2213977.2213991.

22 Ulrich Krengel and Louis Sucheston. On Semiamarts, amarts, and processes with finite value. *Advances in Probability and Related Topics*, 4:197–266, 1978.

23 Aviad Rubinstein. Beyond matroids: secretary problem and prophet inequality with general constraints. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 324–332, 2016. doi:10.1145/2897518.2897540.

24 Aviad Rubinstein and Sahil Singla. Combinatorial Prophet Inequalities. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 1671–1687, 2017. doi:10.1137/1.9781611974782.110.

25 Ester Samuel-Cahn. Comparison of threshold stop rules and maximum for independent nonnegative random variables. *Annals of Probability*, 12(4):1213–1216, 1984.

Implementation in Advised Strategies: Welfare Guarantees from Posted-Price Mechanisms When Demand Queries Are NP-Hard

Linda Cai
Princeton University, Princeton, NJ, USA
tcai@cs.princeton.edu

Clay Thomas
Princeton University, Princeton, NJ, USA
claytont@cs.princeton.edu

S. Matthew Weinberg
Princeton University, Princeton, NJ, USA
smweinberg@princeton.edu

Abstract

State-of-the-art posted-price mechanisms for submodular bidders with m items achieve approximation guarantees of $O((\log\log m)^3)$ [1]. Their truthfulness, however, requires bidders to compute an NP-hard demand-query. Some computational complexity of this form is unavoidable, as it is NP-hard for truthful mechanisms to guarantee even an $m^{1/2-\varepsilon}$-approximation for any $\varepsilon > 0$ [21]. Together, these establish a stark distinction between computationally-efficient and communication-efficient truthful mechanisms.

We show that this distinction disappears with a mild relaxation of truthfulness, which we term implementation in advised strategies. Specifically, *advice* maps a tentative strategy either to that same strategy itself, or one that dominates it. We say that a player follows advice *as long as they never play actions which are dominated by advice*. A poly-time mechanism guarantees an α-approximation in implementation in advised strategies if there exists advice (which runs in poly-time) for each player such that an α-approximation is achieved whenever all players follow advice. Using an appropriate bicriterion notion of approximate demand queries (which can be computed in poly-time), we establish that (a slight modification of) the [1] mechanism achieves the same $O((\log\log m)^3)$-approximation in implementation in advised strategies.

2012 ACM Subject Classification Theory of computation → Algorithmic mechanism design; Theory of computation → Solution concepts in game theory

Keywords and phrases Combinatorial auctions, Posted-Price mechanisms, Submodular valuations, Incentive compatible

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.61

Funding *S. Matthew Weinberg*: Supported by NSF CCF-1717899.

Acknowledgements We thank Sepehr Assadi, Sahil Singla and the anonymous reviewers for helpful discussions.

1 Introduction

Combinatorial auctions have been at the forefront of Algorithmic Game Theory since its inception as a lens through which to study the relative power of *algorithms* for honest agents versus *mechanisms* for strategic agents. Specifically, there are n buyers with combinatorial valuations $v_1(\cdot), \ldots, v_n(\cdot)$ over subsets of m items, and the designer wishes to allocate the items so as to maximize the *welfare*, $\sum_i v_i(S_i)$ (where S_i is the set allocated to bidder i). Without concern for computation/communication/etc., the celebrated Vickrey-Clarke-Groves

mechanism [41, 9, 28] provides a black-box reduction from precisely optimal mechanisms to precisely optimal algorithms. Of course, precisely optimal algorithms are NP-hard and require exponential communication in most settings of interest (for example, when buyers have submodular valuations over the items, which we'll take as the running example for the rest of the introduction), rendering VCG inapplicable. On the algorithmic front, poly-time/poly-communication constant-factor approximation algorithms are known [38, 30, 33, 4, 42, 24] and a central direction in algorithmic mechanism design is understanding whether these guarantees are achievable by computationally/communication efficient truthful mechanisms as well.

From the communication complexity perspective, this problem is still wide open: state-of-the-art truthful mechanisms guarantee an $O((\log\log m)^3)$-approximation [1], yet no lower bounds separate achievable guarantees of mechanisms from algorithms (that is, it could very well be the case that truthful, poly-communication mechanisms can achieve the same guarantees as poly-communication algorithms). From the computational perspective, however, a landmark result of Dobzinski and Vondrak establishes that for all $\varepsilon > 0$, an $m^{1/2-\varepsilon}$-approximation is NP-hard for truthful mechanisms [21]. As poly-time algorithms guarantee a $e/(e-1)$-approximation [42], this establishes a strong separation between computationally-efficient algorithms and computationally-efficient truthful mechanisms.

So while the communication perspective has seen exciting progress in recent years [15, 3], the computational perspective is generally considered fully resolved. In this paper, we present a new dimension to the computational perspective, motivated by the following two examples. Consider first the truthful mechanism of [1]. The core of the mechanism is a posted-price mechanism: it visits each bidder one at a time, posts a price p_j on each remaining item j, and offers the option to purchase any set S of items at total price $\sum_{j\in S} p_j$ (see Section 4 and Appendix B for a full description of their mechanism, which also includes randomization, pre-procesesing, and learning). The auxiliary parts of the mechanism run in poly-time,[1] and the offered prices can also be computed in poly-time. While it might sound like this mechanism should be poly-time, the catch is that it's NP-hard for the buyer find their utility-maximizing set, called a *demand query*. Therefore, the mechanism is either not truthful (because the buyers do not select their utility-maximizing sets), or requires solving an NP-hard problem (because the buyers pick their favorite sets). Still, the analysis of [1] and related mechanisms [12, 18, 31, 11, 26, 14, 22, 14] seems fairly robust, suggesting that perhaps they should maintain their guarantees under reasonable strategic behavior. Indeed, the focus of this paper is a novel solution concept (described below) under which the [1] $O((\log\log m)^3)$-approximation is maintained in polynomial time.

A New Solution Concept: Implementation in Advised Strategies. To get intuition for our solution concept, consider the following example due to [39]: there is only a single buyer, but the buyer can receive only k of the m items (this is the one-buyer case of Combinatorial Public Projects). Since there is just a single buyer, the obvious mechanism for the designer simply allows the buyer to pick any set of size k for free (call this the "Set-For-Free" mechanism). The same catch is that it is NP-hard for the buyer to pick their favorite set, so Set-For-Free is again not truthful (because the buyer picks a suboptimal set) or solving an NP-hard problem (because they find their favorite set). In fact, [39] establishes that it is NP-hard for truthful mechanisms to achieve a $m^{1/2-\varepsilon}$-approximation for any $\varepsilon > 0$. Algorithmically, a poly-time $e/(e-1)$-approximation is known [35], providing again a strong separation.

[1] Rather, they can be slightly modified to run in poly-time – see Section 4 and Appendix B.

We ask instead: what should one reasonably expect to happen if a strategic buyer participated in Set-For-Free? Consider the set S output by the poly-time algorithm of [35]. It is certainly reasonable for the buyer to select some set $T \neq S$: perhaps a different heuristic finds a better set. But it seems irrational for the buyer to select some set T with $v(T) < v(S)$. We therefore pose that there should be *some* reasonable solution concept under which Set-For-Free guarantees an $e/(e-1)$-approximation. Indeed, Set-For-Free guarantees an $e/(e-1)$-approximation under our proposed "implementation in advised strategies."

Formally, we will think of Set-For-Free as simply asking the buyer to report a set of size at most k, and then awarding them that set for free. In addition, the designer provides *advice*: a Turing machine $A(\cdot,\cdot)$ which takes as input the buyer's valuation $v(\cdot)$ (possibly as a circuit/Turing machine itself, or accessing it via value queries) and a tentative set T, then recommends a set S to purchase that is at least as good as T. Specifically in Set-For-Free, we will think of the advice as running the [35] approximation algorithm to get a set S' and outputting $S := \arg\max\{v(T), v(S')\}$. We say that a bidder *follows advice* if they select a set S with $A(v, S) = S$. The idea is that it seems irrational for the buyer to select a set without this property, when the advice gives a poly-time algorithm to improve it.

For a general mechanism, we think of advice as a Turing machine which takes as input the current state of the mechanism, the buyer's valuation, and a tentative action, then advises an (maybe the same, maybe different) action to take. Importantly, we say that advice is *useful* if for all strategies s, either the advice maps s to itself, or to another strategy which dominates it (see Section 2 for full definition). Intuitively, this suggests that it is irrational for a buyer to use a strategy which advice does not map to itself. We postpone to Section 2 a formal definition of what it means to follow advice, but note here that our definition is a natural relaxation of dominant strategies: if a mechanism has a dominant strategy s, then the only strategy which follows advice that recommends s is s itself. We say that mechanism guarantees a poly-time α-approximation in implementation in advised strategies whenever the mechanism itself concludes in poly-time, and there exists poly-time advice A such that an α-approximation is guaranteed whenever all bidders follow advice A. Again, note that the assumption on bidder behavior is quite permissive: they need not play a dominant, or even undominated strategy. We just assume they do not play a strategy which the advice itself dominates.

Advice via Approximate Demand Queries. We now revisit posted-price mechanisms, which achieve approximation guarantees of $O((\log\log m)^3)$, but whose truthfulness requires buyers to compute NP-hard demand queries. Instead, we pursue guarantees in implementation in advised strategies. For a posted-price mechanism with price vector $\mathbf{p}$, our proposed advice will take as input a tentative set T for purchase, and the buyer's valuation $v(\cdot)$, and recommend a set S guaranteeing $v(S) - \mathbf{p}(S) \geq v(T) - \mathbf{p}(T)$.[2] More specifically, our advice will compute a tentative recommendation S' independently of T, then simply recommend $\arg\max\{v(S') - \mathbf{p}(S), v(T) - \mathbf{p}(T)\}$. Again, our behavioral assumption does not assume that the buyer will purchase the set S' tentatively recommended, just that they will not irrationally ignore the advice in favor of a lower-utility set.

The remaining challenge is now to find concrete advice under which the [1] approximation guarantees are maintained. A first natural attempt is simply an approximate demand oracle: have a tentative recommendation S' with $v(S') - \mathbf{p}(S') \geq c \cdot \max_T\{v(T) - \sum_{j\in T} p_j\}$. Unfortunately, even this is NP-hard for any $c = \Omega(1/m^{1-\varepsilon})$ (for any $\varepsilon > 0$) [25]. Instead,

[2] Throughout the paper we will use notation $\mathbf{p}(S) := \sum_{i\in S} \mathbf{p}_i$.

we design bicriterion approximate demand oracles. Specifically, for some $c, d < 1$, a (c, d)-approximate demand oracle produces a set S' satisfying $v(S') - \mathbf{p}(S') \geq c \cdot \max_T\{v(T) - \mathbf{p}(T)/d\}$. That is, the guaranteed utility is at least an c-fraction of the optimum *if all prices were increased by a factor of* $1/d$. We design a simple greedy $(1/2, 1/2)$-approximation in poly-time (based on [33]), and further establish that the [1] mechanism maintains its approximation guarantee up to an additional $\min\{c, d\}$ factor when bidders follow advice provided in this manner by a (c, d)-approximate demand oracle. This allows us to conclude the main result of this paper:

▶ **Theorem 1.** *There exists a poly-time mechanism which achieves an* $O((\log\log m)^3)$*-approximation to the optimal welfare for any number of submodular buyers in implementation in advised strategies.*

1.1 Roadmap

Combinatorial auctions have a long history within AGT, along with related problems like Combinatorial Public Projects. The most related work is overviewed in Section 1, but we provide additional context in Section 1.2. Section 2 contains a formal definition of implementation in advised strategies, repeating our motivating examples and providing additional discussion.

In Section 3, we design our poly-time $(1/2, 1/2)$-approximate demand oracles for submodular valuations. The proof is fairly simple, but we include the complete proof in the body for readers unfamiliar with [33] (readers familiar with [33] will find the outline simliar).

In Section 4, we establish that existing posted-price mechanisms maintain their approximation guarantees as long as buyers follow advice given by $(\Omega(1), \Omega(1))$-approximate demand oracles. We include a complete analysis of the main lemma of [26] concerning "fixed price auctions" for readers unfamiliar with this aspect (readers familiar with [26] will find the outline similar). We defer all aspects of the analysis of [1] to Appendix B.

Finally, in Section 5, we design simple poly-time $(\frac{1}{\sqrt{m}}, \frac{1}{1+\sqrt{m}})$-approximate demand oracles for subadditive valuations. This is essentially the best possible even for XOS valuations,[3] due to known lower bounds on welfare-maximization with value queries [17] and the results of Section 4.

1.2 Discussion and Related Work

There is a vast literature studying combinatorial auctions, which we will not attempt to overview in its entirety here. We summarize the lines of work most relevant to ours below.

VCG-based Mechanisms. The Vickrey-Clarke-Groves mechanism provides a poly-time/poly-communication black-box reduction from precise welfare maximization with a truthful mechanism to precise welfare maximization with an algorithm for any class of valuation functions [41, 9, 28]. The same reduction applies for "maximal-in-range" approximation algorithms, but this approach provably cannot achieve sub-polynomial approximations in poly-time (unless P = NP) or subexponential communication [8, 7, 10]. Still, in some regimes (e.g. arbitrary monotone valuations with poly-time/poly-communication, or XOS valuations with poly-time), no better than a $\Theta(\sqrt{m})$-approximation is achievable even with honest players, and a $\Theta(\sqrt{m})$-approximation is achievable via truthful VCG-based mechanisms [32, 17].

[3] More precisely, it is unconditionally hard to obtain a $(1/m^{1/2-\varepsilon}, 1/m^{1/2-\varepsilon})$-approximate demand query for XOS valuations using polynomially many black-box value queries.

Combinatorial Auctions in the Computational Model. Taking the above discussion into account, valuation function classes above XOS (including subadditive, or arbitrary monotone) are "too hard", in the sense that $\Theta(\sqrt{m})$ is the best approximation achievable in poly-time even without concern for incentives, and this guarantee can be matched by VCG-based truthful mechanisms. Other valuation function classes like Gross Substitutes are "easy", in the sense that precise welfare maximization is achievable in poly-time, so the VCG mechanism is poly-time as well. Submodular valuations are a fascinating middle ground. Here, an algorithmic $e/(e-1)$-approximations is possible in poly-time (and it is NP-hard to do better) [42, 34, 19], but a long series of works establishes that it is NP-hard for a truthful mechanism to even achieve an $m^{1/2-\varepsilon}$-approximation (for any $\varepsilon > 0$) [36, 7, 8, 10, 13, 20, 19, 21].

On this front, our work establishes that significantly better ($O((\log\log m)^3)$) guarantees are achievable with a slightly relaxed solution concept, matching the state-of-the-art in the communication model. In addition to the standalone motivation for the communication model discussed below, our work establishes that resolving key open questions (e.g. is there a constant-factor approximation in the communication model for submodular valuations?) may have strong implications in the computational model as well (via implementation in advised strategies).

Combinatorial Auctions in the Communication Model. In the communication model, only arbitrary monotone valuations are "too hard" per the above discussion: a 2-approximation is possible for subadditive valuations, and a $\frac{1}{1-(1-1/n)^n}$-approximation is possible for XOS valuations in poly-communication, both of which are tight [24, 17, 23]. Yet, no truthful constant-factor approximations are known (the state-of-the-art is $O((\log\log m)^3)$ for submodular/XOS [1] or $O(\log m \log\log m)$ for subadditive [12]). On the lower bounds side, *no* separations are known between the approximation ratios of truthful mechanisms and non-truthful algorithms using $\text{poly}(n, m)$ communication, even for *deterministic* truthful mechanisms (where the $O(\sqrt{m})$-approximation of [17] remains the state-of-the-art). Determining whether such a separation exists is the central open problem of this agenda (e.g. [15, 3]).

On this front, our work in some sense unifies the state-of-the-art for submodular valuations in the communication and computational models via implementation in advised strategies. So in addition to the standalone interest in establishing (or disproving) a separation in the communication model, such a result will now likely have implications in the computational model as well.

Posted Price Mechanisms. Posted-price mechanisms are ubiquitous in mechanism design, owing to their simplicity and surprising ability to guarantee good approximations through a variety of lenses [31, 11, 26, 14, 22, 1]. Very recent work also establishes posted-price mechanisms as the unique class of mechanisms which is "strongly obviously strategy-proof" [37]. One minor downside of these mechanisms is that they require buyers to compute NP-hard demand queries. Our work formally mitigates this downside under implementation in advised strategies. For example, our work immediately extends the price of anarchy bounds of [11] to hold in equilibria which are poly-time learnable for submodular buyers (previously the equilibria required computation of demand queries).

Combinatorial Public Projects. Combinatorial Public Projects is a related problem, which has also received substantial attention. Here, the designer may select any set of k items, but *every* bidder receives all k items (instead of the bidders each receiving disjoint sets of

items, as in auctions). We used the single submodular bidder Combinatorial Public Projects problem as a motivating example due to the hardness results established in [39]: no poly-time truthful mechanism can achieve an approximation ratio better than $O(\sqrt{m})$ (unless P = NP). Contrast this with the general communication model, where Set-For-Free is truthful and precisely optimizes welfare. Follow up work of [6] establishes that while the single-bidder CPPP is inapproximable only because demand queries are NP-hard, the strong multi-bidder inapproximability results of [36, 8] hold even when bidders have access to demand oracles.

[31, 14, 1] already establish that the aforementioned computational separations no longer hold with demand oracles (so the story for combinatorial auctions differs greatly from combinatorial public projects). Our work further establishes that the same guarantees are achievable in truly polynomial time, under a relaxed solution concept.

(c, d)-Approximate Demand Oracles. To the best of our knowledge, bicriterion approximate demand oracles have not previously been considered. However, prior work regarding approximation algorithms for nonnegative submodular functions with bounded curvature subject to a matroid constraint designs a randomized $(1 - 1/e, 1 - 1/e)$-approximate demand oracle for submodular functions (under a different name) [40, 27, 29]. We include our $(1/2, 1/2)$-approximate demand oracles for submodular functions as they are deterministic and significantly simpler (note also that determinism makes our related solution concepts significantly cleaner).

Related Solution Concepts. The most related existing solution concept to ours is Algorithmic Implementation in Undominated Strategies [2]. Here, a mechanism achieves an α-approximation if whenever players play *any undominated strategy*, the resulting allocation is an α-approximation (and for all dominated strategies, there is a poly-time algorithm to find an undominated one which dominates it). For posted-price mechanisms, this solution concept has no bite as the only undominated strategy is to pick the utility-maximizing set, and it is not possible to find this set in poly-time. We establish in Observation 10 that implementation in advised strategies is a relaxation of algorithmic implementation in undominated strategies. The purpose of our novel solution concept is to have bite even when it is NP-hard to find an undominated strategy.

2 Implementation in Advised Strategies

Motivated by the example of [39], we first relax the requirement that a mechanism be truthful, instead requiring that a mechanism achieve its approximation guarantee whenever players behave in a manner which is not clearly irrational. Before proposing our formal definition, let's examine it applied to two motivating examples.

Example One: [39]. There is a single buyer with submodular valuation function $v(\cdot)$. The seller's mechanism (Set-For-Free) allows the buyer to state any set of size k, and receive that set for free. Recall that it is NP-hard for the buyer to find their favorite set of size k – so if the mechanism is to be truthful, it is not poly-time (unless P = NP). The buyer can indeed find an $e/(e-1)$-approximation in poly-time [35], but assuming the buyer will run this particular algorithm (or any specific approximation algorithm) is perhaps too strong an assumption.

Instead, we assume simply that the buyer picks a set yielding *at least as much utility* as this $e/(e-1)$-approximation. Specifically, we will think of the designer as providing a poly-time mechanism (Set-For-Free – the buyer states a set of size k and receives that set

for free), and a poly-time *advice algorithm* (takes as input the buyer's valuation function $v(\cdot)$, and a tentative set S, then runs the $e/(e-1)$-approximation to get a set T and outputs $\arg\max\{v(S), v(T)\}$, tie-breaking for S). Intuitively, we are claiming that it is certainly rational for the buyer to purchase a set other than T, but that it is irrational to purchase a set with $v(S) < v(T)$.

Example Two: Posted-Price Mechanisms. Consider now any posted-price mechanism. Again, we think of the designer as providing a poly-time mechanism (for all i, computes in poly-time a price vector to offer bidder i, based on interactions with bidders $< i$), along with a poly-time advice algorithm (takes as input the buyer's valuation function $v(\cdot)$, a tentative set S, computes in poly-time a set T and outputs $\arg\max\{v(S) - \mathbf{p}(S), v(T) - \mathbf{p}(T)\}$, again tie-breaking for S). Again note that we are claiming that it may be rational for the buyer to purchase a set other than T, but that it is irrational to purchase a set yielding lower utility than T.

Importantly, we emphasize that we assume the buyer achieves at least as much *utility* as recommended (a well-justified behavioral assumption, although not particularly convenient for welfare guarantees), and *not* that the buyer picks a set guaranteeing them at least as much welfare (more convenient for analyzing welfare guarantees, but an unmotivated assumption). With these instantiations in mind, we now build up language to present our formal definition.

▶ **Definition 2** (Mechanism as an Extended Form Game)**.** *Formally, a* mechanism *is just an extended form game: at every state, it solicits actions from one or more players and (possibly randomly) updates its state. With probability one, the mechanism eventually reaches a terminal state, and (possibly randomly) outputs an allocation of items and payments charged.*

A mechanism is poly-time *if every state update is poly-time computable, and the mechanism reaches a terminal state with probability one after* $\text{poly}(n, m)$ *updates.*

▶ **Definition 3** (Strategies, Utility, and Dominance)**.** *A* strategy $s(\cdot)$ *for player i is simply a mapping from the current state x of the mechanism to an action $s(x)$.*

We denote by $u_i(v_i, \vec{s})$ the expected utility of player i when their valuation function is $v_i(\cdot)$ and the players use strategy profile $\vec{s}$.

Strategy $s(\cdot)$ dominates *strategy $s'(\cdot)$ for player i with valuation $v_i(\cdot)$ if for all s_{-i}, $u_i(v_i, s_i; \vec{s}_{-i}) \geq u_i(v_i, s'_i; \vec{s}_{-i})$, and there exists an $\vec{s}_{-i}$ such that $u_i(v_i, s_i; \vec{s}_{-i}) > u_i(v_i, s'_i; \vec{s}_{-i})$.*

▶ **Definition 4** (Advice)**.** *Advice is a function $A(\cdot, \cdot, \cdot)$ which takes as input the valuation $v_i(\cdot)$ of a player, a state x of a mechanism, and a tentative action a, then (possibly randomly) outputs an advised action $A(v_i, x, a)$. We say that advice is poly-time if it is poly-time computable.*

Observe that every advice $A(\cdot, \cdot, \cdot)$, valuation function $v_i(\cdot)$, and tentative strategy $s(\cdot)$ induces a strategy $A^{v_i,s}(\cdot)$ with $A^{v_i,s}(x) := A(v_i, x, s(x))$.

▶ **Definition 5** (Useful Advice)**.** *We say that advice A is* useful *if:*

1. *For all $s(\cdot)$, and all $v_i(\cdot)$, either $A^{v_i,s}(\cdot) = s(\cdot)$, or $A^{v_i,s}(\cdot)$ dominates $s(\cdot)$ (for valuation $v_i(\cdot)$).*
2. *For all $s(\cdot)$ and all $v_i(\cdot)$, $A^{v_i,A^{v_i,s}}(\cdot) = A^{v_i,s}(\cdot)$ (Advice is idempotent – applying advice to $s(\cdot)$ twice is the same as applying it once).*

Intuitively, Property 1 guarantees that the bidder should indeed follow advice given by A instead of whatever strategy they had originally planned. Property 2 guarantees essentially that the bidder does not get "stuck" in an exponentially-long loop trying to repeatedly improve their strategy via advice (because the loop terminates after one iteration). Let us briefly observe the following implication of our definition (which we explore further when revisiting our two main examples):

▶ **Observation 6.** *Let $A(\cdot,\cdot,\cdot)$ be useful, and let $A(v_i, x, a) \neq a$. Then for all s such that $s(x) = a$, $A^{v_i,s}(\cdot)$ dominates $s(\cdot)$.*

Intuitively, useful advice A separates strategies into *advised* strategies (where $A^{v_i,s}(\cdot) = s(\cdot)$), and *ill-advised* strategies (where $A^{v_i,s}(\cdot)$ dominates $s(\cdot)$). We say that a bidder follows advice if they use an advised strategy.

▶ **Definition 7** (Follows Advice)**.** *We say that $s(\cdot)$ is advised for $v_i(\cdot)$ under A if $A^{v_i,s}(\cdot) = s(\cdot)$. A bidder with valuation $v_i(\cdot)$* follows advice *A if they use a strategy which is advised under A.*

Intuitively, we are claiming that it is irrational for a player to use an ill-advised strategy $s(\cdot)$ (because they could instead use the strategy $A^{v_i,s}(\cdot)$, which dominates it).

▶ **Definition 8** (Implementation in Advised Strategies)**.** *We say that a mechanism M guarantees an α-approximation in* implementation in advised strategies *with advice A if A is useful and for all $v_1(\cdot), \ldots, v_n(\cdot)$, if all players follow advice A, the resulting allocation in M achieves (expected) welfare at least $\alpha \cdot \text{OPT}(v_1, \ldots, v_n)$. If both M and A are poly-time, we say that M guarantees a poly-time α-approximation in implementation in advised strategies (without referencing A).*

Let's now briefly revisit our two examples through the formal definitions. First, recall the single-bidder mechanism Set-For-Free. Set-For-Free is poly-time: it takes as input a set and simply outputs that set, and terminates after one iteration. Consider the advice algorithm which takes as input $v(\cdot)$, and a set S, then runs the $e/(e-1)$-approximation algorithm of [35] to get a set T and outputs $\arg\max\{v(S), v(T)\}$, tie-breaking for S (the mechanism has only a single non-terminal state, so this completely specifies the advice). Then the advice indeed recommends a dominating strategy whenever it recommends $T \neq S$, and is idempotent (tie-breaking in favor of S is subtly necessary for this claim – if instead the advice tie-broke for T, then it might recommend an action distinct from S which does not dominate it). Moreover, observe that the bidder follows advice if and only if they choose a higher value set than produced by the algorithm, and therefore we're guaranteed a $e/(e-1)$-approximation whenever the bidder follows advice.

Posted-price mechanisms with poly-time computable prices are poly-time: they iteratively take as input a set from bidder i, assign that set to bidder i, then run a poly-time computation to determine the prices for bidder $i+1$. They terminate after n iterations. We will later design poly-time *approximate demand oracles*, which take as input $v_i(\cdot)$ and the price vector $\mathbf{p}$, and output a recommended set $D(v_i, \mathbf{p})$ in poly-time. For a posted-price mechanism, there are multiple non-terminal states, each corresponding to a different price vector $\mathbf{p}$. Our advice, on input $v_i, S, \mathbf{p}$, will advise the set $\arg\max\{v(S) - \mathbf{p}(S), v(D(v_i, \mathbf{p})) - \mathbf{p}(D(v_i, \mathbf{p}))\}$, again tie-breaking in favor of S.[4] If a strategy follows advice, it must, for all $\mathbf{p}$, select a set S satisfying $v_i(S) - \mathbf{p}(S) \geq v_i(D(v_i, \mathbf{p})) - \mathbf{p}(D(v_i, \mathbf{p}))$. It's not immediately clear why this property should provide meaningful welfare guarantees, but we will later argue that the right pairing of mechanism and notion of approximate demand oracle achieves polynomial-time welfare guarantees which match state-of-the-art guarantees for computationally-unbounded bidders.

[4] Subtly, note that tie-breaking in favor of T would violate the definition of usefulness, via Observation 6. Indeed, consider the strategy $s(\cdot)$ which purchases a utility-maximizing set on all prices $\neq \mathbf{p}$, and set S on $\mathbf{p}$. Then any advice which tie-breaks in favor of T on prices $\mathbf{p}$ does not map $s(\cdot)$ to itself, and does not dominate $s(\cdot)$ (because it generates the same utility on prices $\mathbf{p}$, and cannot generate strictly higher utility on any other prices, because $s(\cdot)$ is optimal).

Brief Discussion of Definitions. We chose our definitions with the goal of (a) providing a strict relaxation of truthfulness, and (b) doing so in a way that permits all rational behavior while (c) still eliminating enough irrational behavior to guarantee good welfare. We include in Appendix A a brief example motivating our decision to think of advice as *improving a given strategy* as opposed to *outright proposing a replacement strategy.* We conclude by establishing that implementation in advised strategies is a strict relaxation of truthfulness and algorithmic implementation in undominated strategies.

▶ **Observation 9.** *If player i with valuation v_i has a dominant strategy $s^*(\cdot)$ in mechanism M, and $A^{v_i,s}(\cdot) := s^*(\cdot)$ for all $s(\cdot)$, then the only strategy which follows advice is $s^*(\cdot)$ itself.*

▶ **Observation 10.** *Let M achieve an α-approximation in algorithmic implementation in undominated strategies. Then M achieves an α-approximation in implementation in advised strategies.*

Proof. Recall that algorithmic implementation in undominated strategies implies for all $v_i(\cdot)$ the existence of a poly-time function $f_{v_i}(\cdot)$ which takes as input a strategy $s(\cdot)$ and outputs a new strategy $f_{v_i}(s)$ such that if $s(\cdot)$ is dominated, $f_{v_i}(s)$ dominates s. Extend this so that $f_{v_i}(s) := s$ when $s(\cdot)$ is undominated. Now simply define $A^{v_i,s}(\cdot) := f_{v_i}(s)$, and then A is useful and poly-time. Moreover, the advised strategies are exactly the undominated strategies, so because M achieves an α-approximation whenever all players use undominated strategies, it also achieves an α-approximation whenever all players follow advice. ◀

Of course, the whole point of our definitions is that the designer may not be able to find a dominant (or even undominated) strategy in poly-time, and implementation in advised strategies has bite even under these circumstances while the previous solution concepts do not.

3 Approximate demand oracles

In this section, we develop our poly-time advice for posted-price mechanisms in the form of an *approximate demand oracle*. Recall that a demand oracle for valuation function $v(\cdot)$ takes as input a price vector $\mathbf{p}$ and outputs a set in $\arg\max_{S\subseteq M}\{v(S) - \mathbf{p}(S)\}$. Recall also that implementing a demand oracle is NP-hard when $v(\cdot)$ is submodular. In fact, it is NP-hard to even guarantee better than a m-approximation when $v(\cdot)$ is submodular (more precisely, for any $\varepsilon > 0$ it is NP-hard to guarantee a set T satisfying $v(T) - \mathbf{p}(T) \geq \frac{1}{O(m^{1-\varepsilon})} \cdot \max_S\{v(S) - \mathbf{p}(S)\}$ [25]). Motivated by this, we pursue instead a bicriterion approximation. Specifically:

▶ **Definition 11.** *For any $c, d \leq 1$, a (c, d)-*approximate demand oracle *takes as input a valuation function $v(\cdot)$ and a price vector $\mathbf{p}$ and outputs a set of items S such that*

$$v(S) - \mathbf{p}(S) \geq c \cdot \max_T\{v(T) - \mathbf{p}(T)/d\}.$$

That is, a (c, d)-demand oracle outputs a set guaranteeing at least a c-fraction of the optimal utility *if all prices were blown up by a factor of* $1/d$. We refer to the utility of the optimal bundle with these higher prices (i.e. $\max_T\{v(T) - \mathbf{p}(T)/d\}$) as the *benchmark* (so our goal is to be c-competitive with the benchmark). In this section, we establish that poly-time $(1/2, 1/2)$-approximate demand oracles exist for submodular functions, based on the simple greedy algorithm of [33].

Algorithm 1 SimpleGreedy$(v, \mathbf{p}, M)$.

$S \leftarrow \emptyset$
for $j = 1, \ldots, m$ **do** ▷ For items in an arbitrary order
  if $v(S \cup \{j\}) - v(S) \geq 2\mathbf{p}(j)$ **then** ▷ If the marginal gain is at least twice the price
    $S \leftarrow S \cup \{j\}$ ▷ Then allocate that item
return S

▶ **Proposition 12.** *When $v(\cdot)$ is submodular, SimpleGreedy is a $(1/2, 1/2)$-approximate demand oracle.*

Proof. Our proof follows by induction on the number of items m. Importantly, observe that SimpleGreedy is recursive. Specifically, if we do not allocate item 1, then the remainder of the for loop is simply SimpleGreedy$(v, \mathbf{p}, M \setminus \{1\})$. If we do allocate item 1, then the remainder of the for loop is simply SimpleGreedy$(v_{\{1\}}, \mathbf{p}, M \setminus \{1\})$, where $v_S(T) := v(S \cup T) - v(S)$. Also importantly, observe that $v_S(\cdot)$ is submodular whenever $v(\cdot)$ is submodular (like [33], this is the only part of the proof which requires submodularity instead of subadditivity).

Now we begin with the base case. Observe that when $m = 1$, SimpleGreedy purchases the item if and only if the value exceeds twice the price. So when SimpleGreedy purchases the item, it is optimal. When SimpleGreedy doesn't purchase the item, the benchmark is 0 (because we compete with the optimal utility when the prices are doubled, which is zero). So in both cases, it guarantees a the required $(1/2, 1/2)$-approximation. This proves the base case.

Now assume that the proposition holds for a fixed $m \geq 1$, and consider the case with $m + 1$ items. First, observe that if SimpleGreedy does not allocate item 1, it is because $v(1) < 2\mathbf{p}(1)$. By submodularity of $v(\cdot)$ (in fact, subadditivity suffices), this implies that $v(S) - 2\mathbf{p}(S) > v(S \cup \{1\}) - 2\mathbf{p}(S \cup \{1\})$ for all $S \ni 1$ (and in particular, that the optimum when prices are doubled does not contain item 1). By the inductive hypothesis, SimpleGreedy finds a $(1/2, 1/2)$-approximation for $v(\cdot)$ on $M \setminus \{1\}$, which by the previous sentence is also a $(1/2, 1/2)$-approximation for $v(\cdot)$ on M, completing the inductive step in this case.

It remains to consider the case where SimpleGreedy allocates item 1. Let $S_2 := S \setminus \{1\}$ denote the set output by SimpleGreedy$(v_{\{1\}}, \mathbf{p}, M \setminus \{1\})$, and let $O^* := \arg\max\{v(Y) - 2\mathbf{p}(Y)\}$ be the optimum bundle if prices were doubled. Then the inductive hypothesis guarantees:

$$v_{\{1\}}(S_2) - \mathbf{p}(S_2) \geq v_{\{1\}}(O^*)/2 - \mathbf{p}(O^* \setminus \{1\}).$$

Suppose first $1 \in O^*$. The inductive hypothesis then implies:

$$\begin{aligned} v(O^*)/2 - \mathbf{p}(O^*) &= v_{\{1\}}(O^*)/2 - \mathbf{p}(O^* \setminus \{1\}) + v(\{1\})/2 - \mathbf{p}(\{1\}) \\ &\leq v_{\{1\}}(S_2) - \mathbf{p}(S_2) + v(\{1\})/2 - \mathbf{p}(\{1\}) \\ &\leq v(S) - \mathbf{p}(S). \end{aligned}$$

Above, the first and third lines are simply expanding the definition of $v_{\{1\}}(\cdot)$, and the second line follows by inductive hypothesis. Observe that this concludes a $(1/2, 1/2)$-approximation in the case that $1 \in O^*$. Now, suppose instead that $1 \notin O^*$. Then we have:

$$\begin{aligned} v(O^*)/2 - \mathbf{p}(O^*) &\leq v(O^* \cup \{1\})/2 - \mathbf{p}(O^*) = v_{\{1\}}(O^*)/2 + v(\{1\})/2 - \mathbf{p}(O^*) \\ &\leq v_{\{1\}}(S_2) - \mathbf{p}(S_2) + v(\{1\})/2 \\ &\leq v_{\{1\}}(S_2) - \mathbf{p}(S_2) + v(\{1\}) - \mathbf{p}(1) \\ &= v(S) - \mathbf{p}(S). \end{aligned}$$

Above, the first line follows by monotonicity and expanding the definition of $v_{\{1\}}(\cdot)$. The second line follows by inductive hypothesis. The third line follows as $v(\{1\}) \geq 2\mathbf{p}(1)$ by assumption that SimpleGreedy allocates item 1. The final line follows again by expanding $v_{\{1\}}(\cdot)$. This concludes both cases of the inductive step, and the proof of the proposition. ◀

This concludes our development of bicriterion approximate demand oracles. The following section establishes that a wide class of posted-price mechanisms that achieve good guarantees when buyers use precise demand queries maintain their guarantees when buyers follow advice given by bicriterion approximate demand oracles.

4 Welfare Guarantees with Approximate Demand Oracles

In this section, we demonstrate that a slight modification of the $O((\log\log m)^3)$ approximation of [1] (which is truthful when buyers implement precise demand oracles) maintains its approximation guarantee when buyers follow advice recommended by a $(1/2, 1/2)$-approximate demand oracle. We begin with the main insight below, followed by a precise statement of our main result.

4.1 Fixed Price Auctions with Approximate Demand Oracles

A key component of the [1] (and related) auctions is the notion of a *Fixed Price Auction*. A fixed price auction simply sets a price $\mathbf{p}(j)$ on item j, visits the buyers one at a time, and offers the buyer the option to purchase any set S of remaining items for price $\mathbf{p}(S)$ (so it is a posted-price mechanism which sets the same prices for all bidders).

A key lemma used by these works establishes that there *exists* a fixed price auction generating good welfare (when bidders implement exact demand oracles) for any instance with submodular bidders (or even XOS bidders).[5] One can view the [14, 1] auctions as attempting to *learn* such a "good" fixed price auction. The key intuition behind our extension is that good fixed price auctions still exist when bidders only implement approximate demand oracles. This is captured formally by Lemma 15 below, which first requires the notion of *supporting prices*.

▶ **Definition 13.** $\mathbf{q}$ *are* supporting prices *for* $v_1(\cdot), \ldots, v_n(\cdot)$ *and allocation* $S_1, \ldots, S_n$ *if:*

- *For all* i, T, $v_i(T) \geq \mathbf{q}(S_i \cap T)$.
- *For all* i, $v_i(S_i) = \mathbf{q}(S_i)$.

▶ **Fact 14.** *When all* $v_i(\cdot)$ *are XOS, supporting prices exist for any allocation* $S_1, \ldots, S_n$.[6]

Much prior work leverages the fact that with precise demand queries, the fixed-price auction with prices $\mathbf{q}/2$ achieves half the optimal welfare. The intuition for our main result is that this key lemma extends to (c, d)-approximate demand queries by losing an additional $\min\{c, d\}$ factor. In the statement below, we will slightly abuse notation and say that a bidder "follows advice given by a (c, d)-approximate demand oracle" if they follow advice given by an algorithm which on input $v(\cdot), S$ computs a (c, d)-approximate demand query T, then advises $\arg\max\{v(S) - \mathbf{p}(S), v(T) - \mathbf{p}(T)\}$.

[5] This lemma appears at least as early as [16].

[6] Recall that submodular functions are XOS, and a function is XOS if it can be written as the maximum of additive functions. Supporting prices for items in S_i are defined by simply taking the additive function which defines $v_i(S_i)$.

▶ **Lemma 15.** *Let* $\mathbf{q}$ *be supporting prices for* $v_1(\cdot), \ldots, v_n(\cdot)$ *and* $S_1, \ldots, S_n$*. Then the fixed-price auction with prices* $d\mathbf{q}/2$ *guarantees welfare at least* $\min\{c, d\} \cdot \sum_i v_i(S_i)/2$ *when all bidders follow advice given by a* (c, d)*-approximate demand oracle.*

Proof. Let $S := \cup_i S_i$. Let also T_i denote the set purchased by bidder i (following advice given by a (c, d)-approximate demand oracle), and denote by $\textsc{Sold} = \bigcup_{i \in N} T_i$. Define $A_i = S_i \setminus \textsc{Sold}$. Because items in A_i are never allocated when bidder i is chosen to act (meaning that bidder i could choose to purchase the set A_i), and bidder i will choose a set guaranteeing at least as much utility as a (c, d)-approximate demand oracle, we have:

$$v_i(T_i) - d\mathbf{q}(T_i)/2 \geq c\left(v_i(A_i) - \frac{d\mathbf{q}(A_i)/2}{d}\right) = c\left(v_i(A_i) - \frac{1}{2}\mathbf{q}(A_i)\right).$$

The welfare achieved ($\sum_{i \in N} v_i(T_i)$) is exactly the sum of the utilities of each bidder ($v_i(T_i) - d\mathbf{q}(T_i)/2$) and the total revenue of the auction ($\sum_{i \in N} d\mathbf{q}(T_i)/2 = d\mathbf{q}(\textsc{Sold})/2$). By the definition of supporting prices (and the fact that $A_i \subseteq S_i$), we know that $v_i(A_i) \geq \mathbf{q}(A_i)$. Thus:

$$\begin{aligned}
\frac{d}{2}\mathbf{q}(\textsc{Sold}) + \sum_{i=1}^{n} v_i(T_i) - \frac{d}{2}\mathbf{q}(T_i) &\geq \frac{d}{2}\mathbf{q}(\textsc{Sold}) + c\left(\sum_{i=1}^{n} v_i(A_i) - \frac{1}{2}\mathbf{q}(A_i)\right) \\
&\geq \frac{d}{2}\mathbf{q}(\textsc{Sold}) + \frac{c}{2}\sum_{i=1}^{n}\mathbf{q}(A_i) \\
&\geq \frac{\min\{c, d\}}{2}\big(\mathbf{q}(\textsc{Sold}) + \mathbf{q}(S \setminus \textsc{Sold})\big) \\
&\geq \frac{\min\{c, d\}}{2}\mathbf{q}(S) = \frac{\min\{c, d\}}{2}\sum_i v_i(S_i).
\end{aligned}$$

The first inequality follows as each bidder follows advice of a (c, d)-approximate demand oracle. The second follows as $v_i(A_i) \geq \mathbf{q}(A_i)$ for all i (by definition of supporting prices). The third follows as $\cup_i A_i = S \setminus \textsc{Sold}$. The final inequality follows by basic arithmetic, and the final equality follows as $\mathbf{q}(S) = \sum_i v_i(S_i)$ by definition of supporting prices. ◀

Lemma 15 captures the main intuition for why existing posted-price guarantees can be extended to accommodate bicriterion approximate demand queries. Of course, the [1] mechanism is not just a single posted-price mechanism, and Lemma 15 is just one technical lemma used along the way (to be more precise, a generalization of Lemma 15 is used along the way, but the overly technical statement hides the intuition). But an outline similar to the proof of Lemma 15 establishes the more general claim. Section 4.2 formally states our main result, and all details of the proof aside from the above intuition can be found in Appendix B.

4.2 Formal Statement of Main Result

▶ **Theorem 16.** *Let* $\mathcal{V}$ *be a subclass of XOS valuations and let* D *be a poly-time* (c, d)*-approximate demand oracle for valuation class* $\mathcal{V}$*. Then there exists a poly-time mechanism for welfare maximization when all valuations are in* $\mathcal{V}$ *with approximation guarantee* $O\left(\max\left\{\frac{1}{c}, \frac{1}{d}\right\} \cdot (\log\log m)^3\right)$ *in implementation in advised strategies with polynomial time computable advice.*

Theorem 1 now follows from Theorem 16 as submodular valuations are a subclass of XOS which admits poly-time $(1/2, 1/2)$-approximate demand oracles. The poly-time mechanism witnessing Theorem 16 is a slight modification of [1]. The high-level approach of their

mechanism is the following: because $\mathcal{V}$ is XOS, Lemma 15 establishes that there *exists* a fixed-price mechanism which achieves an $1/O(\min\{c,d\}) = O(\max\{1/c, 1/d\})$ approximation in implementation in advised strategies. Of course, implementing this fixed-price auction requires complete knowledge of $v_1(\cdot), \ldots, v_n(\cdot)$, which the seller lacks. The mechanism of [1] essentially tries to iteratively guess a better and better set of fixed prices, and then pick one uniformly at random.

Intuitively, our adapted [1] mechanism works with (c,d)-approximated demand oracles for the same reason that Lemma 15 works with approximate demand oracles. Formally establishing this requires a bit of work, but much of the analysis of [1] treats the $(c,d) = (1,1)$ case of Lemma 15 as a black box, and therefore we can leverage most of their analysis as a black box as well. The generalized Lemma 15 (Lemma 20) provides all the properties of demand queries which their proof requires (and all the properties of approximate demand queries which our adaptation requires). A complete proof appears in Appendix B.

5 Approximate Demand Queries beyond Submodular

In this section, we explore approximate demand queries beyond submodular valuation functions. As the approximation guarantees of [1] hold for XOS valuations with precise demand queries, a poly-time $(\Omega(1), \Omega(1))$-approximate demand query would immediately extend their guarantees to XOS valuations under implementation in advised strategies. Interestingly, this very fact establishes that for all $\varepsilon > 0$, no poly-time $(\Omega(m^{-1/2+\varepsilon}), \Omega(m^{-1/2+\varepsilon}))$-approximate demand oracle exists for XOS valuations using subexponentially-many value queries.

▶ **Proposition 17.** *For all $\varepsilon > 0$, there is no $(\Omega(m^{-1/2+\varepsilon}, \Omega(m^{-1/2+\varepsilon}))$-approximate demand oracle for XOS valuations using $\mathrm{poly}(m)$ value queries.*

Proof. Assume for contradiction that the proposition were false. Then by Theorem 16, there exists an algorithm using $\mathrm{poly}(n,m)$ value queries that approximates the optimal welfare within $O\left(m^{1/2-\varepsilon} \cdot (\log\log m)^3\right) \in O(m^{1/2-\varepsilon/2})$ for XOS valuations. However, Theorem 6.1 of [17] proves that no such algorithm exists. ◀

To complete the picture, we also design poly-time $(\Omega(1/\sqrt{m}), \Omega(1/\sqrt{m}))$-approximate demand oracles for subadditive valuations (defined immediately below, based on the $\Omega(1/\sqrt{m})$-approximation of [17]), which is the best possible using subexponentially-many value queries.

Algorithm 2 SingleOrBundle$(v, \mathbf{p}, M)$.

$j \leftarrow \arg\max_{j \in M} v(j) - \mathbf{p}(j)$
$M^* \leftarrow \{j \in M : v(j) - (1+\sqrt{m})\mathbf{p}(j) > 0\}$
if $v(M^*) - \mathbf{p}(M^*) > v(j) - \mathbf{p}(j)$ **then**
  return M^*
else
  return $\{j\}$

▶ **Proposition 18.** SingleOrBundle$(v, \mathbf{p}, M)$ *is a $(\frac{1}{\sqrt{m}}, \frac{1}{1+\sqrt{m}})$-approximate demand oracle for subadditive valuation functions.*

Proof. Let S be the set that maximizes utility $(v(S) - (1+\sqrt{m}) \cdot \mathbf{p}(S))$. If $S = \emptyset$, then the benchmark is 0, and SingleOrBundle achieves non-negative utility. It remains to consider the case $v(S) - \mathbf{p}(S) > 0$.

In this case, let T be the set returned by SingleOrBundle$(v, \mathbf{p}, M)$. Call an item j *special* if:

$$v(j) - \mathbf{p}(j) \geq \frac{1}{\sqrt{m}}(v(S) - (1+\sqrt{m}) \cdot \mathbf{p}(S)).$$

Observe that if any item j is special, then we conclude:

$$v(T) - \mathbf{p}(T) \geq v(j) - \mathbf{p}(j) \geq \frac{1}{\sqrt{m}}(v(S) - (1+\sqrt{m}) \cdot \mathbf{p}(S)),$$

which is a $(\frac{1}{\sqrt{m}}, \frac{1}{1+\sqrt{m}})$-approximate demand oracle. If no item is special, then $\forall j \in M^*$, $v(j) - \mathbf{p}(j) < \frac{1}{\sqrt{m}}(v(S) - (1+\sqrt{m})\mathbf{p}(S))$. Summing this for all $j \in M^*$ yields:

$$\left(\sum_{j \in M^*} v(j)\right) - \mathbf{p}(M^*) < \frac{|M^*|}{\sqrt{m}}(v(S) - (1+\sqrt{m}) \cdot \mathbf{p}(S)) \leq \sqrt{m} \cdot v(S) \leq \sqrt{m} \cdot v(M^*). \tag{1}$$

The final inequality follows as S cannot contain items for which $v(j) < (1+\sqrt{m})p(j)$, as $v(\cdot)$ is subadditive. We can then conclude that:

$$\begin{aligned} v(M^*) - \mathbf{p}(M^*) &> \frac{1}{\sqrt{m}}\left(\left(\sum_{j \in M^*} v(j)\right) - \mathbf{p}(M^*)\right) - \mathbf{p}(M^*) \\ &= \frac{1}{\sqrt{m}}\left(\sum_{j \in M^*} v(j) - (1+\sqrt{m})\mathbf{p}(j)\right) \\ &\geq \frac{1}{\sqrt{m}}\left(\sum_{j \in S} v(j) - (1+\sqrt{m})\mathbf{p}(j)\right) \\ &\geq \frac{1}{\sqrt{m}}\left(v(S) - (1+\sqrt{m})\mathbf{p}(S)\right). \end{aligned}$$

The first inequality follows directly from (1). The second follows as $v(j) > (1+\sqrt{m})\mathbf{p}(j)$ for all $j \in M^*$, and $S \subseteq M^*$ (because $v(\cdot)$ is subadditive, and S is the utility-maximizing set at prices $(1+\sqrt{m})\mathbf{p}$). The third follows from subadditivity of $v(\cdot)$. We conclude that when there are no special items, the proposition is satisfied as well, completing the proof. ◀

References

1. Sepehr Assadi and Sahil Singla. Exponentially Improved Truthful Combinatorial Auctions with Submodular Bidders. In *Proceedings of the Sixtieth Annual IEEE Foundations of Computer Science (FOCS)*, 2019.
2. Moshe Babaioff, Ron Lavi, and Elan Pavlov. Single-value combinatorial auctions and algorithmic implementation in undominated strategies. *J. ACM*, 56(1):4:1–4:32, 2009. doi:10.1145/1462153.1462157.
3. Mark Braverman, Jieming Mao, and S. Matthew Weinberg. On Simultaneous Two-player Combinatorial Auctions. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 2256–2273, 2018. doi:10.1137/1.9781611975031.146.
4. Patrick Briest, Piotr Krysta, and Berthold Vöcking. Approximation techniques for utilitarian mechanism design. In *the 37th Annual ACM Symposium on Theory of Computing (STOC)*, 2005.

5 Niv Buchbinder, Moran Feldman, Joseph Naor, and Roy Schwartz. A Tight Linear Time (1/2)-Approximation for Unconstrained Submodular Maximization. *SIAM J. Comput.*, 44(5):1384–1402, 2015. doi:10.1137/130929205.

6 Dave Buchfuhrer. A Theory of Robust Hardness for Truthful Mechanism Design. Manuscript, 2011. URL: http://users.cms.caltech.edu/~dave/papers/oracles.pdf.

7 David Buchfuhrer, Shaddin Dughmi, Hu Fu, Robert Kleinberg, Elchanan Mossel, Christos H. Papadimitriou, Michael Schapira, Yaron Singer, and Christopher Umans. Inapproximability for VCG-Based Combinatorial Auctions. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2010.

8 David Buchfuhrer, Michael Schapira, and Yaron Singer. Computation and incentives in combinatorial public projects. In *Proceedings 11th ACM Conference on Electronic Commerce (EC-2010), Cambridge, Massachusetts, USA, June 7-11, 2010*, pages 33–42, 2010. doi:10.1145/1807342.1807348.

9 Edward H. Clarke. Multipart Pricing of Public Goods. *Public Choice*, 11(1):17–33, 1971.

10 Amit Daniely, Michael Schapira, and Gal Shahaf. Inapproximability of Truthful Mechanisms via Generalizations of the VC Dimension. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 401–408, 2015. doi:10.1145/2746539.2746597.

11 Nikhil R. Devanur, Jamie Morgenstern, Vasilis Syrgkanis, and S. Matthew Weinberg. Simple Auctions with Simple Strategies. In *Proceedings of the Sixteenth ACM Conference on Economics and Computation, EC '15, Portland, OR, USA, June 15-19, 2015*, pages 305–322, 2015. doi:10.1145/2764468.2764484.

12 Shahar Dobzinski. Two Randomized Mechanisms for Combinatorial Auctions. In *Proceedings of the 10th International Workshop on Approximation and the 11th International Workshop on Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 89–103, 2007.

13 Shahar Dobzinski. An Impossibility Result for Truthful Combinatorial Auctions with Submodular Valuations. In *Proceedings of the 43rd ACM Symposium on Theory of Computing (STOC)*, 2011.

14 Shahar Dobzinski. Breaking the Logarithmic Barrier for Truthful Combinatorial Auctions with Submodular Bidders. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2016, pages 940–948, New York, NY, USA, 2016. ACM. doi:10.1145/2897518.2897569.

15 Shahar Dobzinski. Computational Efficiency Requires Simple Taxation. In *FOCS*, 2016.

16 Shahar Dobzinski, Noam Nisan, and Michael Schapira. Truthful randomized mechanisms for combinatorial auctions. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 644–652. ACM, 2006.

17 Shahar Dobzinski, Noam Nisan, and Michael Schapira. Approximation Algorithms for Combinatorial Auctions with Complement-Free Bidders. *Math. Oper. Res.*, 35(1):1–13, 2010. doi:10.1287/moor.1090.0436.

18 Shahar Dobzinski, Noam Nisan, and Michael Schapira. Truthful randomized mechanisms for combinatorial auctions. *J. Comput. Syst. Sci.*, 78(1):15–25, 2012. doi:10.1016/j.jcss.2011.02.010.

19 Shahar Dobzinski and Jan Vondrák. From query complexity to computational complexity. In *Proceedings of the 44th Symposium on Theory of Computing (STOC)*, 2012.

20 Shahar Dobzinski and Jan Vondrak. The Computational Complexity of Truthfulness in Combinatorial Auctions. In *Proceedings of the ACM Conference on Electronic Commerce (EC)*, 2012.

21 Shahar Dobzinski and Jan Vondrák. Impossibility Results for Truthful Combinatorial Auctions with Submodular Valuations. *J. ACM*, 63(1):5:1–5:19, 2016. doi:10.1145/2786754.

22 Paul Duetting, Michal Feldman, Thomas Kesselheim, and Brendan Lucier. Prophet Inequalities Made Easy: Stochastic Optimization by Pricing Non-Stochastic Inputs. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 540–551, 2017. doi:10.1109/FOCS.2017.56.

23 Tomer Ezra, Michal Feldman, Eric Neyman, Inbal Talgam-Cohen, and S. Matthew Weinberg. Settling the Communication Complexity of Combinatorial Auctions with Two Subadditive Buyers. In *the 60th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2019.

24 Uriel Feige. On Maximizing Welfare When Utility Functions Are Subadditive. *SIAM J. Comput.*, 39(1):122–142, 2009. doi:10.1137/070680977.

25 Uriel Feige and Shlomo Jozeph. Demand Queries with Preprocessing. In *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part I*, pages 477–488, 2014. doi:10.1007/978-3-662-43948-7_40.

26 Michal Feldman, Nick Gravin, and Brendan Lucier. Combinatorial Auctions via Posted Prices. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '15, pages 123–135, Philadelphia, PA, USA, 2015. Society for Industrial and Applied Mathematics. URL: http://dl.acm.org/citation.cfm?id=2722129.2722139.

27 Moran Feldman. Guess Free Maximization of Submodular and Linear Sums. In *Algorithms and Data Structures - 16th International Symposium, WADS 2019, Edmonton, AB, Canada, August 5-7, 2019, Proceedings*, pages 380–394, 2019. doi:10.1007/978-3-030-24766-9_28.

28 Theodore Groves. Incentives in Teams. *Econometrica*, 41(4):617–631, 1973.

29 Chris Harshaw, Moran Feldman, Justin Ward, and Amin Karbasi. Submodular Maximization beyond Non-negativity: Guarantees, Fast Algorithms, and Applications. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, pages 2634–2643, 2019. URL: http://proceedings.mlr.press/v97/harshaw19a.html.

30 Stavros G. Kolliopoulos and Clifford Stein. *Approximating Disjoint-Path Problems Using Greedy Algorithms and Packing Integer Programs*, pages 153–168. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998. doi:10.1007/3-540-69346-7_12.

31 Piotr Krysta and Berthold Vöcking. Online Mechanism Design (Randomized Rounding on the Fly). In *Automata, Languages, and Programming - 39th International Colloquium, ICALP 2012, Warwick, UK, July 9-13, 2012, Proceedings, Part II*, pages 636–647, 2012. doi:10.1007/978-3-642-31585-5_56.

32 Ron Lavi and Chaitanya Swamy. Truthful and Near-Optimal Mechanism Design via Linear Programming. In *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2005.

33 Benny Lehmann, Daniel Lehmann, and Noam Nisan. Combinatorial Auctions with Decreasing Marginal Utilities. In *the 3rd Annual ACM Conference on Electronic Commerce (EC)*, 2001.

34 Vahab S. Mirrokni, Michael Schapira, and Jan Vondrák. Tight information-theoretic lower bounds for welfare maximization in combinatorial auctions. In *Proceedings 9th ACM Conference on Electronic Commerce (EC-2008), Chicago, IL, USA, June 8-12, 2008*, pages 70–77, 2008. doi:10.1145/1386790.1386805.

35 George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. An analysis of approximations for maximizing submodular set functions. *Mathematical Programming*, 14(1):265–294, 1978.

36 Christos H. Papadimitriou, Michael Schapira, and Yaron Singer. On the Hardness of Being Truthful. In *Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2008.

37 Marek Pycia and Peter Troyan. Obvious Dominance and Random Priority. In *Proceedings of the 2019 ACM Conference on Economics and Computation, EC 2019, Phoenix, AZ, USA, June 24-28, 2019.*, page 1, 2019. doi:10.1145/3328526.3329613.

38 Prabhakar Raghavan. Probabilistic Construction of Deterministic Algorithms: Approximating Packing Integer Programs. *J. Comput. Syst. Sci.*, 37(2):130–143, October 1988. doi:10.1016/0022-0000(88)90003-7.

39 Michael Schapira and Yaron Singer. Inapproximability of Combinatorial Public Projects. In *Internet and Network Economics, 4th International Workshop, WINE 2008, Shanghai, China, December 17-20, 2008. Proceedings*, pages 351–361, 2008. doi:10.1007/978-3-540-92185-1_41.

40 Maxim Sviridenko, Jan Vondrák, and Justin Ward. Optimal approximation for submodular and supermodular optimization with bounded curvature. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 1134–1148, 2015. doi:10.1137/1.9781611973730.76.

41 William Vickrey. Counterspeculations, Auctions, and Competitive Sealed Tenders. *Journal of Finance*, 16(1):8–37, 1961.

42 Jan Vondrák. Optimal approximation for the submodular welfare problem in the value oracle model. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 67–74, 2008. doi:10.1145/1374376.1374389.

A Brief Discussion of Definitions

The following example will motivate our decision to think of advice as *improving a given strategy* as opposed to *outright proposing a replacement strategy*.

Consider, for example, a single-bidder mechanism where the bidder faces one of k posted-price vectors $\mathbf{p}_1, \dots, \mathbf{p}_k$ chosen uniformly at random, and is asked to submit their desired sets $S_1, \dots, S_k$ before knowing which price is "real." Then the strategy which submits $S_i := \arg\max\{v(S) - \mathbf{p}_i(S)\}$ is dominant. In this case, advice could indeed simply propose this strategy to replace whatever else the bidder might try.

Things get more interesting, however, if the designer cannot recommend a dominant strategy. Consider instead a recommended strategy $T_1, \dots, T_k$ where $T_i \notin \arg\max\{v(S) - \mathbf{p}_i(S)\} \cup \arg\min\{v(S) - \mathbf{p}_i(S)\}$ for any i (call this strategy $\vec{T}$). If the designer shares the sets $T_1, \dots, T_k$ with the buyer, a reasonable buyer should certainly submit sets S_i satisfying $v(S_i) - \mathbf{p}_i(S_i) \geq v(T_i) - \mathbf{p}_i(T_i)$ for all i (because they could just swap any set violating this for T_i and strictly improve their utility). Consider then the strategy which sets $S_j \in \arg\max\{v(S) - \mathbf{p}_j(S)\}$ (for a single $j \in [k]$) and $S_i \in \arg\min\{v(S) - \mathbf{p}_i(S)\}$ for all $i \neq j$ (picks the optimal set for $\mathbf{p}_j$, and worst possible sets for all other $\mathbf{p}_i$, call this strategy $\vec{S}$). We don't want to say that a bidder originally planning to use $\vec{S}$ should instead use $\vec{T}$ (indeed, $\vec{T}$ does not dominate $\vec{S}$, and it's not a priori clear which strategy yields higher expected utility). But we do want to say that a bidder originally planning to use $\vec{S}$ should stick with S_j, and update S_i to T_i for all $i \neq j$. But in order to recommend such a strategy without knowing j in advance, $\vec{T}$ would need to be the dominant strategy itself. So in order for the solution concept to meaningfully apply to posted-price mechanisms without advising the dominant strategy itself, advice should really take the form of improving a tentative strategy rather than outright recommending a replacement. Lemma 29 below provides a representative example of how this solution concept can be harnessed for existing state-of-the-art mechanisms.

B Proof of Theorem 16

The full definition of "implementation in advised strategies" is very powerful, but a bit awkward to carry around. Throughout this appendix, we use the following definition of (c, d)-competitive sets, which simply says that a set of items will give the bidder utility at least as high as a (c, d)-approximate demand oracle.

▶ **Definition 19.** *A set S is a* (c, d)-competitive subset of M for v_i with prices $\mathbf{p}$ *if*

$$v_i(S) - \mathbf{p}(S) \geq c \cdot \max_{T \subseteq M} \{v_i(T) - \mathbf{p}(T)/d\}.$$

We say a bidder i picks (c, d) competitive sets *in a fixed price auction if, when the fixed price auction visits i, they pick a set which is a (c, d)-competitive subset of the collection of remaining items.*

The full proof of theorem 16 is fairly involved. We start off this section by providing the more technical version of lemma 15 in section B.1, which captures most of the properties of fixed price auctions with approximate demand queries which we need. Next in section B.2, we describe the "core algorithm" PriceLearningMechanism of [1]. Then in section B.3, using fairly elementary properties of fixed-price auctions, we prove the correctness of the PriceLearningMechanism, as long as 1) bidders pick (c, d)-competitive sets in every fixed price auction they participate in, and 2) we make the simplifying assumption 21. In section B.4, we remove the simplifying assumption, and prove that there exists poly time computable advice such that, when bidders are following the advice, they always pick (c, d)-competitive sets.

B.1 Generalization of Lemma 15

In this subsection we state and prove the generalization of Lemma 15, which will be used in the analysis of the PriceLearningMechanism.

The first term in the maximum below (and the "moreover" part of the lemma) relates the achieved welfare with the value of the unsold items, and will be used to handle "learning" phases in the mechanism. The second term of the maximum shows that once we have learned the prices well, we definitely get good welfare.

▶ **Lemma 20.** *Suppose* $\{T_i\}_{i \in N} \leftarrow \text{FixedPriceAuction}(M, N, d\mathbf{p})$*, where each bidder i picks a subset of the remaining items which is (c, d)-competitive set for v_i with prices $\mathbf{p}$. Let $\{O_i\}_{i \in N}$ be any allocation with supporting prices $\mathbf{q}$. Let S_i be the set of items j where $\delta\mathbf{q}(j) \leq \mathbf{p}(j) \leq \frac{1}{2}\mathbf{q}(j)$ and $j \in O_i$. Denote $S = \bigcup_{i \in N} S_i$ and $\textsc{Sold} = \bigcup_i T_i$. Then*

$$\sum_i v_i(T_i) \geq \max \begin{cases} \frac{c}{2} \cdot \mathbf{q}(S \setminus \textsc{Sold}), \\ \min\left(\frac{c}{2}, \quad \delta d\right) \cdot \mathbf{q}(S). \end{cases}$$

Moreover, suppose k is the last bidder in N. We also have $v_i(T_k) \geq \frac{c}{2}\mathbf{q}(S_k \setminus \textsc{Sold}_{<k})$, where $\textsc{Sold}_{<k} = \bigcup_{i<k} T_i$.

Proof. Let $A_i = S_i \setminus \textsc{Sold}$. Because items in A_i are never allocated when bidder i is chosen to act (and because each bidder picks a (c, d)-competitive set with prices $\mathbf{p}$), the utility of each bidder i satisfies

$$v_i(T_i) - d \cdot \mathbf{p}(T_i) \geq c \cdot (v_i(A_i) - \mathbf{p}(A_i)).$$

As $A_i \subseteq S_i$, we know $\delta \mathbf{q}(A_i) \leq \mathbf{p}(A_i) \leq \frac{1}{2}\mathbf{q}(A_i)$, and by the definition of supporting prices, we know that $\mathbf{q}(A_i) \leq v_i(A_i)$ Thus, $\{T_i\}_{i \in N}$ achieves welfare

$$\begin{aligned}\sum_{i \in N} v_i(T_i) &= \sum_{i \in N} \big(v_i(T_i) - d \cdot \mathbf{p}(T_i)\big) + d \sum_{i \in N} \mathbf{p}(T_i) \\ &\geq c \sum_{i \in N} \big(v_i(A_i) - \mathbf{p}(A_i)\big) + d \cdot \mathbf{p}(\textsc{Sold}) \\ &\geq \sum_i c(\mathbf{q}(A_i) - \frac{1}{2}\mathbf{q}(A_i)) + d\delta \cdot \mathbf{q}(\textsc{Sold}). \qquad (*)\end{aligned}$$

Observe that $S \setminus \textsc{Sold} = \bigcup_{i \in N} A_i$. Thus, ignoring the term $d\delta\mathbf{q}(\textsc{Sold})$ from (*), we can conclude the auction gets welfare at least $\frac{c}{2}\mathbf{q}(S \setminus \textsc{Sold})$. Moreover, (*) tells us we get welfare at least

$$\min\left(\frac{c}{2}, \delta d\right) \cdot \big(\mathbf{q}(S \setminus \textsc{Sold}) + \mathbf{q}(\textsc{Sold})\big) \geq \min\left(\frac{c}{2}, \delta d\right) \cdot \mathbf{q}(S),$$

from which we can conclude main statement of the lemma.

For the "moreover" component, simply observe that when bidder k was picked by the mechanism, the items in $A' := S_k \setminus \textsc{Sold}_{<k}$ were still available, and that $S_k \subseteq O_k$, so

$$v_i(T_k) \geq v_i(T_k) - d\mathbf{p}(T_k) \geq c(v_i(A') - \mathbf{p}(A')) \geq c(\mathbf{q}(A') - \mathbf{q}(A')/2) = \frac{c}{2}\mathbf{q}(A'). \qquad \blacktriangleleft$$

B.2 The Mechanism

For the reader's convenience, we first briefly describe the mechanism in [1] and quote the mechanism verbatim (the only change we need to make is that every price used by the mechanism is "discounted" by an extra factor of d, plus some slight simplifications in the "removing extra assumptions" step). Then we present a slightly condensed version of the analysis.

High-level overview. Posted price mechanisms for combinatorial auctions typically use the following high-level strategy: attempt to (approximately) *learn* the supporting prices (definition 13) of an optimal allocation, then sell the items at those prices. The key innovation of [1] is to "explore" prices for each item individually using a *price tree* in which each successive layer of the tree corresponds to a finer "granularity" of prices. Initially, each item is set at a price corresponding to the root of the tree, and in each successive round of the mechanism, the price of each item moves one layer down in the tree to a "more precise" price which corresponds to some child node of the old price.

The mechanism of [1] runs several fixed-price auctions for each round (i.e. each layer of the price tree). In each of these successive auctions, each item is priced higher and higher in a way corresponding to the children of the "old" price node of the item. The price in the next round of the mechanism is then the highest price in the next layer where the item was still sold. The idea here is that, in the next layer of prices, we need to make the prices as high as possible such that the items will still sell. Intuitively, this serves to refine our estimate for the supporting prices as we move a layer down in the tree.

In fact, the story is more subtle than this. The mechanism may not actually achieve a better approximation to the prices in each layer, but [1] prove that if you *do not* get a better estimate for the prices, then you can *already* get a good approximation to the optimal welfare at the current prices. These two cases exactly correspond to the "learnable" or "allocatable" cases in lemma 22 below. For this reason, for every layer of the price tree, the mechanism

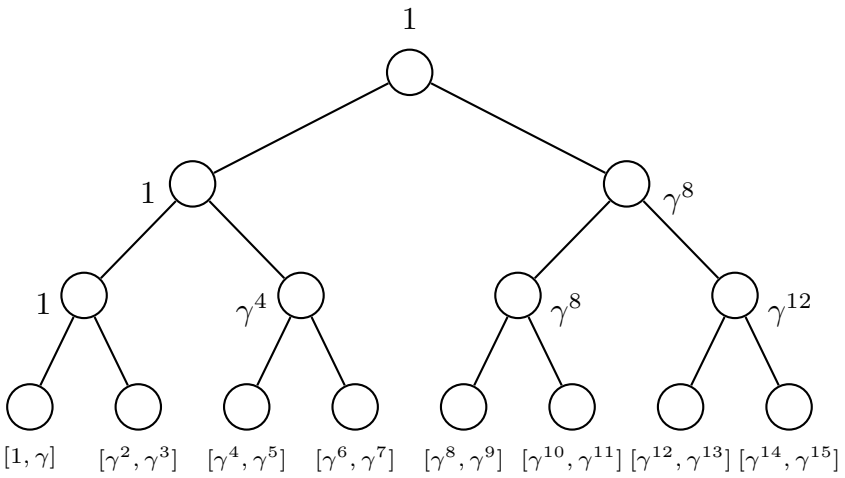


Figure 1 [1, Figure 2] An illustration of a price tree T^e with $\alpha = 2$, $\beta = 3$, and $\psi_{min} = 1$.

has some chance (proportional to the number of layers) of stopping early and allocating the items according to some fixed price auction in that layer. Thus, regardless of whether we always learn prices or if we hit the "allocatable" in some step, we will use a good auction with some probability.

Simplifying Assumption. It's useful for posted price mechanisms to know ahead of time the *range* of possible supporting prices of an optimal allocation. This assumption can be removed in a fairly "modular" way, as done in [1] (though we make some modifications in order to more easily fit our solution concept).

Let $\mathbf{q}$ be the supporting prices of an optimal allocation. Formally, our simplifying assumption is the following:

▶ **Assumption 21.** *There are known numbers ψ_{min}, ψ_{max} such that the supporting prices of any item in $\mathbf{q}$ are either 0 or in $[\psi_{min}, \psi_{max}]$, and ψ_{max}/ψ_{min} is polynomial in m.*

The price tree and mechanism parameters. We now formally describe the price tree in terms of three parameters:

- $\alpha = \Theta(1)$ is the branching factor of the tree (and the number of auctions in each iteration of the mechanism).
- $\beta = \Theta(\log\log(\psi_{max}/\psi_{min})) = \Theta(\log\log m)$ is the number of layers in the price tree (and the number of iterations of the mechanism).
- $\gamma = \Theta(\alpha\beta) = \Theta(\log\log m)$ is the "accuracy factor" of the prices.

We would like the leaves of the price tree correspond to "price buckets"

$$P = \{[\psi_{min}\gamma^i, \psi_{min}\gamma^{i+1}] \mid i = 0, 1, \ldots, k\}$$

for some k large enough that all prices in $[\psi_{min}, \psi_{max}]$ are considered. Informally, we take "learning a price $\mathbf{q}$ correctly" to mean that we find the bucket in P to which $\mathbf{q}$ belongs. We will assign "actual" prices in $\mathbf{p}$ according to the smallest price $\psi_{min}\gamma^i$ in the corresponding bucket. Our goal is that, if we "learn the price of $\mathbf{q}$ correctly", then the price used in $\mathbf{p}$ is within a γ factor of the true price in $\mathbf{q}$.

However, for technical reasons, we need a *gap* of at least γ between the prices of consecutive nodes in each layer of the tree (not just the leaves), so that prices will be guided to the closest leaf node *below* the price in $\mathbf{q}$ (this is desirable because lemma 20 requires prices in $\mathbf{p}$ to be less than in $\mathbf{q}$). To ensure this, the mechanism creates a price gap of factor γ between nodes by splitting P into

$$P^o = \{[\psi_{min}\gamma^{2k-1}, \psi_{min}\gamma^{2k}] \mid k = 1, \dots, \alpha^\beta\}$$
$$P^e = \{[\psi_{min}\gamma^{2k}, \psi_{min}\gamma^{2k+1}] \mid k = 0, \dots, \alpha^\beta - 1\}$$

Trees T^o and T^e are constructed with leaf nodes P^o and P^e respectively. We use T^* to denote either of T^o or T^e, and P^* will denote the corresponding P^o or P^e.

The price tree T^* is an α-branching tree with depth β (i.e. with $\beta + 1$ layers). The leaf nodes correspond, in left-to-right (depth-first search) order, to the price buckets in P^* (in increasing order). Furthermore, any non-leaf node x corresponding to a single price, which is the minimum value in any bucket of any leaf node which is a descendant of x. Thus, the prices corresponding to consecutive level-i nodes differ by a factor of $\gamma^{2\alpha^{\beta+1-i}}$.

Let $\mathbf{p}$ be a "level-i price vector", i.e. a vector in which the price of each item is a price which corresponds to some level-i node. We let $\text{next}_j^{(i)}(\mathbf{p}) = \mathbf{p}'$ denote the price vector constructed as follows: for each item $\ell \in M$, let x be the level-i node whose corresponding price is $\mathbf{p}(\ell)$. Then set $\mathbf{p}'(\ell)$ to the price corresponding to the jth child node of x. Thus, a precise formula is given by $\mathbf{p}'(\ell) = \gamma^{2\alpha^{\beta-i}(j-1)}\mathbf{p}(\ell)$. In words, $\text{next}_j^{(i)}(\mathbf{p})$ sets the price of each item ℓ to be the jth largest "refined price" below the current price of item ℓ.

The mechanism. We start by randomly picking a price tree T^o or T^e. The mechanism then proceeds in β iterations (though it may terminate early) in which a price vector $\mathbf{p}^{(i)}$ is constructed in each iteration i. Initially, $\mathbf{p}^{(1)}$ is the (unique) level-1 price vector of T^*. In each iteration, a $\frac{1}{10\beta}$ fraction of the bidders are selected uniformly at random, and the α different price vectors $\mathbf{p}_j^{(i)} = \text{next}_j^{(i)}(\mathbf{p}^{(i)})$ for $j = 1, \dots, \alpha$ are considered. The mechanism runs fixed price auction with the current set of bidders on prices $d\mathbf{p}_j^{(i)}/2$ for $j = 1, \dots, \alpha$. The new (level-$(i+1)$) vector $\mathbf{p}^{(i+1)}$ is then constructed as follows: for each item ℓ, $\mathbf{p}^{(i+1)}(\ell) = \mathbf{p}_j^{(i)}(\ell)$, where j is the highest index such that item ℓ sold in the auction with prices $d\mathbf{p}_j^{(i)}/2$. (or $\mathbf{p}^{(i)}(\ell)$ if no such j exists). In words, the new price of ℓ is the price of ℓ in the highest auction in iteration i for which item ℓ was sold. For each fixed price auction described in this paragraph, there is a $1/\Omega(\alpha\beta)$ chance that the mechanism will terminate early and return the allocation determined by the auction. This serves to strictly incentivizes bidder to pick good sets, but also serves an important purpose for achieving the desired approximation grantee, as discussed in the overview.

If the mechanism does not terminate early in iteration $1, \dots, \beta$, then the final step of the mechanism is to run a fixed price auction with all of the remaining bidders on prices $d\mathbf{p}^{(\beta)}/2$. (The hope is that, for a large fraction (weighted by $\mathbf{q}$) of the items, the price of the items is in the level-β bin which is closest to the price in $\mathbf{q}$, and thus we can apply lemma 20.)

The exact mechanism in [1] is quoted in Algorithm 3.

B.3 The Modified Analysis

Notation

We follow [1] and depart somewhat from conventional notation for the analysis of the mechanism. We let i denote an iteration of the mechanism, j denote an auction inside some iterations, b denote a bidder, and ℓ denote an item.

Algorithm 3 PriceLearningMechanism(N, M).

procedure PARTITION(N)
  Permute N uniformly at random.
  for $i = 1, 2, ..\beta$ **do**
    Remove $\frac{|N|}{10\beta}$ bidders uniformly at random from N; assign them to the set N_i.
  Put the remaining items in N into $N_{\beta+1}$.
procedure PRICEUPDATE$(A_1,\ldots,A_\alpha, \mathbf{p}_1,\ldots,\mathbf{p}_\alpha)$
  For each $\ell \in M$, let $\mathbf{p}'(\ell) = \mathbf{p}_j(\ell)$ for the
    highest value of j such that ℓ is allocated in A_j (or $\mathbf{p}_1(\ell)$ if no j exists)
  Return $\mathbf{p}'$

Let $(N_1, N_2, ...N_{\beta+1}) \leftarrow$ Partition(N)
Pick one of the modified trees T^o or T^e uniformly at random and denote it by T^*
Let $\mathbf{p}^{(1)}$ be the (unqiue) level-1 (root) price of T^*
for $i = 1, \ldots, \beta$ **do**
  For $j = 1, \ldots, \alpha$, let $\mathbf{p}_j^{(i)} = \text{next}_j^{(i)}(\mathbf{p}^{(i)})$
  For $j = 1, \ldots, \alpha$: run FixedPriceAuction(N_i, M, $d\mathbf{p}_j^{(i)}/2$) and let $A_j^{(i)}$ be the allocation
  With probability $(1/\beta)$, pick $j^* \in [\alpha]$ u.a.r. and return $A_{j^*}^{(i)}$ as the final allocation
  Otherwise, let $\mathbf{p}^{(i+1)} \leftarrow$PriceUpdate$(A_1^{(i)}, ..., A_\alpha^{(i)}, \mathbf{p}_1^{(i)}, ..., \mathbf{p}_\alpha^{(i)})$, and continue
Run FixedPriceAuction$(N_{\beta+1}, M, d\mathbf{p}^{(\beta+1)}/2)$ and return the allocation A^*

Let O be an optimal allocation with supporting prices $\mathbf{q}$ and OPT be the optimal welfare resulting from allocation O. Let $\mathbf{q}^*$ be $\mathbf{q}$ restricted to items whose prices are in some bucket of P^*. Let O^* be the collection of those items. Let $N_1, \ldots, N_{\beta+1}$ denote the groups of bidders from the Partition function. Given $(N_i)_i$ and T^* as picked by the mechanism, define price vectors $\mathbf{q}^{(i)}$ as $\mathbf{q}^*$, restricted to items which are allocated in O^* to bidders from $N_i, N_{i+1}, \ldots, N_{\beta+1}$ (intuitively, we restricted attention to items which could still go to the same bidder in A as in O, and give price 0 to items that can no longer be allocated to the right bidder in O). Call item ℓ correctly priced at iteration i if $\mathbf{q}^{(i)}(\ell)$ is in the bin corresponding to some leaf node which is a child of the node corresponding to $\mathbf{p}^{(i)}(\ell)$. Let $C^{(i)}$ denote all items priced correctly before iteration i begins. Note that $C^{(1)} = O^*$ and that an item can only be in $C^{(i)}$ if it is also in $C^{(j)}$ for $j = 1, \ldots, i-1$, so $C^{(1)} \supseteq C^{(2)} \supseteq \ldots \supseteq C^{(\beta+1)}$. We separate $C^{(i)}$ into $C_1^{(i)}, C_2^{(i)}, ...C_\alpha^{(i)}$, where $C_j^{(i)}$ is the subset of items in $C^{(i)}$ that are priced correctly in $\text{next}_j^{(i)}(\mathbf{p}^{(i)})$. For any set of bidders N' and items D, let $O^D_{N'}$ be the restriction of O^* to items in D and bidders in N'.

Assumptions

Throughout the claims in this section, we assume all bidders pick (c, d)-competitive sets in every fixed price auction they participate in, though we may not restate this assumption in every claim statement[7]. We also assume that the optimal allocation O has supporting prices $\mathbf{q}$.

[7] It is somewhat easier to prove that PriceLearningMechanism is implementable in advised strategies compared to GeneralizedMechanism below. However, we hold off and only demonstrate that Generalized-Mechanism is implementable in advised strategies, both for completeness, and in order to demonstrate that our solution concept "composes well" to be useful for complicated mechanisms.

The following lemma is the heart of the proof of the approximation ratio of mechanism 3.

▶ **Lemma 22** (Learnable-Or-Allocable Lemma from [1]). *Assume 21, and suppose all bidders pick (c, d)-competitive sets in every fixed price auction they participate in. For any iteration $i \in [\beta]$, conditioned on any outcome of first $i-1$ iterations and choice of T^*,*

1. *either* $\mathbb{E}\left[\mathbf{q}^{(i+1)}(C^{(i+1)})\right] \geq \mathbf{q}^{(i)}(C^{(i)}) - \frac{OPT}{3\beta}$*, where the expectation is over N_i;*
2. *or* $\mathbb{E}\left[\text{val}(A_{j^*})^{(i)}\right] \geq \frac{c}{O(\alpha\beta^2)} OPT$.

First we prove a series of claims before proving the Learnable-Or-Allocable lemma, following the same outline as [1]. For claims 23 and 24, we fix some $j \in [\alpha]$ and let $D = C_j^{(i)}$. Note that $\mathbf{q}^{(i)}(C_j^{(i)}) = \mathbf{q}^{(i)}(O^D_{N_{\geq i}})$, as $\mathbf{q}^{(i)}$ zeros out items allocated in O to bidders from $N_{<i}$.

▷ Claim 23. (5.3 from [1]) Deterministically, $\text{val}(A_j^{(i)}) \geq \frac{c}{2} \cdot \mathbf{q}^{(i)}(O^D_{N_i} \setminus A_j^{(i)})$.

Proof. Recall that $O^D_{N_i}$ is the restriction of O^* to items in $C_j^{(i)}$ and bidders in N_i. The definition of item ℓ being "priced correctly" means that $\mathbf{p}^{(i)}(\ell) \leq \mathbf{q}^{(i)}(\ell)$. Thus, for any $\ell \in O^D_{N_i}$ we get that $0 \cdot \mathbf{q}^{(i)}(\ell) \leq d\mathbf{p}^{(i)}(\ell)/2 \leq \mathbf{q}^{(i)}(\ell)/2$. Thus, the claim follows from lemma 20. ◁

▷ Claim 24. (5.4 from [1]) By randomness of choice of N_i from $N_{\geq i}$, $\mathbb{E}\left[\text{val}(A_j^{(i)})\right] \geq (\frac{c}{20\beta}) \cdot \mathbb{E}\left[\mathbf{q}^{(i)}(O^D_{N>i} \setminus A_j^{(i)})\right]$.

Proof. Consider picking a bidder $k \in N_{>i}$ uniformly at random and running an imaginary fixed price auction on $N_i \cup \{k\}$, where k is the last bidder chosen to act. Then by Lemma 20 (parameters in the lemma take values $N = N_i \cup \{k\}$, $\text{Sold}_{<k} = A_j^{(i)}$, $S_k = O_k^D$), the value bidder k gets from the imaginary fixed price auction satisfy $v_k(T_k) \geq \frac{c}{2} \cdot \mathbf{q}^{(i)}(O_k^D \setminus A_j^{(i)})$. We now take the expectation over the randomness on bidders $N_i \cup k$,

$$\begin{aligned}\mathop{\mathbb{E}}_{N_i, k\in N_{>i}}[v_k(T_k)] \geq \frac{c}{2} \cdot \mathop{\mathbb{E}}_{N_i, k\in N_{>i}}\left[\mathbf{q}^{(i)}(O_k^D \setminus A_j^{(i)})\right] &= \frac{c}{2} \cdot \mathop{\mathbb{E}}_{N_i}\left[\frac{1}{|N_{>i}|} \cdot \sum_{k\in N_{>i}} \mathbf{q}^{(i)}(O_k^D \setminus A_j^{(i)})\right] \\ &= \frac{1}{|N_{>i}|} \cdot \frac{c}{2} \cdot \mathbb{E}\left[\mathbf{q}^{(i)}(O^D_{N_{>i}} \setminus A_j^{(i)})\right].\end{aligned}$$

Observe that the expectation of $\text{val}(A_j^{(i)})$ is the same as the expected welfare of bidders in N_i in the imaginary fixed price auction. Since the bidders in N_i arrive before bidder k, their expected welfare in the imaginary fixed price auction is larger equal to that of bidder k. Thus by linearity of expectation

$$\mathbb{E}\left[\text{val}(A_j^{(i)})\right] \geq \frac{|N_i|}{|N_{>i}|} \cdot \frac{c}{2} \cdot \mathbb{E}\left[\mathbf{q}^{(i)}(O^D_{N_{>i}} \setminus A_j^{(i)})\right] \geq \left(\frac{c}{20\beta}\right) \cdot \mathbb{E}\left[\mathbf{q}^{(i)}(O^D_{N_{>i}} \setminus A_j^{(i)})\right]. \quad ◁$$

▷ Claim 25. (5.2 from [1]) For any j, we have

$$\frac{22\beta}{c} \cdot \mathbb{E}\left[\text{val}(A_j^{(i)})\right] + \mathbb{E}\left[\mathbf{q}^{(i)}(C_j^{(i)} \cap A_j^{(i)})\right] \geq \mathbb{E}\left[\mathbf{q}^{(i)}(C_j^{(i)})\right].$$

Proof. By combining Claim 23 and 24, we have

$$\begin{aligned}\left(\frac{20\beta}{c} + \frac{2}{c}\right)\mathbb{E}\left[\text{val}(A_j^{(i)})\right] &\geq \mathbb{E}\left[\mathbf{q}^{(i)}(O^D_{N>i} \setminus A_j^{(i)})\right] + \mathbb{E}\left[\mathbf{q}^{(i)}(O^D_{N_i} \setminus A_j^{(i)})\right] \\ &= \mathbb{E}\left[\mathbf{q}^{(i)}(C_j^{(i)} \setminus A_j^{(i)})\right].\end{aligned}$$

Because $O^D_{N_{\geq i}}$ is exactly $C^{(i)}_j$. Thus, we get

$$\frac{20\beta+2}{c}\cdot\mathbb{E}\left[\text{val}(A^{(i)}_j)\right]+\mathbb{E}\left[\mathbf{q}^{(i)}(C^{(i)}_j\cap A^{(i)}_j)\right]\geq\mathbb{E}\left[\mathbf{q}^{(i)}(C^{(i)}_j)\right].$$ ◁

The previous claim can be thought of as a preliminary version of the entire learnable-or-allocatable lemma. In expectation, we get something comparable to the items which are correctly priced in auction j of round i (i.e. $\mathbf{q}^{(i)}(C^{(i)}_j)$). The contribution come from either the items which sold in the round they were "supposed to" (i.e. $C^{(i)}_j\cap A^{(i)}_j$) or the welfare of the current allocation (i.e. $A^{(i)}_j$) (with an extra $O(\beta)$ factor). The previous claims dealt with individual auctions within an iteration – next we handle iterations as a whole.

We still have to account for two things: items which sell in auctions where the prices are *too high* and the loss in welfare from the fact that bidders in N_i will no longer be allocated items in later rounds. The proofs in [1] hold as written – only the properties of the price tree and the structure of the auctions are used.

▷ Claim 26. (5.5 from [1])

$$\mathbf{q}^{(i)}(C^{(i+1)})\geq\sum_{j=1}^{\alpha}\mathbf{q}^{(i)}(A^{(i)}_j\cap C^{(i)}_j)-\frac{OPT}{10\beta}.$$

Proof. The key observation here is that the set of "overpriced" items represent a small fraction of the optimal revenue. Let U be the set of items that are allocated in FixedPriceAuction with price above their correct price in round i. The set of items that are allocated in the correct round but not priced correctly is exactly $\left(\bigcup_j A^{(i)}_j\cap C^{(i)}_j\right)\setminus C^{(i+1)}$. This must be a subset of U. Thus, $\mathbf{q}^{(i)}(C^{(i+1)})\geq\sum_{j=1}^{\alpha}\mathbf{q}^{(i)}(A^{(i)}_j\cap C^{(i)}_j)-\mathbf{q}(U)$.

Consider an allocation that gives all items in U to the bidder in the highest priced auction where it is ever allocated. Such an allocation must give welfare $\leq OPT$, but $\geq\gamma\mathbf{q}^{(i)}(U)$ due to the price gap in the tree structure. Thus $\mathbf{q}^{(i)}(U)\leq\frac{1}{\gamma}OPT\leq\frac{OPT}{10\beta}$ (by choosing $\gamma=\theta(\log\log m)\geq 10\beta$) . ◁

▷ Claim 27. (5.6 from [1])

$$\mathbb{E}\left[\mathbf{q}^{(i+1)}(C^{(i+1)})\right]\geq\mathbb{E}\left[\mathbf{q}^{(i)}(C^{(i+1)})\right]-\frac{OPT}{10\beta}.$$

Proof. This follows simply from the fact that $\mathbf{q}^{(i+1)}$ is exactly $\mathbf{q}^{(i)}$ with items corresponding (under O) to bidders in N_i set to zero, and that bidders join N_i with probability $1/(10\beta)$. ◁

Proof of Learnable-Or-Allocable Lemma, Lemma 22. By Claim 26 and 27,

$$\mathbb{E}\left[\mathbf{q}^{(i+1)}(C^{(i+1)})\right]\geq\sum_{j=1}^{\alpha}\mathbb{E}\left[\mathbf{q}^{(i)}(A^{(i)}_j\cap C^{(i)}_j)\right]-\frac{OPT}{5\beta},\tag{2}$$

We now have two cases. First, assume

$$\sum_{j=1}^{\alpha}\mathbb{E}\left[\mathbf{q}^{(i)}(A^{(i)}_j\cap C^{(i)}_j)\right]\geq\mathbb{E}\left[\mathbf{q}^{(i)}(C^{(i)})\right]-\frac{2}{15\beta}OPT.\tag{3}$$

Together with (2) this immediately implies that

$$\mathbb{E}\left[\mathbf{q}^{(i+1)}(C^{(i+1)})\right] \geq \mathbb{E}\left[\mathbf{q}^{(i)}(C^{(i)})\right] - \frac{OPT}{3\beta}.$$

and we are in the "learnable case".

On the other hand, if equation (3) is false, then we can sum the inequality in claim 25 for each $j = 1, \ldots, \alpha$ to get

$$\begin{aligned}\mathbb{E}\left[\mathbf{q}^{(i)}(C^{(i)})\right] &\leq \frac{22\beta}{c}\sum_{j=1}^{\alpha}\mathbb{E}\left[\mathrm{val}(A_j^{(i)})\right] + \sum_{j=1}^{\alpha}\mathbb{E}\left[\mathbf{q}^{(i)}(C_j^{(i)} \cap A_j^{(i)})\right] \\ &< \frac{22\beta}{c}\sum_{j=1}^{\alpha}\mathbb{E}\left[\mathrm{val}(A_j^{(i)})\right] + \mathbb{E}\left[\mathbf{q}^{(i)}(C^{(i)})\right] - \frac{2}{15\beta}OPT.\end{aligned}$$

Thus

$$\begin{aligned}&\frac{22\beta}{c}\sum_{j=1}^{\alpha}\mathbb{E}\left[\mathrm{val}(A_j^{(i)})\right] \geq \frac{2}{15\beta}\cdot OPT \\ &\Rightarrow \mathbb{E}\left[\mathrm{val}(A_{j^*}^{(i)})\right] = \frac{1}{\alpha}\sum_{j=1}^{\alpha}\mathbb{E}\left[\mathrm{val}(A_j^{(i)})\right] \geq \frac{2c}{22 * 15\alpha\beta^2}\cdot OPT = \frac{c}{O(\alpha\beta^2)}\cdot OPT.\end{aligned}$$

and we are in the "allocatable" case. ◀

Theorem 16 now follows readily follow from the Learnable or Allocable Lemma.

▶ **Theorem 28.** *Suppose ψ_{min} and ψ_{max} are given and satisfy assumption 21. Suppose the optimal allocation O has supporting prices $\mathbf{q}$, and suppose bidders pick (c, d)-competitive sets in every fixed price auction they participate in. Then mechanism 3 achieves an $O\left(\max\left\{\frac{1}{c}, \frac{1}{d}\right\} \cdot (\log\log m)^3\right)$ approximation to the optimal welfare.*

Proof. Note that by the Learnable or Allocable Lemma, in the mechanism there are only two situation that can occur, 1) event E_1: "learnable" occurs in every iteration $i = 1, 2, ...\beta$, or 2) event E_2: "allocable" occurs in some iteration k. Denote the welfare from the mechanism as $Welf$. Then $\mathbb{E}[Welf]$ satisfy the equation

$$\mathbb{E}[Welf] \geq \min\left(\mathbb{E}[Welf \mid E_1], \mathbb{E}[Welf \mid E_2]\right).$$

Now we bound $\mathbb{E}[Welf \mid E_1]$ and $\mathbb{E}[Welf \mid E_2]$, respectively.

- Suppose that "learnable" occurs for each iteration $i = 1, 2, ...\beta$ in the mechanism. Because $C^{(1)}$ consist of items whose prices belong to the bins of P^*, we know that $\mathbb{E}\left[\mathbf{q}^{(1)}(C^{(1)})\right] = OPT/2$. Thus,

  $$\mathbb{E}\left[\mathbf{q}^{\beta+1}(C^{\beta+1})\right] \geq \mathbb{E}\left[\mathbf{q}^{(1)}(C^{(1)}) - \frac{OPT}{3}\right] = \frac{OPT}{2} - \frac{OPT}{3} = \frac{OPT}{6}.$$

  Let $W_{\beta+1}$ be the welfare achieved when the mechanism allocate in the last iteration of fixed price auction. Since for any correctly priced item $j \in C^{(\beta+1)}$, $\frac{1}{2}\mathbf{q}(j) \geq \mathbf{p}(j) \geq \frac{1}{\gamma}\mathbf{q}(j)$, by lemma 20, $W_{\beta+1} \geq \min\left(\frac{c}{2}, \frac{d}{\gamma}\right) \cdot \mathbb{E}\left[\mathbf{q}^{\beta+1}(C^{\beta+1})\right] = O\left(\min\left(c, \frac{d}{\beta}\right)\right) \cdot OPT$.
  It's easy to verify that the mechanism allocates in last iteration with constant probability. Thus, in this case we get $\mathbb{E}[Welf|E_1]$ at least $O\left(\min\left(c, \frac{d}{\beta}\right)\right) \cdot OPT$.

- In the case where "learnable" does not occur for some iteration i, "allocable" must occur at this iteration. Thus

$$\mathbb{E}\left[\text{val}(A^{(i)}_{j^*})\right] = \frac{c}{O(\alpha\beta^2)} \cdot OPT.$$

  The mechanism allocate in iteration i with probability $(1-1/\beta)^{i-1} \cdot 1/\beta = O(1/\beta)$, thus in this case $\mathbb{E}[Welf \mid E_2]$ is at least $\frac{c}{O(\alpha\beta^3)} \cdot OPT$.

Since $\beta = \Theta(\log\log m)$, we conclude that mechanism M achieves an approximation ratio of

$$O\left(\max\left(\frac{1}{c}, \frac{\beta}{d}, \frac{\alpha\beta^3}{c}\right)\right) = \max\left(\frac{1}{c}, \frac{1}{d}\right) \cdot O(\log\log m)^3. \quad \blacktriangleleft$$

B.4 Removing Assumptions

In this section we prove Theorem 16 in full generality by 1) removing the assumption that the supporting price lies in $\{0\} \cup [\psi_{min}, \psi_{max}]$, where $\psi_{max}/\psi_{min} = \text{poly}(m)$, and 2) showing that this generalized mechanism can be implemented in advised strategies. We use a similar (but slightly simplified) extension to PriceLearningMechanism following previous work on truthful mechanisms for XOS bidders [12, 18, 14, 1][8]. Our variation both simplifies the analysis and allows us to satisfy the formal definition of implementation in advised strategies more easily.

The final mechanism is as follows.

Algorithm 4 GeneralizedMechanism(N, M).

1: Pick a subset of bidders $N_{stat} \subseteq N$ by sampling each bidder in N independently and with probability $\frac{1}{2}$. Let $N_{mech} = N \setminus N_{stat}$.
2: Run a second price auction on the grand bundle M with bidders in N_{stat}. Let SPA be the welfare of the resulting allocation. With probability $\frac{1}{2}$, return the resulting allocation and terminate. With the remaining probability, continue.
3: Set $\psi_{min} = \frac{1}{4m^2} \cdot SPA$ and $\psi_{max} = 4m \cdot SPA$.
4: Run PriceLearningMechanism (Mechanism 3) on bidders in N_{mech} with ψ_{min} and ψ_{max} and return the allocation.

First, we show that implementation in advised strategies allows us to force bidders to play truthfully in the second-price auction of mechanism 4, and to pick (c, d)-competitive sets in the PriceLearningMechanism.

▶ **Lemma 29.** *Suppose we are given a (c, d)-approximate demand oracle D for valuations $\mathcal{V}$. Then there exists a useful poly-time computable advice A for mechanism 4 such that, if a strategy s is advised for v_i under A, then any bidder in N_{stat} will play truthfully in the second price auction, and any bidder in N_{mech} will pick (c, d)-competitive sets in every fixed price auction they participate in.*

Proof. As in prior works [1, 12, 14], to formally meet our solution concept we need all actions by a single bidder to happen simultaneously in order to preclude bidders from "threatening" each other (for example, if a different bidder will only let me have items in future auctions if

[8] Prior works have some probability of selling the grand bundle M in a second price auction (to handle "dominant bidders") or running a different algorithm to collect basic "statistics" on the bidders. We combine the two approaches by using the result of the second price auction to calculate the statistics (at the cost of some loss in the polynomial factor in assumption 21).

I lie in the current auction, then truthful play does not dominate lying). Thus, we formally implement GeneralizedMechanism as a game where each bidder can act in exactly one node. If the bidder is assigned in N_{stat}, the mechanisms simultaneously asks all bidders in N_{stat} for a single bid on the grand bundle. If the bidder is put in N_{mech}, and then into N_i for $i < \beta + 1$, then the bidder needs to participate in α fixed-price auctions simultaneously in a single game node. Thus, the bidder reports a list $(T_j)_{j=1,\dots,\alpha}$ of α subsets of M, where T_j is still available in auction j of the mechanism when it is bidder i's turn to pick a set. Bidders in $N_{\beta+1}$ report similarly, but participate in only one auction.

Recall that the advice function $A(v_i, x, a)$ takes as input the valuation function v_i of player i, a node x of the game, and a "tentative" action a which the player may play. The advice works as follows: for a node x which corresponds to a bidder in N_{stat}, A can ignore the tentative action a and recommend truthful play in the second price auction, i.e. $A(v_i, x, a) = v_i(M)$ in this case. If x corresponds to a bidder put in $N_i \subseteq N_{mech}$ for some $i < \beta$, then the tentative action a is some list of sets $(S_1, \dots, S_\alpha)$ which bidder i may choose in each auction. For each of the α auctions, A will run the (c, d)-approximate demand query D (with prices and remaining items known from the node x) to get a sets $T_1, \dots, T_\alpha$. Then, A will return $(S'_1, \dots, S'_\alpha)$, where S'_i is whichever of S_i or T_i that gives bidder i higher utility. The advice behaves similarly for bidders in $N_{\beta+1} \subseteq N_{mech}$.

It's clear that, if D is computable in poly-time, then $A(v_i, x, a)$ is computable in poly-time.

We now show that A is *useful* (definition 5). A satisfies the required idempotency property, because for bidders in N_{stat}, the result of A is a constant, and for bidders in N_{mech}, the result is given by taking the max of sets S_j with the result of D (which is fixed given bidder's valuation v_i and a node x of the game).

For any s_i and for any randomness in the mechanism, it's clear that $A^{v_i,s}$ gets i utility at least as high as s. For, if i is in N_{stat}, then $A^{v_i,s}$ recommends a dominant strategy, and if i is in N_{mech}, then the utility of i is completely determined by the unique node in which i is chosen to act, and $A^{v_i,s}$ will differ from s only in selecting sets with higher utility for i. Moreover, if $s \neq A^{v_i,s}$, then either s and $A^{v_i,s}$ differ for some node corresponding to a bidder in N_{stat}, or s and $A^{v_i,s}$ differ for some node corresponding to a bidder in N_{mech}. In the first case, because $A^{v_i,s}$ is dominant, there exists v_{-i} and random outcomes of the mechanism which get i strictly higher utility. In the second case, there must be some auction in which the advice $A^{v_i,s}$ selects strictly better sets than s, and because there is positive probability that each auction is the allocation returned by the mechanism, there are some random outcomes of the mechanism which get the bidder strictly more utility. Thus, if $A^{v_i,s} \neq s$, then $A^{v_i,s}$ dominates s.

Finally, it's clear that if a bidder plays according to strategy $A^{v_i,s}$for any s, then if the bidder is in N_{stat} then they play truthfully, and if the bidder is in N_{mech} then they select (c, d)-competitive sets. ◀

Now, we show that algorithm 4 successfully allows us to remove assumption 21. Let S be any set of bidders and let $OPT(S)$ denote the optimal welfare possible for bidders in S. We say (ψ_{min}, ψ_{max}) is *correct for* S if $\psi_{min} \leq OPT(S)/m^2$ and $\psi_{max} \geq OPT(S)$. We call a bidder i dominant for a set S if $v_i(O_i) > \frac{OPT(S)}{8}$.

▶ **Lemma 30.** *Let* $\mathbf{q}$ *be the supporting prices of an optimal allocation of items to bidders in some set* S*. If* (ψ_{min}, ψ_{max}) *is correct for* S*, then the supporting prices of a* $(1 - o(1))$ *fraction of the items (weighted by their supporting prices) are in the range* $I = [\psi_{min}, \psi_{max}]$*. More formally,* $\sum_{j \in M} \mathbb{1}[\mathbf{q}(j) \in I] \cdot \mathbf{q}(j) \geq (1 - \frac{1}{m}) \cdot OPT(S)$.

Proof. Since $\psi_{max} \geq OPT(S)$, we know that for all item j, $\mathbf{q}(j) \in [0, \psi_{max}]$. Now we count the sum over supporting prices of items whose supporting price is $\leq OPT(S)/m^2$.

$$\sum_{\mathbf{q}(j) \leq OPT(S)/m^2} \mathbf{q}(j) \leq m \cdot OPT(S)/m^2 = OPT(S)/m.$$

Thus

$$\sum_{j \in [m]} \mathbb{1}\left[\mathbf{q}(j) \in I\right] \cdot \mathbf{q}(j) \geq \sum_{j \in [m]} \mathbf{q}(j) - \sum_{\mathbf{q}(j) \leq OPT(S)/m^2} \mathbf{q}(j) \geq \left(1 - \frac{1}{m}\right) \cdot OPT(S). \quad \blacktriangleleft$$

▶ **Corollary 31.** *For bidders in S and items in M, if (ψ_{min}, ψ_{max}) is correct for S, and $\psi_{max}/\psi_{min} = \text{poly}(m)$, then* PriceLearningMechanism(S, M) *returns an allocation with expected welfare $\frac{1}{r} \cdot (1 - \frac{1}{m}) \cdot OPT(S)$, where $r = O\left(\max\left\{\frac{1}{c}, \frac{1}{d}\right\} (\log\log m)^3\right)$.*

Proof. Again let $\mathbf{q}$ be the supporting prices of an optimal allocation of items to bidders in some set S. Observe that although Theorem 28 assumes all supporting price to be in $0 \cup [\psi_{min}, \psi_{max}]$, the proof holds as is for approximating $\sum_{j \in [m]} \mathbb{1}\Big[\mathbf{q}(j) \in [\psi_{min}, \psi_{max}]\Big] \cdot \mathbf{q}(j)$ (i.e. the contribution to the optimal welfare of items whose supporting price is in $[\psi_{min}, \psi_{max}]$). If (ψ_{min}, ψ_{max}) is correct for S, then

$$\sum_{j \in [m]} \mathbb{1}\Big[\mathbf{q}(j) \in [\psi_{min}, \psi_{max}]\Big] \cdot \mathbf{q}(j) \geq (1 - \frac{1}{m}) \cdot OPT(S).$$

We conclude that PriceLearningMechanism returns an allocation with expected welfare

$$\frac{1}{r} \cdot \sum_{j \in [m]} \mathbb{1}\Big[\mathbf{q}(j) \in [\psi_{min}, \psi_{max}]\Big] \cdot \mathbf{q}(j) = \frac{1}{r} \cdot (1 - \frac{1}{m}) \cdot OPT(S),$$

where $r = O\left(\max\left\{\frac{1}{c}, \frac{1}{d}\right\} (\log\log m)^3\right)$. $\blacktriangleleft$

The following lemma follows from a standard application of chernoff bound and is quoted verbatim from [1]. It allows us to show that, with constant probability, a good fraction of the welfare is achievable by bidders in both N_{stat} and N_{mech}.

▶ **Lemma 32.** [12, 18, 14, 1] *Let $O = (O_1, ...O_n)$ be an optimal allocation of items M to bidders N with welfare OPT. Suppose we sample each $i \in N$ w.p. ρ independently to obtain N'. If for every $i \in N$, we have $v_i(O_i) \leq \epsilon \cdot OPT$, then $\sum_{i \in N'} v_i(O_i) \geq (\rho/2) \cdot OPT$ w.p. at least $1 - 2 \cdot \exp(-\frac{\rho}{2\epsilon})$.*

Finally, once the previous lemma has been applied, we will need this lemma to prove that we set the parameters correctly for PriceLearningMechanism.

▶ **Lemma 33.** *If N_{stat} satisfy $OPT(N_{stat}) \geq \frac{1}{4} \cdot OPT$ and $OPT(N_{mech}) \geq \frac{1}{4} \cdot OPT$, then (ψ_{min}, ψ_{max}) is correct for N_{mech}.*

Proof. Assume N_{stat} satisfy $OPT(N_{stat}) \geq \frac{1}{4} \cdot OPT$ and $OPT(N_{mech}) \geq \frac{1}{4} \cdot OPT$. We know that $SPA < OPT(N_{stat})$. Thus

$$4 \cdot OPT(N_{mech}) \geq OPT \geq OPT(N_{stat}) \geq SPA,$$
$$\Rightarrow \psi_{min} = \frac{1}{4m^2} \cdot SPA \leq \frac{1}{m^2} \cdot OPT(N_{mech}).$$

Moreover, since SPA is at least the value of M for any bidder in N_{stat}, we have $m \cdot SPA \geq OPT(N_{stat})$. Thus

$$\psi_{max} = 4m \cdot SPA \geq 4 \cdot OPT(N_{stat}) \geq OPT \geq OPT(N_{mech}).$$

We conclude that (ψ_{min}, ψ_{max}) is correct for N_{mech}. ◀

▶ **Theorem 34.** *For valuation functions $v_1, \ldots, v_n$, suppose the optimal allocation O has supporting prices $\mathbf{q}$. Let D be a (c,d)-approximate demand oracle for valuation in $\{v_1, \ldots, v_n\}$. Then mechanism 4 with advice A as in lemma 29 gets a $O\left(\max\left\{\frac{1}{c}, \frac{1}{d}\right\} \cdot (\log\log m)^3\right)$ fraction of the optimal welfare in implementation in advised strategies.*

Proof. Lemma 29 shows that there exists poly time computable advice such than, whenever a bidder in N_{stat} follows advice, they play truthfully, and whenever a bidder in N_{mech} follows advice, they pick (c,d)-competitive sets in every fixed price auction they participate in.

Recall that a bidder is *dominant* if they contribute more than a 1/8 fraction of the welfare of an optimal allocation. Next we show that whether there is a dominant bidder or not, the expected welfare from GeneralizedMechanism is an $O\left(\max\left\{\frac{1}{c}, \frac{1}{d}\right\}(\log\log m)^3\right)$ approximation to OPT in implementation in advised strategy with advice B.

- When there is a dominant bidder, then with $\frac{1}{2}$ probability the dominant bidder would be selected in the N_{stat} group. Conditioned on this, with $\frac{1}{2}$ probability the resulting allocation from running second price auction on the N_{stat} group would be realized. Since a dominant bidder is in N_{stat} group, the welfare from the second price auction is at least $\frac{OPT}{8}$. Thus the expected welfare of GeneralizedMechanism, conditioned on there being a dominant bidder, is at least $\frac{1}{2} \cdot \frac{1}{2} \cdot \frac{OPT}{8} = \frac{OPT}{32}$.
- When there is no dominant bidder, then by Lemma 32, $OPT(N_{stat}) \geq \frac{1}{4} \cdot OPT$ with probability at least $1 - 2e^{-2}$, which means $OPT(N_{stat}) < \frac{1}{4} \cdot OPT$ with probability $< 2e^{-2}$. Symmetrically, $O(N_{mech}) < \frac{1}{4} \cdot OPT$ with probability $< 2e^{-2}$. By union bound, both $OPT(N_{stat})$ and $OPT(N_{mech})$ is $\geq \frac{1}{4} \cdot OPT$ with probability at least $1 - 4e^{-2}$, which is still a positive, constant probability.
  Let's call the event where $OPT(N_{stat}) \geq \frac{1}{4} \cdot OPT$ and $OPT(N_{mech}) \geq \frac{1}{4} \cdot OPT$ the *good* event.
  By Lemma 33, if the good event occurs, then (ψ_{min}, ψ_{max}) is correct for N_{mech}. By construction in GeneralizedMechanism, $\psi_{max}/\psi_{min} = O(m^3)$. By Corollary 31, conditioned on ψ_{min} and ψ_{max} begin set correctly and $\psi_{max}/\psi_{min} = poly(m)$, priceLearningMechanism returns an allocation that achieves welfare $\frac{1}{r} \cdot (1 - \frac{1}{m}) \cdot OPT(N_{mech})$, where $r = O\left(\max\left\{\frac{1}{c}, \frac{1}{d}\right\}(\log\log m)^3\right)$. Since the good event occurs, $OPT(N_{mech}) \geq \frac{1}{4} \cdot OPT$. We conclude that conditioned on the good event, the expected welfare from PriceLearningMechanism $O\left(\max\left\{\frac{1}{c}, \frac{1}{d}\right\}(\log\log m)^3\right)$ approximates OPT. As the event "the good event happens and GeneralizedMechanism runs PriceLearningMechanism in setp 4" occurs with constant probability, we conclude that Generalized mechanism achieves expected welfare at least $OPT/O\left(\max\left\{\frac{1}{c}, \frac{1}{d}\right\}(\log\log m)^3\right)$ when there is no dominant bidder.

Together with the fact that every allocation for XOS valuation functions has supporting prices, we immediately get theorem 16.

▶ **Theorem 16.** *Let $\mathcal{V}$ be a subclass of XOS valuations and let D be a poly-time (c,d)-approximate demand oracle for valuation class $\mathcal{V}$. Then there exists a poly-time mechanism for welfare maximization when all valuations are in $\mathcal{V}$ with approximation guarantee $O\left(\max\left\{\frac{1}{c}, \frac{1}{d}\right\} \cdot (\log\log m)^3\right)$ in implementation in advised strategies with polynomial time computable advice.* ◀

C Approximate Demand Queries vs. Approximate Welfare Approximation

As it happens, both SimpleGreedy and SingleOrBundle were inspired by simple known algorithms for approximate welfare maximization, combined with the following simple observation:

▶ **Proposition 35.** *S is the return of a demand query on prices $\mathbf{p}$ if and only if $(S, M \setminus S)$ is a welfare maximizing bundle for the following two player auction: one bidder has valuation function v, and the other bidder has additive valuation function given by $\mathbf{p}$.*

Proof. The utility of v is $v(S) - \mathbf{p}(S)$, which differs from the welfare $v(S) + \mathbf{p}(M \setminus S)$ only by the constant $\mathbf{p}(M)$. So maximizing these two objectives is equivalent. ◀

In particular, SimpleGreedy is exactly the 2-approximation algorithm from [33] played by a regular bidder and a "price bidder". SingleOrBundle is similarly inspired by the $\sqrt{m}$ approximation of [17]. However, we show below that approximate demand queries do not, in general, reduce to approximate welfare maximization.

▶ **Example 36.** Consider a budget additive valuation v with value 2 for every item and budget of $2\sqrt{m}$. That is, $v(S) = \max\{2|S|, 2\sqrt{m}\}$. Let $\mathbf{p}$ have price 1 for each item, i.e. $\mathbf{p}(S) = |S|$. The result of a demand query on $(v, \mathbf{p})$ is any set of size $\sqrt{m}$, with utility $\sqrt{m}$.

However, consider running an approximate welfare maximization mechanism $\mathcal{A}$ with two bidders: one with valuation $v(S)$ for bundle S and one with valuation $\mathbf{p}(S)$ for bundle S. The optimal allocation is to give any S of size exactly $\sqrt{m}$ to v, and give the rest of the items to $\mathbf{p}$. This has welfare $m + \sqrt{m}$. However, the allocation giving every item to $\mathbf{p}$ has welfare m. Thus, any constant factor approximation algorithm (for which no other grantees hold) may return this allocation, as $m + \sqrt{m} = (1 + o(1))m$.

This corresponds to an approximate demand query giving the bidder the empty set. As this has zero utility, it will fail to be any factor approximation ration of the optimal.

Moreover, the above example would still go through if we consider a few simple variations on the reduction given by Proposition 35. For example, if we discount prices by a constant factor, say d, it's still the case that $d(m - \sqrt{m}) + 2\sqrt{m} = (1 + o(1))dm$, so a constant-factor approximation algorithm $\mathcal{A}$ might give all items to the "price player".

Thus, approximate demand queries do not reduce to approximate welfare maximization (at least not as outlined by Proposition 35).

D Other Algorithms for approximate demand oracles

Here we give another algorithm for computing a $(1/2, 1/2)$-approximate demand oracle. Instead of being inspired by known welfare maximization algorithms, this technique is inspired by known submodular maximization algorithms. Namely, the algorithm MeetInMiddle below is exactly the algorithm DeterministicUSM from [5], run on the submodular function f given by $f(S) = v(S) - \mathbf{p}(S)$. When f is a nonnegative (possibly decreasing) submodular function, [5] shows that it gives a $1/3$ approximation to the maximum value of f. Unfortunately, the submodular utility function we are interested in is possibly negative, so this result does not apply (indeed, it is NP hard to achieve *any* nontrivial approximation ration for possibly negative submodular maximization, as we discussed in section 3).

Algorithm 5 MeetInMiddle$(v, \mathbf{p}, M)$.

$X \leftarrow \emptyset$ and $Y \leftarrow M$
for $j = 1, \ldots, m$ **do** ▷ For items in an arbitrary order
  Set $a_j \leftarrow v(X \cup j) - v(X) - \mathbf{p}(j)$ ▷ Invariant: $Y = X \cup \{j, \ldots, m\}$
  Set $b_j \leftarrow v(Y \setminus j) - v(Y) + \mathbf{p}(j)$
  if $a_j \geq b_j$ **then**
    Set $X \leftarrow X \cup j$
  else
    Set $Y \leftarrow Y \setminus j$
return X ▷ or return Y (as $X = Y$ by now)

For SimpleGreedy and SingleOrBundle, we needed to run an existing algorithm with the "higher" prices $\mathbf{p}/d$ to attain a (c, d)-approximate demand oracle for (i.e. a set S for which $v(S) - \mathbf{p}(S) \geq c \max_T v(T) - \mathbf{p}(T)/d$). Interestingly, we show that MeetInMiddle need to take the lower ("discounted") prices as input in order to provide an approximation guarantee.

We show that

1. For any $\epsilon > 0$, $S =$MeetInMiddle$(v, \mathbf{p}, M)$ is not a (ϵ, ϵ) approximate demand oracle for prices $\epsilon\mathbf{p}$ (i.e. there exists a valuation function v such that $v(S) - \epsilon\mathbf{p}(S) < \max_T\{v(T) - \mathbf{p}(T)\}$).
2. MeetInMiddle$(v, \mathbf{p}/2, M)$ is an $(\frac{1}{2}, \frac{1}{2})$ approximate demand oracle for prices $\mathbf{p}/2$ (i.e. for any submodular v we have $v(S) - \mathbf{p}(S)/2 \geq \frac{1}{2}\max_T\{v(T) - \mathbf{p}(T)\}$).

▶ **Example 37.** For any $\epsilon > 0$, let $K = 4/\epsilon$ and $N = 2 + (K-1)/\epsilon$ and $M = \{1, 2, ...N\}$. Consider the price vector $\mathbf{p}(1) = \frac{K}{2} + 1$ and $\forall i > 1 : \mathbf{p}(i) = 1 - \epsilon$ and the bidder valuation function $v(S) = K$ for any $S \ni 1$ and $v(S) = 1 + (|S| - 1)\epsilon$ for any $S \ni 1$. One can check that the valuation function is submodular.

MeetInMiddle will remove the first item from X, since $v(M-1) - v(M) + \mathbf{p}(1) = \frac{K}{2} + 1 > \frac{K}{2} - 1 = v(1) - \mathbf{p}(1)$. Similarly, one can check that the algorithm will then remove all items except the last item N, which it will keep. Thus the algorithm returns set $T = \{N\}$, so $v(T) - \mathbf{p}(T) = \epsilon$.

However, the optimal set is $O = \{1\}$. We have $\epsilon(v(O) - \mathbf{p}(O)) = \epsilon(\frac{2}{\epsilon} - 1) = 2 - \epsilon$. Thus $v(T) - \epsilon(p(T)) < \epsilon(v(O) - \mathbf{p}(O))$, and MeedInMiddle$(v, \mathbf{p}, M)$ is not a (ϵ, ϵ) approximate demand oracle for all constant $1 > \epsilon > 0$.

▷ Claim 38. If v is submodular, $S =$MeetInMiddle$(v, \mathbf{p}/2, M)$ is an $(\frac{1}{2}, \frac{1}{2})$ approximate demand oracle for prices $\mathbf{p}/2$ (i.e. $v(S) - \mathbf{p}(S)/2 \geq \frac{1}{2}\max_T\{v(T) - \mathbf{p}(T)\}$.

Proof. Let $T \leftarrow$MeetInMiddle$(v, \mathbf{p}/2, M)$ and $O = \arg\max_{S \subseteq M} v(S) - \mathbf{p}(S)$. We use induction on $|M|$.

For the base case, let $|M| = 1$. Observe that $a_1 = v(1) - \mathbf{p}(1)/2 = -b_1$. Thus, $T = \emptyset$ only when $v(1) \geq \mathbf{p}(1)/2$, so T is exactly $\arg\max_S v(S) - \mathbf{p}(S)/2 \geq v(O) - \mathbf{p}(O)$.

Now, let $|M| > 1$, and assume by induction that the claim is true for all $m' < |M|$. Consider the following two cases:

- If $1 \notin T$, then

$$v(1) - \frac{\mathbf{p}(1)}{2} = a_1 < b_1 = v(M \setminus 1) - v(M) + \frac{\mathbf{p}(1)}{2}$$
$$\Rightarrow \mathbf{p}(1) > v(1) + v(M) - v(M \setminus 1) \geq v(1). \qquad (*)$$

  thus $1 \notin O$. If $M' = M \setminus 1$, then $T =$MeetInMiddle$(v, \mathbf{p}/2, M')$ and $O = \arg\max_{S \subseteq M'} v(S) - \mathbf{p}(S)$. By the inductive hypothesis, $v(T) - \frac{1}{2}\mathbf{p}(T) \geq \frac{1}{2}(v(O) - \mathbf{p}(O))$.

- Suppose $1 \in T$. Let $M' = M \setminus 1$ and let O_2 be the set that maximizes utility on M' for v at prices $\mathbf{p}$ (i.e. $O_2 = \arg\max_{S \subseteq M'} v(S) - \mathbf{p}(S)$). The negation of $(*)$ plus the submodularity of v tells us that

$$\mathbf{p}(1) \leq v(1) + v(M) - v(M \setminus 1) \leq v(1) + v(O_2 \cup 1) - v(O_2). \tag{$\dagger$}$$

Define a new submodular function v' on M' such that $v'(S) = v(S \cup 1) - v(1)$ for all $S \subseteq M'$. One can check that an item > 1 is added to X in MeetInMiddle$(v, \mathbf{p}/2, M)$ if and only if it is added to X in MeetInMiddle$(v', \mathbf{p}/2, M')$. Thus, $T \setminus 1 =$MeetInMiddle$(v', \mathbf{p}/2, M')$, and the inductive hypothesis tells us that $v'(T \setminus 1) - \frac{1}{2}\mathbf{p}(T \setminus 1) \geq \frac{1}{2}(v'(O_2') - \mathbf{p}(O_2'))$, where O_2' is the set that maximizes utility on M' for v' on prices $\mathbf{p}$ (i.e. $O_2' = \arg\max_{S \subseteq M'} v'(S) - \mathbf{p}(S)$).
We now analyze two subcases:

- If $1 \in O$, then $O = 1 \cup O_2'$. Thus, applying the inductive hypothesis we know

$$\begin{aligned} v(T) - \frac{\mathbf{p}(T)}{2} &= v(1) - \frac{1}{2}\mathbf{p}(1) + (v'(T \setminus 1) - \frac{1}{2}\mathbf{p}(T \setminus 1)) \\ &\geq v(1) - \frac{1}{2}\mathbf{p}(1) + \frac{1}{2}(v'(O_2') - \mathbf{p}(O_2')) \\ &\geq \frac{1}{2}\big(v(1) + v'(O_2') - \mathbf{p}(1) - \mathbf{p}(O_2')\big) = \frac{1}{2}(v(O) - \mathbf{p}(O))). \end{aligned}$$

- If $1 \notin O$, then $O = O_2$. By rearranging $(\dagger)$, we get

$$\begin{aligned} &\mathbf{p}(1) \leq 2v(1) + v'(O_2) - v(O_2) \\ &\Rightarrow v(O_2) - v'(O_2) \leq 2\left(v(1) - \frac{\mathbf{p}(1)}{2}\right). \end{aligned} \tag{$\S$}$$

Thus

$$\begin{aligned} \frac{1}{2}(v(O) - \mathbf{p}(O)) = \frac{1}{2}\big(v(O_2) - \mathbf{p}(O_2)\big) &= \frac{1}{2}\big(v'(O_2) - \mathbf{p}(O_2) + v(O_2) - v'(O_2)\big) \\ &\leq \frac{1}{2}\big(v'(O_2') - \mathbf{p}(O_2') + v(O_2) - v'(O_2)\big) \\ &\leq v'(T \setminus 1) - \frac{\mathbf{p}(T \setminus 1)}{2} + v(1) - \frac{\mathbf{p}(1)}{2} \\ &= v(T) - \frac{\mathbf{p}(T)}{2}. \end{aligned}$$

Where the first inequality follows from the definition of O_2', and the second follows from the inductive hypothesis combined with $(\S)$. ◁

Toward a General Complexity Theory of Motion Planning: Characterizing Which Gadgets Make Games Hard

Erik D. Demaine
MIT Computer Science and Artificial Intelligence Laboratory,
32 Vassar Street, Cambridge, MA 02139, USA
edemaine@mit.edu

Dylan H. Hendrickson
MIT Computer Science and Artificial Intelligence Laboratory,
32 Vassar Street, Cambridge, MA 02139, USA
dylanhen@mit.edu

Jayson Lynch
MIT Computer Science and Artificial Intelligence Laboratory,
32 Vassar Street, Cambridge, MA 02139, USA
jaysonl@mit.edu

Abstract

We begin a general theory for characterizing the computational complexity of motion planning of robot(s) through a graph of "gadgets", where each gadget has its own state defining a set of allowed traversals which in turn modify the gadget's state. We study two general families of such gadgets within this theory, one which naturally leads to motion planning problems with polynomially bounded solutions, and another which leads to polynomially unbounded (potentially exponential) solutions. We also study a range of competitive game-theoretic scenarios, from one player controlling one robot to teams of players each controlling their own robot and racing to achieve their team's goal. Under certain restrictions on these gadgets, we fully characterize the complexity of bounded 1-player motion planning (NL vs. NP-complete), unbounded 1-player motion planning (NL vs. PSPACE-complete), and bounded 2-player motion planning (P vs. PSPACE-complete), and we partially characterize the complexity of unbounded 2-player motion planning (P vs. EXPTIME-complete), bounded 2-team motion planning (P vs. NEXPTIME-complete), and unbounded 2-team motion planning (P vs. undecidable). These results can be seen as an alternative to Constraint Logic (which has already proved useful as a basis for hardness reductions), providing a wide variety of agent-based gadgets, any one of which suffices to prove a problem hard.

2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness

Keywords and phrases motion planning, computational complexity, NP, PSPACE, EXP, NEXP, undecidability, games

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.62

Related Version A full version of the paper is available at `https://arxiv.org/abs/1812.03592`.

Acknowledgements This work grew out of an open problem session from MIT class on Algorithmic Lower Bounds: Fun with Hardness Proofs (6.890) from Fall 2014.

1 Introduction

Most hardness proofs are based on **gadgets** – local fragments, each often representing corresponding fragments of the input instance, that combine to form the overall reduction. Garey and Johnson [10] called gadgets "basic units" and the overall technique "local replace-

ment proofs". The search for a hardness reduction usually starts by experimenting with small candidate gadgets, seeing how they behave, and repeating until amassing a sufficient collection of gadgets to prove hardness.

This approach leads to a natural question: what gadget sets suffice to prove hardness? There are many possible answers to this question, depending on the precise meaning of "gadget" and the style of problem considered. Schaefer [17] characterized the complexity of all **Boolean constraint satisfiability** gadgets, with a dichotomy between polynomial problems (e.g., 2SAT, Horn SAT, dual-Horn SAT, XOR SAT) and NP-complete problems (e.g., 3SAT, 1-in-3SAT, NAE 3SAT). At STOC'97, Khanna, Sudan, Trevisan, and Williamson [13] refined this result to characterize **approximability** of constraint satisfaction problems, forking into polynomial, APX-complete, Poly-APX-complete, Nearest-Codeword-complete, and Min-Horn-Deletion-complete. Introduced at CCC'08, **Constraint Logic** [7, 11] proves sufficiency of small sets of gadgets on directed graphs that always satisfy one local rule (weighted in-degree at least 2), in many game types (1-player, 2-player, and team games, both polynomially bounded and unbounded), although the exact minimal sets of required gadgets remain unknown.

The aforementioned general techniques naturally model "global" moves that can be made anywhere at any time (while satisfying the constraints). Nonetheless, the techniques have been successful at proving hardness for problems where moves must be made local to an agent/robot that traverses the instance. For single-player agent-based problems, the **doors-and-buttons** framework (described in [9] and improved by [19] and [18]) is a good example of classifying a universe of abstract motion planning problems which can then be applied. In addition, the **door gadget** used to prove Lemmings [20] and various Nintendo games [2] PSPACE-complete served as a primary example of the form of gadget we wanted to generalize.

In this paper, we analyze which gadgets suffice for hardness in a general **semi-static motion planning problem** where one or more agents/robots traverse a given environment, which only changes in response to the agent's actions, from given start location(s) to given goal location(s). We study a very general model of gadget, where the gadget changes state when it gets traversed by an agent according to a general transition function, enabling and/or disabling certain traversals in the future. We study this model from the traditional single-robot (one-player) perspective, extending our initial work on this case [6], as well as from the perspective of two robots or teams of robots competing to reach their respective goals. We also analyze natural settings where the number of moves is polynomially bounded, because each gadget can be traversed only a bounded number of times, or more general settings where gadgets can be re-used many times and thus the number of moves can be exponential in the environment complexity. In each case, we partially or fully characterize which gadgets suffice to make the motion planning problem hard (NP-hard, PSPACE-hard, EXPTIME-hard, NEXPTIME-hard, or RE-hard, depending on the scenario), and conversely which gadgets result in a polynomially solvable problem (NL or P). Table 1 summarizes our results.

1.1 Gadget Model and Motion-Planning Games

In general, we model a **gadget** as consisting of a finite number of **locations** (entrances/exits) and a finite number of **states**; see Figure 1 for two examples. We may also consider a family of gadgets parameterized by the problem size. In this case we restrict the number of locations and states to be polynomial in the size of the problem. Each state s of the gadget defines a labeled directed graph on the locations, where a directed edge (a, b) with label s'

Table 1 Summary of our results for k-tunnel gadgets (with additional constraints listed in the left column). A "full characterization" means that we give an easily checkable condition on the allowed gadget set that determines the complexity of the corresponding motion planning problem; a "partial characterization" means that we give two easily checkable conditions on the allowed gadget set, one for the easy class and one for the hard class, each of which suffices to establish the complexity of the corresponding motion planning problem.

	1-Player Game	2-Player Game	Team Game
Polynomially Bounded (DAG gadgets)	**NL** vs. **NP-complete**: full characterization [§5]	**P** vs. **PSPACE-complete**: full characterization [§6]	**P** vs. **NEXPTIME-complete**: full characterization [§7]
Polynomially Unbounded (reversible, deterministic gadgets)	**NL** vs. **PSPACE-complete**: full characterization [§2] **Planar**: equivalent [§2.3]	**P** vs. **EXPTIME-complete**: partial characterization [§3]	**P** vs. **RE-complete** (⇒ **Undecidable**): partial characterization [§4]

means that a robot can enter the gadget at location a and exit at location b, and that such a traversal forcibly changes the state of the gadget to s'. Equivalently, a gadget is specified by its **transition graph**,[1] a directed graph whose vertices are state/location pairs, where a directed edge from (s, a) to (s', b) represents that the robot can traverse the gadget from a to b if it is in state s, and that such traversal will change the gadget's state to s'. Gadgets are **local** in the sense that traversing a gadget does not change the state of any other gadgets.

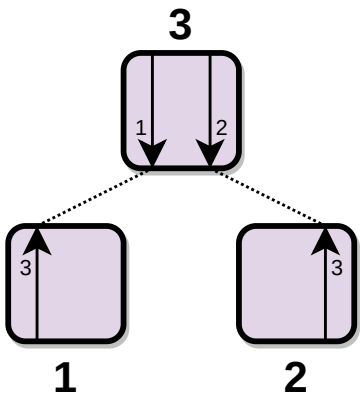
A **system of gadgets** consists of gadgets, their initial states, and a **connection graph** on the gadgets' locations. [2] If two locations a, b of two gadgets (possibly the same gadget) are connected by a path in the connection graph, then the robot can traverse freely between a and b (outside the gadgets). (Equivalently, we can think of locations a and b as being identified, effectively contracting connected components of the connection graph.) These are all the ways that the robot can move: exterior to gadgets using the connection graph, and traversing gadgets according to their current states.

We define a general family of **motion planning** problems involving one or more robots, each with their own start and goal location, in a system of gadgets. In a **one-player game**, we are given a system of gadgets, a single robot that starts at a specified start location, and we want to decide whether there is a sequence of moves that brings the robot to a specified goal location. (This problem is perhaps the most common setting for robot motion planning.)

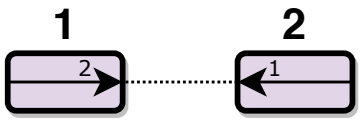
In a **two-player game**, we are given a system of gadgets and the start and goal locations of two robots, and two players alternate moving their own robot by traversing a single gadget (entering at a location reachable from the robot's current location via the connection graph). Both players have complete information about the locations of the robots, the locations of the gadgets, and the states of the gadgets. Here we count gadget traversals as costing one move, and view movement in the connection graph as instantaneous/free. The goal is to decide whether the first player has a **forced win**, that is, their robot can reach their goal

[1] In [6], the transition graph is called the "state space", but we feel that "transition graph" more clearly captures the automaton nature of transitions, which are discrete and directed.

[2] In [6], locations could only be matched to exactly one other location and a 'branching hallway' gadget was introduced to fulfill the need of the connection graph.



(a) The locking 2-toggle gadget (L2T). In the top state 3, you can traverse either tunnel going down, which blocks off the other tunnel until you reverse the initial traversal.



(b) The 1-toggle gadget. Traversing the tunnel reverses the direction that it can be traversed.

Figure 1 Basic gadgets that can be simulated by any interacting-k-tunnel reversible deterministic gadget, as shown in Section 2.1.

location before the second player's does, no matter how the second player responds to the first player's moves. In a **team game**, there are more than two robots, each controlled by a single player, and the robots/players are partitioned into two teams; the goal of each team is to get any one of its player's robot to their goal location. Crucially, after a team game begins, each player has only partial information of the current gadgets' states: they can only see the state of the gadgets reachable by their robot via the connection graph.

We also define **planar motion planning**. In this case, the cyclic order of locations on a gadget is specified, and the system of gadgets must be embedded in the plane without intersections. Specifically, construct the following graph from a system of gadgets: replace each gadget with a wheel graph, which has a cycle of vertices corresponding to the locations on the gadget in the appropriate order, and a central vertex connected to each location. Connect locations on these wheels with edges according to the connection graph. The system of gadgets is **planar** if this graph is planar. In planar motion planning, we restrict the problem to planar systems of gadgets. Note that this allows rotations and reflections of gadgets, but no other permutation of their locations. In some contexts, one may want to disallow reflections of gadgets, which corresponds to imposing a handedness constraint on the planar embedding of each wheel.

1.2 Gadget Types

We define different subclasses of gadgets that naturally model motion planning where the number of moves is either polynomially bounded or unbounded (potentially exponential).

In both cases, we require that the various states of a gadget differ only in their orientations of the possible traversals. More precisely, a ***k*-tunnel** gadget has $2k$ locations, paired in a perfect matching whose pairs are called **tunnels**, such that each state defines which direction(s) each tunnel can be traversed. All of the gadgets we consider in this paper are k-tunnel.

In the polynomial case, we focus on "DAG" gadgets. First define the **state-transition (multi)graph** of a gadget to have a vertex for each state, and a directed edge from s to s' for each possible traversal of the gadget in state s that leads to state s'. (This graph can be

obtained from the transition graph by combining together all vertices with the same state.) Then a gadget is a **DAG** if its state-transition graph is a directed acyclic graph. Such gadgets naturally lead to polynomially bounded motion planning, as every gadget traversal consumes potential within that gadget, as measured by the state (e.g., in a topological ordering of the state-transition graph). The total number of traversals is thus bounded by the total number of states in all gadgets in the system. (It is not enough to require that the transition graph be acyclic, because the robot can use the connection graph and other gadgets to reach other locations of this gadget in between traversals.)

In the polynomially unbounded case, we focus on gadgets that are "deterministic" and "reversible". A gadget is **deterministic** if its transition graph has maximum out-degree ≤ 1; i.e., a robot entering the gadget at some location in some state can exit at only one location and in only one state. A gadget is **reversible** if its transition graph has the reverse of every edge, i.e., it is the bidirectional version of an undirected graph. Thus a robot can immediately undo any gadget traversal.[3] Together, determinism and reversibility are equivalent to requiring that the transition graph is the bidirectional version of a matching.

We also consider **planar motion planning** problems with a **planar** system of gadgets, where the gadgets and connections are drawn in the plane without crossings. Planar gadgets are drawn as small regions (say, disks) with their locations as points in a fixed clockwise order along their boundary. A single gadget type thus corresponds to multiple planar gadget types, depending on the choice of the clockwise order of locations. Connections are drawn as paths connecting the points corresponding to the endpoint locations, without crossing gadget interiors or other connections.

The gadget model described above is an extension of the model introduced in [6], which characterized 2-state deterministic reversible k-tunnel gadgets that make for PSPACE-complete one-player games (polynomially unbounded), and showed that this characterization is the same when restricting to planar systems of gadgets. This prior result corresponds to the 2-state special case of our result in the bottom-left cell of Table 1. In this paper, we generalize that characterization to gadgets with arbitrarily many states, and generalize to 2-player games, team games, and (polynomially bounded) DAG k-tunnel gadgets.

1.3 Our Characterizations

In each type of motion planning problem where we obtain a full characterization of easy vs. hard gadget sets (bounded one-player, bounded two-player, bounded team, and unbounded one-player), we identify a class of gadgets such that motion planning with any single gadget in that class is hard, while motion planning with any collection of gadgets not in the class is easy. Thus, we do not see a difference in hardness between one and multiple gadget types; it is not possible for two "easy" gadgets to combine into a hard motion planning problem. This result is in surprising contrast to Constraint Logic where multiple gadgets were required for hardness in any setting.

For one-player motion planning, the key property of a gadget is **interacting tunnels**: the traversal of some tunnel must affect the traversability of some other tunnel in the same gadget.[4] In the unbounded case (Section 2), we show that any such gadget (that is also

[3] This notion is different than the sense of "reversible" in reversible computing, which would mean that we could derive which move to undo from the current state; here the undoing move only needs to be an option.

[4] This is roughly what [6] calls 'non-trivial' gadgets.

reversible and deterministic) can be used to simulate two specific gadgets, the "locking 2-toggle" and "1-toggle" (shown in Figure 1), which together suffice to prove PSPACE-hardness. This argument involves surprisingly little case analysis, in contrast to the prior work in this area [6], which simply enumerated and analyzed all 2-state gadgets. On the other hand, we show that any fixed collection of gadgets without interacting tunnels reduces (via a shortcutting argument) to graph traversal, which can be solved in NL. We furthermore show that this dichotomy still holds for 1-player planar motion planning (Section 2.3). In the bounded case (Section 5), we examine the naturally bounded class of DAG gadgets. We again obtain a somewhat more complicated full characterization, which mostly depends on the existence of interacting tunnels.

For two-player motion planning, it turns out that interacting tunnels are not required for hardness. In the bounded case (Section 6), we show that PSPACE-completeness holds for any DAG gadget that is **nontrivial**, i.e., has at least one transition in some state. We show that any nontrivial DAG gadget can simulate one of two one-tunnel gadgets, "single-use unidirectional edge" or "single-use bidirectional edge", and surprisingly either suffices to prove PSPACE-completeness. A single use-use edge is a transition in a gadget such that after taking that transition, there are no further transitions between the two associated locations. Obviously, two-player motion planning with trivial gadgets is in P: the robots can only traverse the connection graph, and one merely needs to see which is closer to their goal. In the unbounded case (Section 3), we show that any gadget with interacting tunnels suffices for EXPTIME-completeness, and it remains an open problem whether some weaker condition suffices.

For team motion planning, interacting tunnels are again not required for hardness. In the bounded case (Section 7), we show that NEXPTIME-completeness holds for any nontrivial DAG gadget, again by showing that any single-use edge gadget suffices. In the unbounded case (Section 4), we again show that any gadget with interacting tunnels suffices for undecidability, and it remains an open problem whether some weaker condition suffices.

Armed with the general framework of this paper, it should be much easier to prove hardness of most games that involve motion planning of robots in an environment with nontrivial local state. You simply need to pick a gadget that is hard according to our characterization (with the matching boundedness and number of players/teams), draw a single figure of how to build that gadget within the game of interest, and check that it is possible to connect these gadgets together. While this paper focuses on general theory building, we return to possible applications in Section 8.

2 1-Player Unbounded Motion Planning

In this section, we study reversible, deterministic gadgets, extending the work in [6] which only considered gadgets with two or fewer states. Here we give a complete categorization as either in NL or PSPACE-complete for reversible, deterministic gadget. For the NL half of the characterization, Theorem 2 below shows that 1-player motion planning problems with non-interacting-k-tunnel gadgets is in NL. For the PSPACE-completeness half of the characterization, we introduce a new base gadget, the **locking 2-toggle (L2T)** shown in Figure 1a. In Section 2.1 we show that all interacting-k-tunnel reversible deterministic gadgets are able to simulate the locking 2-toggle. Then in Section 2.2 we show that 1-player motion planning with locking 2-toggles is PSPACE-complete by simulating Nondeterministic Constraint Logic. Section 2.2 shows how to adapt the construction to show these gadgets remain PSPACE-hard even for the planar 1-player motion planning problem.

▶ **Lemma 1.** *1-player motion planning with any set of gadgets is in PSPACE.*

Proof. This was shown in [6], but included here for convenience. A configuration of the system of gadgets consists of the state of each gadget and the location of the robot, and has polynomial length. The algorithm that repeatedly nondeterministically picks a legal transition, and updates the configuration based on it, accepting when the robot reaches the goal location, decides the reachability problem in nondeterministic polynomial space. By Savitch's theorem, the problem is in PSPACE. ◀

▶ **Theorem 2.** *1-player motion planning with any k-tunnel gadget that does not have interacting tunnels is in NL.*

Proof. We first show that if a system of such gadgets has a solution, then it has a solution which visits each location at most once. Suppose there is a solution, and consider the last time a solution of minimal length visits a previously visited location, assuming there is any such time. Let v be the vertex of this last self-intersection. After leaving v for the last time, every transition the robot makes is through a tunnel that it had not previously traversed. Since the gadget does not have interacting tunnels, these tunnels have the same traversability when the robot goes through them as they do originally. We modify the solution by 'shortcutting': remove the portion of the solution between the first visit to v and the last visit to v, so the robot only visits v once, and skips the loop that begins and ends at v. The new path is still a solution: the segment before v is identical to the unmodified solution, and the segment after v consists of tunnels whose traversability is not changed before the robot goes through them. The shortcut path is shorter than the original solution, which was assumed to be minimal. Thus a solution of minimal length has no self-intersections.

We'll want to treat the system of gadgets as though it were a directed graph by replacing each tunnel with an edge in the appropriate direction, or a pair edges if it is traversable in either direction. We can locally walk through all the available transitions in a gadget, assess which locations they lead to, and non-deterministically pick one to try, allowing this to be executed in NL. A path from the start location to the end location in this graph is exactly a solution for the system of gadgets with no self-intersections; the traversability of each tunnel used in such a solution does not change before the tunnel is used.

Since reachability in directed graphs is in NL, the motion planning problem is also in NL. Moreover, if the gadget has any state in which a tunnel can be traversed in one direction but not the other, the motion planning problem is NL-complete, and otherwise it is in L. ◀

2.1 Reducing to Locking 2-Toggles

In this section, we introduce the locking 2-toggle shown in Figure 1a, and we show that all interacting-k-tunnel reversible deterministic gadgets can simulate it. The proof first examines what constraints on a gadget are implied by being interacting-k-tunnel, reversible, and deterministic, and goes on to identify that all such gadgets have a pair of special states with some useful common properties. From this pair of states we construct a 1-toggle, and then combine that with our special states to build a locking 2-toggle. One of the major insights is identifying this special pair of states which belongs to all gadgets in the class, and after that the primary challenge is in preventing undesired transitions, which are plentiful when allowing such a wide class of gadgets.

▶ **Theorem 3.** *Every interacting-k-tunnel reversible deterministic gadget simulates a locking 2-toggle.*

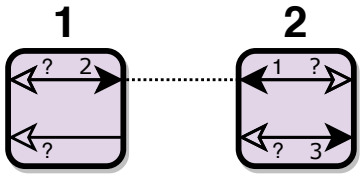
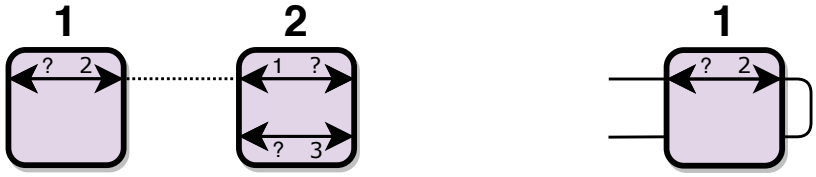


Figure 2 An arbitrary interacting-k-tunnel reversible deterministic gadget. Hollow arrows indicate traversals that may or may not be possible. Solid or absent arrows indicate traversals that are or are not possible, respectively.



(a) State graph, refining Figure 2. **(b)** Simulating a one-directional edge.

Figure 3 An arbitrary interacting-k-tunnel reversible deterministic gadget which has no one-directional edge.

Proof. We begin by examining an arbitrary interacting-k-tunnel reversible deterministic gadget, as shown in Figure 2. Because the gadget has interacting tunnels, we can find a pair of states in which traversing the top line can change the traversability of the bottom line to the right. Since it is also reversible, the inverse transition is also possible, so traversing the top line can change in either direction the left-to-right traversability of the bottom line. Then without loss of generality, the gadget has the form shown in Figure 2: in state 1, traversing the top line to the right switches to state 2, and the bottom line is not traversable to the right. In state 2, traversing the top line to the left switches to state 1, and the bottom line is traversable to the right, say to state 3 (which may be the same as state 1). All other traversals may or may not be possible in either state, indicated by the question marks.

▶ **Lemma 4.** *Every interacting-k-tunnel reversible deterministic gadget simulates a **one-directional edge**, that is, a tunnel which (in some state) can be traversed in one direction but not the other.*

Proof. If in some state, some edge in the gadget can be traversed in one direction but not the other, then it is a one-directional edge. Otherwise, the gadget has the form shown in Figure 3a. Then the construction in Figure 3b is equivalent to a one-directional edge: currently the gadget is in state 1, so the path from the bottom to the top is blocked by the bottom edge, but from the top, you can go across the top edge, switching the gadget to state 2, and then back across the bottom edge. ◀

▶ **Lemma 5.** *Every interacting-k-tunnel reversible deterministic gadget simulates a 1-toggle (Figure 1b).*

Proof. By the previous lemma, we can build a one-directional edge, which has the structure shown in Figure 4a: in state 1, we can traverse the edge to the right and switch to state 2, but not to the left. In state 2, we can undo this transition, and possibly also traverse the

(a) Form of state graph. **(b)** Simulating a 1-toggle.

Figure 4 A one-directional edge gadget.

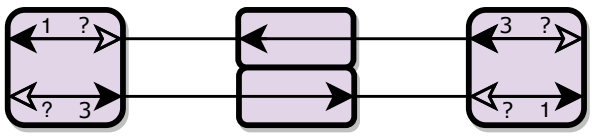
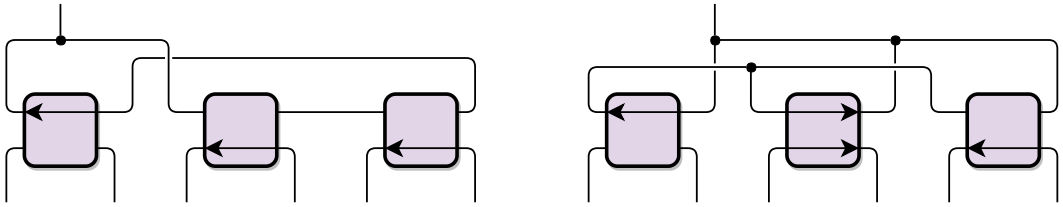


Figure 5 An arbitrary interacting-k-tunnel reversible deterministic gadget and a 1-toggle simulate a locking 2-toggle.

edge to the right. The construction in Figure 4b is then a 1-toggle. In the current state, it can be traversed to the right but not to the left because of the gadget on the left. After making this traversal, it becomes the rotation of the current state, and it cannot be traversed to the right again because of the gadget on the right. ◀

To build a locking 2-toggle, we put the arbitrary gadget (in state 2), an antiparallel pair of 1-toggles, and the rotation of the arbitrary gadget (also in state 2) in series, as in Figure 5. Currently, the top edge is traversable to the left and the bottom edge is traversable to the right, but not in the other direction. After traversing the top edge to the left, the 1-toggles prevents us from traversing either edge to the left, and the leftmost gadget (in state 1) prevents us from traversing the bottom edge to the right, so the only legal traversal is going back across the top edge to the right. Similarly after traversing the bottom edge, the only legal traversal is across the bottom edge in the opposite direction. Thus this construction is equivalent to a (antiparallel) locking 2-toggle.

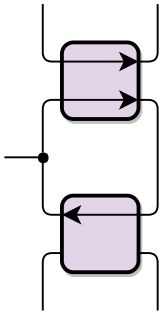
Traversing the simulated locking 2-toggle takes either 4 or 6 transitions of the raw gadget, depending on whether it contains a one-directional edge (from Lemma 4). For simplicity, we can include additional gadgets (e.g. another pair of 1-toggles) to ensure it always takes exactly 6 transitions; this will be relevant to timing considerations in multiplayer games. ◀



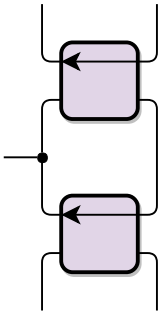
(a) An AND vertex gadget. The leftmost edge has weight two and is pointing in (up). The other edges have weight one and are pointing away (down).

(b) An OR vertex gadget. All edges are weight 2. The leftmost edge is pointing in (up), the middle edge is free, and the rightmost edge is pointing away (down).

Figure 6 Vertex gadgets in the NCL reduction.



(a) An edge gadget pointed up, in the unlocked state. The gadget is accessed by the loose end on the left.



(b) The same edge gadget in the locked state.

Figure 7 Edge gadget in the NCL reduction.

2.2 PSPACE-hardness

In this section, we show that 1-player motion planning with the locking 2-toggle is PSPACE-complete by a reduction from Nondeterministic Constraint Logic (NCL). See Appendix A.1 for a definition of NCL. We represent edges by pairs of locking 2-toggles. The construction requires **edge gadgets** which are directed and can be flipped, as well as AND and OR **vertex gadgets** which apply constraints on how many edges must be directed towards them at any given point in time.

▶ **Theorem 6.** *1-player motion planning with the locking 2-toggle is PSPACE-complete.*

Proof. Motion planning with the gadget is in PSPACE by Lemma 1. We use a reduction from Nondeterministic Constraint Logic (NCL) to show PSPACE-hardness. See Appendix A.1 for a definition of NCL.

The **edge gadget**, shown in Figure 7, contains two locking 2-toggles, each of which is also attached to a vertex gadget. It is oriented towards one of the vertices, can be either **locked** or **unlocked**. Specifically, the edge gadget is unlocked (Figure 7a) if either locking 2-toggle is in the middle state (with both lines traversable), and locked (Figure 7b) otherwise. It is oriented towards the vertex attached to the locking 2-toggle whose edge not accessible from the edge gadget is traversable. The robot can access the free line on the left. If the edge gadget is unlocked, the robot can traverse a loop through one edge of each locking 2-toggle to change the orientation of the edge gadget. The edge gadget switches between being locked and unlocked when the robot moves through a vertex gadget to traverse one of the edges not accessible from the edge gadget.

The **vertex gadgets** are shown in Figure 6. The robot can access the free line on the top, and traverse loops to lock and unlock edge gadgets, enforcing the constraints of vertices. Specifically, if all three edges are pointing towards an AND vertex, the robot can traverse a loop to lock both weight-1 edges and unlock the weight-2 edge, or vice versa. If multiple edges are pointing towards an OR vertex, the robot can traverse a loop to unlock the currently locked edge and lock another edge. Observe that for both vertex gadgets, the sum of the weights of locked edges does not change.

Given an NCL graph, we construct a maze of locking 2-toggles. Each edge in the graph corresponds to an edge gadget (Figure 7). Each locking 2-toggle in the edge gadget corresponds to a vertex incident to the edge. When three edges meet at a vertex, we put a vertex gadget on the locking 2-toggles corresponding to that vertex. We use an AND vertex

gadget (Figure 6a) or an OR vertex gadget (Figure 6b) depending on the type of vertex. The vertical "entrance" line on each vertex gadget and horizontal "entrance" line on each edge gadget is connected to the starting location. Each edge is oriented as in the NCL graph. For each vertex, we pick a set of edges initially pointing at the vertex with total weight 2. The edge gadgets corresponding to the chosen edges are locked, and other edge gadgets are unlocked. The goal location is placed inside the edge gadget corresponding to the target edge so that it is reachable if and only if the target edge is unlocked.

If the original NCL graph is solvable, the robot can perform the same sequence of edge flips, visiting vertex gadgets to lock and unlock edges as necessary, and reach the goal location. If the robot can reach the goal location, the same sequence of edge flips solves the NCL graph. So the maze is solvable if and only if the NCL graph was. ◀

This reduction is also possible without edge gadgets, and leads to a system with only one L2T for each constraint logic edge. We use edge gadgets because the reduction is easier to understand, and adaptations of this construction in Sections 2.3, 3, and 4 will need them.

▶ **Corollary 7.** *1-player motion planning with any interacting-k-tunnel reversible deterministic gadget is PSPACE-complete.*

Proof. Hardness follows from Theorems 3, and 6. For any such gadget, we have a reduction from mazes of locking 2-toggles to mazes of that gadget by replacing each locking 2-toggle with a simulation of one built from the arbitrary gadget. Motion planning with the gadget is in PSPACE by Lemma 1. ◀

2.3 Planarity

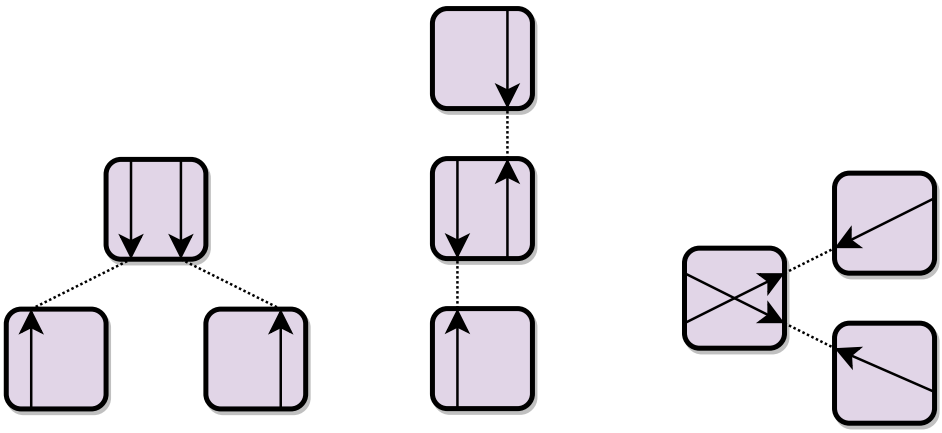
In this section, we show that interacting-k-tunnel reversible deterministic gadgets are PSPACE-complete even for the planar 1-player motion planning problem. We once again work with the locking 2-toggle, showing that each of its planar versions can simulate each other. From there we use the crossing locking 2-toggle to build an A / BA crossover, which is less powerful than a full crossover but will suffice to make our reduction in Section 2.2 planar. An interesting question is whether the locking 2-toggle is powerful enough to build a full crossover, which can be done with any of the 2 state gadgets. Although not needed here, it would allow the multiplayer game results later in this paper to carry over to the planar case.

Recall for the planar problem we allow rotations and reflections of gadgets. This leaves three distinct embeddings of the locking 2-toggle into a plane: parallel, antiparallel, and crossing, shown in Figure 8, and which we abbreviate PL2T, APL2T, and CL2T. (Up to only rotation, there are four, the other being the antiparallel locking 2-toggle with the other handedness). We will allow reflections of gadgets, so these are the three kinds of locking 2-toggles we will consider.

▶ **Lemma 8** ([6])**.** *Parallel, antiparallel, and crossing locking 2-toggles all simulate each other in planar graphs.*

Proof. Figure 9 shows APL2T simulating CL2T, Figure 10 shows CL2T simulating PL2T, and Figure 11 shows PL2T simulating APL2T. Note that we use both APL2Ts of both handednesses, so we need to be able to reflect gadgets. ◀

▶ **Theorem 9.** *Every interacting-k-tunnel reversible deterministic gadget simulates each type of locking 2-toggle in planar graphs.*



(a) A parallel locking 2-toggle (PL2T). **(b)** An antiparallel locking 2-toggle (APL2T). **(c)** A crossing locking 2-toggle (CL2T).

Figure 8 Types of locking 2-toggles in planar mazes.

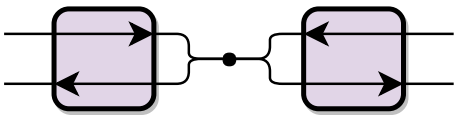


Figure 9 APL2T simulating CL2T. (Based on [6, Figure 4].)

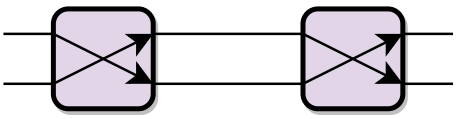


Figure 10 CL2T simulating PL2T. (Based on [6, Figure 5].)

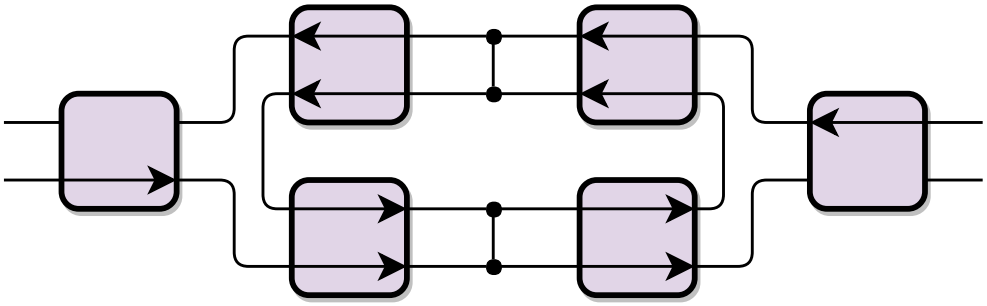


Figure 11 PL2T simulating APL2T. (Based on [6, Figure 13].)

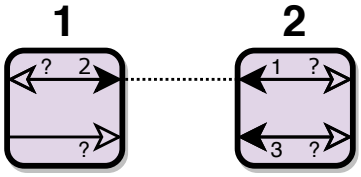


Figure 12 The antiparallel case of an arbitrary interacting-k-tunnel reversible deterministic gadget.

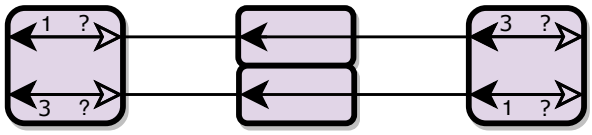


Figure 13 An arbitrary antiparallel interacting-k-tunnel reversible deterministic gadget and a 1-toggle simulate a PL2T.

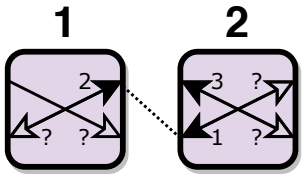


Figure 14 The crossing case of an arbitrary interacting-k-tunnel reversible deterministic gadget.

Proof. We follow the proof of Theorem 3. As before, we assume that traversing a line to switch from state 1 to state 2 makes a traversal on another line legal. This new traversal can be parallel to, antiparallel to, or cross the first traversal; we consider each case. If the new traversal is parallel, the construction in the proof of Theorem 3 works to simulate an APL2T in a planar graph.

If it is antiparallel, the gadget has the form shown in Figure 12. Either the gadget has a one-directional edge, or it has the form in Figure 3a, and simulates a one-directional edge by the construction in Figure 3b. Thus it simulates a 1-toggle by the construction in Figure 4b. Then the construction in Figure 13 simulates a PL2T: currently either edge can be traversed to the left, if the top edge is traversed, the left gadget blocks the bottom edge, and if the bottom edge is traversed, the right gadget blocks the top edge.

Finally, if the new traversal crosses the first traversal, the gadget has the form shown in Figure 14. Either it has a one-directional edge, or the construction in Figure 15 simulates a one-directional edge, similarly to Lemma 4. So the gadget simulates a 1-toggle by the construction in Figure 4b. Then the construction in Figure 16 simulates a PL2T, similarly to the previous case.

Once the gadget simulates some locking 2-toggle, we can use Lemma 8 to simulate all three types. ◀

▶ **Theorem 10.** *1-player planar motion planning with any interacting-k-tunnel reversible deterministic gadget is PSPACE-complete.*

Proof. We begin by constructing some weak crossover gadgets. The crossover locking 2-toggle is itself a very weak crossover. We use it to construct an **A/BA crossover**, shown in Figure 17a. Calling the traversal from top to bottom A and that from left to right B, we can

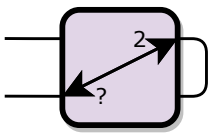


Figure 15 A crossing interacting-k-tunnel reversible deterministic gadget simulates a one-way edge.

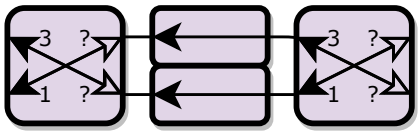


Figure 16 An arbitrary crossing interacting-k-tunnel reversible deterministic gadget and a one-toggle simulate a PL2T.

perform either of the sequences A and BA. Since everything is reversible and deterministic, we can also undo those sequences. The A/BA crossover is sufficient for the rest of the proof; we abbreviate it as shown in Figure 17b.

We modify the proof of Theorem 6, giving a reduction from planar NCL to planar mazes with locking 2-toggles. By Theorem 9, this is sufficient to show PSPACE-hardness. Our gadgets use PL2Ts, CL2Ts, and A/BA crossovers; they do not use APL2Ts.

The edge gadget is shown in Figure 18, and vertex gadgets are shown in Figure 19. Given a planar NCL graph, we construct a mazes as follows.

Pick a rooted spanning tree of the dual of the NCL graph, directed away from the root; the robot will use this tree to navigate the graph. The system of gadgets will contain a vertex for each face f of the NCL graph, which is a vertex of the spanning tree.

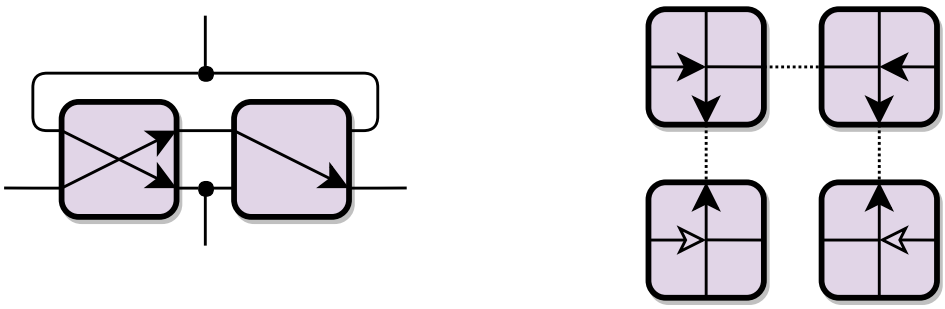
For each edge of the graph, we place an edge gadget. When an edge is in the spanning tree, we orient it so that the A/BA crossover points, from entrance to exit, in the same direction as the edge points in the spanning tree (left to right in Figure 18, and away from the root). If an edge is in the spanning tree and has target f, we connect its exit to f. For each edge e, we connect its entrance to the vertex f corresponding to the face containing its entrance, i.e. the face adjacent to e to which we can connect its entrance without crossings. If e is in the spanning tree, this connects the entrance of e to the source f of e.

Now we place a vertex gadget of the appropriate type for each vertex of the NCL graph, so that the gadget shares a PL2T with each incident edge gadget. AND vertex gadgets must be oriented so the weight-2 edge has the appropriate PL2T (the bottom one in Figure 19a). The entrance of each vertex gadget is connected to the vertex f corresponding to the face containing the entrance.

We set each edge gadget to the orientation of its corresponding edge. For each vertex, we select edges directed towards it with total weight 2, and set the selected edges to locked and other edges to unlocked. The goal location is placed inside the target edge so that reaching it requires flipping the target edge. The starting location is the vertex corresponding to the root of the spanning tree.

Play on this maze proceeds as follows: the robot travels down the spanning tree, crossing edges until it reaches some face. It goes into an edge or vertex attached to that face, and manipulates it. Then the robot travels back up the spanning tree and down a different branch, manipulating another edge or vertex, and so on. The edge and vertex gadgets enforce the NCL constraints. If the target edge can be flipped, the robot can reach the goal location. Thus the maze is solvable if and only if the NCL graph was. The maze is planar by its construction, using the planarity of the NCL graph.

This completes the proof of PSPACE-hardness. Containment in PSPACE is by Lemma 1, so the problem is PSPACE-complete. ◀



(a) Simulating an A/BA crossover using CL2Ts.

(b) A state diagram and notation for the A/BA crossover.

Figure 17 An A/BA crossover gadget: the robot can traverse top to bottom (A), or traverse left to right (B) and then top to bottom. Thinking of the gadget as a crossing pair of 1-toggles, the vertical 1-toggle is always traversable, and the horizontal 1-toggle is traversable when the vertical one is pointing down.

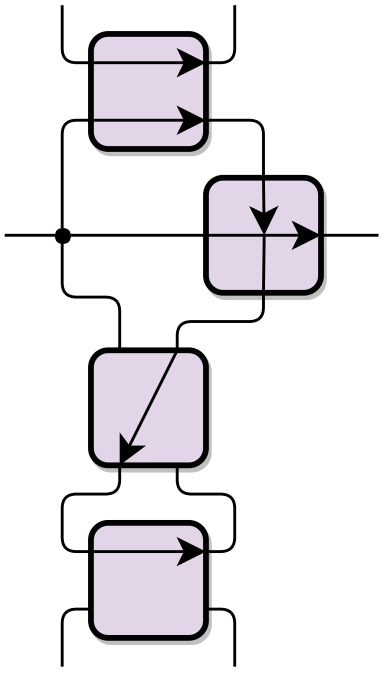
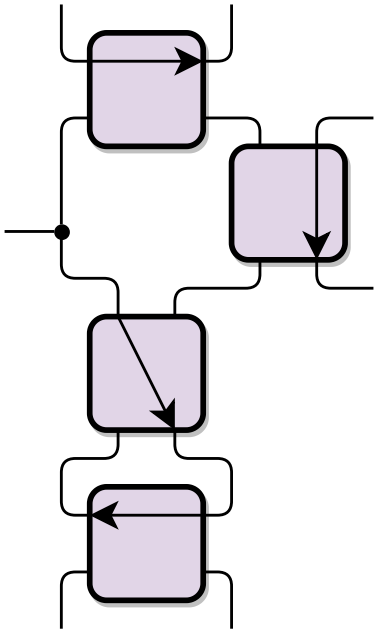
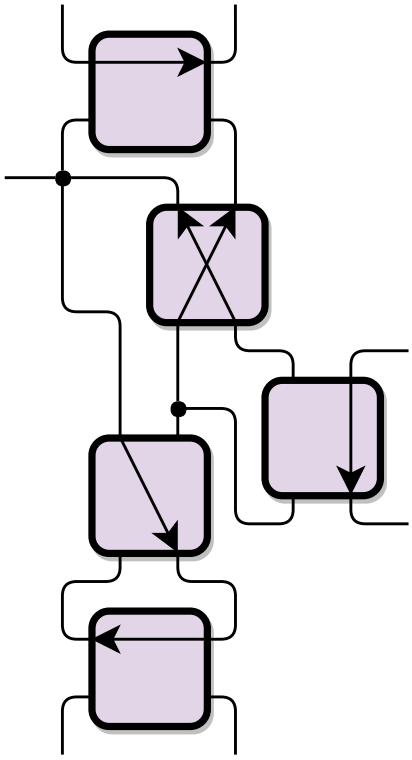


Figure 18 An edge gadget for planar graphs, currently unlocked and directed up. This is analogous to Figure 7, with two changes. First, the bottom PL2T is "twisted" to have the same handedness as the top PL2T for connecting to vertex gadgets; the CL2T is sufficient for the crossing caused by this. Second, the A/BA crossover allows the robot to cross the edge from left to right, regardless of the state of the edge. We call the line on the left the **entrance** and the line on the right, on the other side of the A/BA crossover, the **exit**.



(a) An AND vertex for planar graphs. Currently the weight-2 edge, connected at the bottom PL2T, is directed towards the vertex and locked, and both weight-1 edges are directed away. If the weight-1 edges become directed towards the vertex, the robot can visit the vertex gadget and traverse a loop through all three PL2Ts, locking the weight-1 edges and unlocking the weight-2 edge. The CL2T is a sufficient crossover.



(b) An OR vertex for planar graphs. Currently the edge containing the bottom PL2T is directed towards the vertex, and the other edges are directed away. If multiple edges are ever directed towards the vertex, the robot can visit the vertex gadget, unlock the locked edge, and lock another edge.

Figure 19 NCL vertex gadgets for planar graphs, analogous to the gadgets in Figure 6. In each gadget, each of the three PL2Ts is also part of an edge gadget. The robot enters at the line on the left, called the **entrance**, traverses loops that enforce the NCL constraints, and then leaves at the entrance.

3 2-Player Unbounded Motion Planning

In this section, we analyze 2-player motion planning games with k-tunnel reversible deterministic gadgets. We show that any such game which includes an interacting-tunnels gadget is EXPTIME-complete. We do so by a reduction from 2-player unbounded constraint logic, allowing us to reuse some of the work in the prior section. In addition to building the single player AND and OR vertices, we show how to adapt the gadgets to allow different players to have control of different edges. We also build up the needed infrastructure to enforce turn taking in the simulated game.

The construction of crossovers using interacting-k-tunnel reversible deterministic gadgets with two states should allow one to show hardness for the planar version of this problem with those gadgets and any others that simulate them. Care must be taken with the layout, timing, and interaction between crossovers so we do not go on to prove such a result in this paper. Unfortunately, the crossover created by the locking 2-toggle in Section 2.3 does not suffice and thus leaves the question partially open. In addition, the question of noninteracting-k-tunnels reversible deterministic gadgets has not been resolved. We are not able to show problems with such gadgets are easy, and Section 6 suggests they should be at least PSPACE-hard.

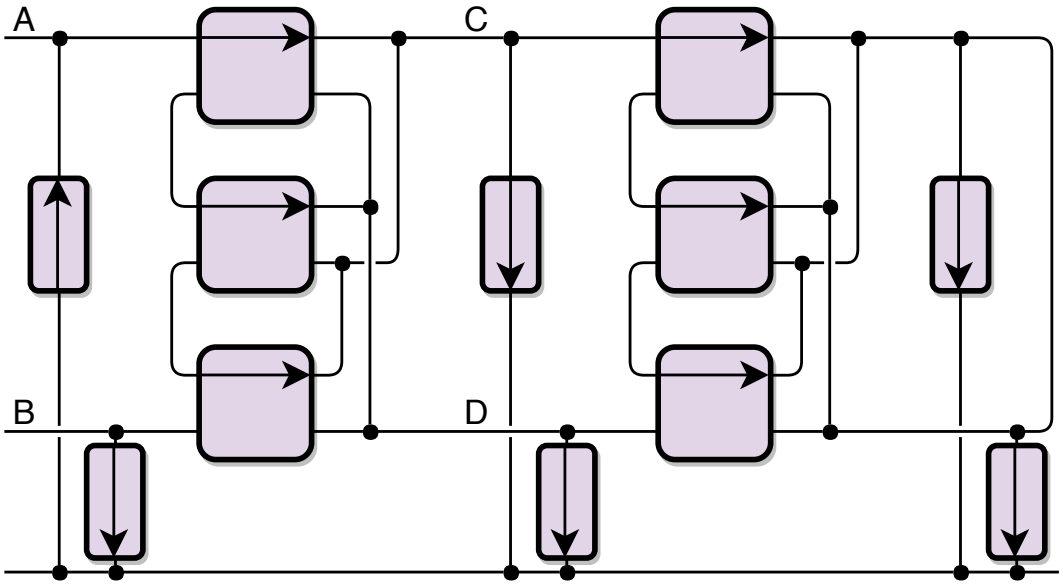


Figure 20 The timer gadget used in the 2CL reduction, made of PL2Ts and 1-toggles. In order to travel between A and B, a player must travel between C and D three times. The timer can be extended to the right; two iterations are shown.

▶ **Lemma 11.** *2-player motion planning with any set of gadgets is in EXPTIME.*

Proof. A configuration of the maze consists of the state of each gadget and the location of the robot, and has polynomial length. There is a polynomial-space alternating Turing machine which nondeterministically guesses moves for each player and keeps track of the configuration, using existential quantifiers for player 1 and universal quantifiers for player 2. This Turing machine accepts exactly when player 1 has a forced win. Thus the problem is in APSPACE = EXPTIME. ◀

▶ **Theorem 12.** *2-player motion planning with the locking 2-toggle gadget is EXPTIME-complete.*

Proof. This game is in EXPTIME by Lemma 11. We use a reduction from 2-player Constraint Logic (2CL) to show EXPTIME-completeness. See Appendix A.1 for a definition of 2CL.

We begin by describing a timer gadget, shown in Figure 20. Suppose one player has access to the bottom line. They can enter the gadget at A, and begin going through the timer, eventually reaching a victory gadget at B. The timer has two key properties:

1. Reaching B takes a number of transitions exponential in the size of the timer. In order to get from A to B, the player goes though the top PL2T to C, recursively travels from C to D, goes around the loop through the top two PL2Ts, goes back from D to C, traverses the bottom loop, once again goes from C to D, and finally proceeds to B. If traveling between C and D takes m transitions, then traveling between A and B takes $3m + 6$ transitions. If the timer gadget is repeated k times, it takes at least 3^k transition to get from A to B.
2. A player in the timer has an opportunity to exit the timer at least every 2 turns, and exiting takes 1 turn; in particular, they can always exit within 3 turns while progressing the timer. The player uses a 1-toggle to exit to the bottom line. They can then later reenter using the same 1-toggle, resuming their work on the timer where they left off. If

the player is in the timer, the next step in progressing the timer is either traversing a loop between to PL2Ts, which takes 2 transitions, or moving horizontally between timer segments, which takes 1 transition. Thus in 3 transition, the player can complete the current or next step and exit to the bottom line.

The constraint logic gadgets are similar to those used in Theorem 6 for the 1-player game, with the modification shown in Figure 21. We have added 1-toggles allowing a player at an edge to visit and configure the incident vertices, without allowing the player to travel to other edges. Each player's goal location is inside the gadget corresponding to their target edge, so that they can reach it if they can flip the edge.

Unlike the 1-player version, we need gadgets to enforce the turn order. The overall construction is shown in Figure 22. The maze consists of three main regions: the White area, the Black area, and the constraint logic. Each player will spend most of their time in their own area, occasionally entering the constraint logic to flip an edge. The players' areas are designed to enforce turn order and progression of the game. A player can never enter the other player's area.

There is a single L2T separating the constraint logic area from each player's area. This prevents both players from being in the constraint logic at the same time.

Each player's area contains an edge selection gadget, which consist of a locking 2-toggle for each edge they can control. The other line in the L2T is accessible by entering the constraint logic area and passing through a delay line four 1-toggles, and is connected to the corresponding edge gadget. In order to access an edge gadget, the player must activate the appropriate L2T, which requires deactivating the previously activated L2T. This ensures that only one edge gadget is accessible by each player at any time. There is a 1-toggle separating the edge selection gadget from the rest of the player's area, so that switching the selected edge requires at least 4 turns (we use one tunnel of a L2T for a 1-toggle).

Each player's area has a timer, of length t_w for White and t_b for black. If a player finishes their timer, they win.

Each player begins inside their edge selection gadget, and White goes first. The game begins with White picking an edge and going to the constraint logic area, while Black goes to their timer.

A round of normal play proceeds as follows:

- White moves from edge selection to the constraint logic area. Black is currently in their timer.
- White enters the constraint logic, walks to their selected edge, and flips it. Black continues working on their timer.
- White returns through their constraint logic delay line. Once they pass the first 1-toggle, Black finishes their current step in the timer and exits, moving towards edge selection.
- White begins working on their timer. Black selects an edge, enters the constraint logic, and flips the edge.
- Black returns through their constraint logic delay line. Once they pass the first 1-toggle, White exits their timer and moves to edge selection.
- White selects an edge as Black enters their timer.

Suppose Black has just flipped an edge gadget; they have nothing to do but return through the delay line of length 4. When Black is past the first 1-toggle, White will leave their timer to flip an edge. Black might try turning around to go back to the constraint logic area. It takes Black at least 6 turns to flip the edge back, during which White has enough time to select an edge and reenter their timer. The game is now in the same situation as before, except that White has progressed their timer; thus Black does not want to do this.

Black might instead try waiting at the central L2T after White has selected an edge. White will then go to their timer, forcing Black to exit eventually. When Black is not next to the central L2T, White exits their timer and moves to constraint logic. Because of the 1-toggle separating edge selection from the central L2T, for Black to change their selected edge, they must spend multiple turns away from the L2T, allowing White to enter constraint logic; similarly if Black works on their timer, White can enter constraint logic. So Black has no choice but to pass the turn to White.

Since White can always exit their timer within 3 turns, and Black has three more 1-toggles to get through when White begins looking to exit, White will reach edge selection before Black can reach edge selection, so White will be the first player ready to enter constraint logic again. Nothing Black can do will prevent White from taking the next turn in the 2CL game. Similarly after White flips an edge, Black will be able to take a turn next. So either player can force the alternation of constraint logic turns.

The sizes of the timers are chosen to satisfy the following. First, if White cannot win the constraint logic, Black should win, so Black's timer is shorter: $t_b < t_w$. Second, if White can win the constraint logic game, White should win first, even if Black ignores the constraint logic game and just works on their timer. If the constraint logic graph has n edges, it takes at most 2^n constraint logic turns for White to win. Each constraint logic turn for White takes 6 turns to select an edge and return to the constraint logic, 8 turns to cross the constraint logic delay line twice, 4 turns to access and flip an edge, and up to 5 turns to access and configure an incident vertex, so 25 turns in total during which Black can work on their timer. Both players might be in their timers simultaneously at most 4 times each cycle, and each time for at most 4 turns, so Black spends at most 41 turns in their timer for each constraint logic turn. Thus, since it takes Black at least 3^{t_b} turns to win through the timer, we need $41 \cdot 2^n < 3^{t_b}$; $t_b = n + 6$ suffices, and we can set $t_w = 2n + 12$.

Using these timer sizes, it is clear that the constraint logic game will resolve before either timer if the players follow normal play. We need the timers so that Black cannot force a draw by sitting in the constraint logic forever, preventing White from winning; White will progress on their timer if Black attempts this.

Hence White has a forced win in the motion planning game if and only if they have a forced win in the constraint logic game. Since 2CL is EXPTIME-complete, the 2-player game on systems of locking 2-toggles is EXPTIME-hard. The maze used in the reduction has only $O(n)$ L2Ts. ◀

▶ **Theorem 13.** *2-player motion planning with any interacting-k-tunnel reversible deterministic gadget is EXPTIME-complete.*

Proof. This game is in EXPTIME by Lemma 11. We adapt the 2CL reduction in the proof of Theorem 12. Replace each locking 2-toggle in that 2CL reduction with the simulation of a locking 2-toggle from the arbitrary gadget in Theorem 3. In the new maze, each tunnel in a simulated L2T takes 6 transitions to traverse, so the game goes 6 times slower.

The simulation still works with two players, as long as both players do not have access to the gadget at the same time. Each L2T in the turn enforcement area is accessible only by one player, and only one player can be in the constraint logic area at any time. The only L2T both players have simultaneous access to is the central gadget which gives access to the constraint logic area, so we look more carefully at that gadget.

The state with both edges traversable is shown in Figure 5 (the 1-toggle simulation still works). Note that the simulation is of an APL2T, but the gadget in the 2CL reduction is a PL2T; this is not a problem because we are not concerned with planarity. Suppose both

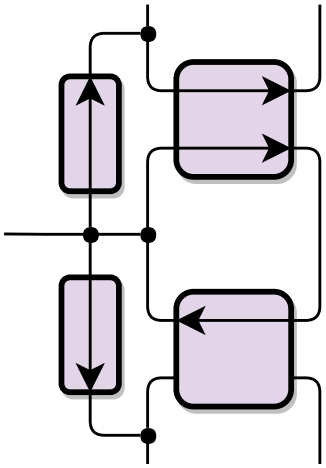


Figure 21 A modified edge gadget for the 2CL reduction. A player can visit the vertex gadgets attached to the edge gadget, and then return to the edge gadget.

players approach the gadget, one from the right on the top line and one from the left on the bottom line. Whoever reaches the gadget first should "win the race", and lock out the other player. The simulation implements this correctly, provided that the player who arrives first is a full turn ahead in the L2T maze, or 6 turns ahead in the new maze. The only time the players might be within 6 turns of each other is at the very beginning of the game, so we put a delay of 6 turns for Black to get from their start location to edge selection to ensure White wins the race by 6 turns. If a player would arrive less than 6 turn before the other player, they should go to their timer instead; since this is a zero-sum game and the players would have to collaborate to break the simulation, one player will choose not to.

The other way players can interact at this gadget is when one player is exiting the constraint logic area, and the other player is waiting just outside and enters as soon as they can. The state of the simulation is shown in Figure 23 (the other possible state is symmetric). One player, say White, has traversed the top edge to enter the constraint logic area, and is about to exit by traversing the top line to the right. Black is waiting at the left end of the bottom line, ready to enter the constraint logic area. The leftmost gadget prevents Black from making any transitions until White begins exiting. Once White begins exiting, the leftmost gadget switches to state 2, so Black can follow parallel to White and one turn behind. As long as White continues through the construction at full speed, Black interacts with the construction as though White has already finished their traversal, so it correctly simulates a L2T. Again breaking the simulation would require the players to cooperate, and the game is zero-sum, so at least one player will ensure the simulation works. ◀

4 Team Unbounded

In this section, we show that team imperfect information games with interacting-k-tunnel reversible deterministic gadgets is RE-complete, implying the problem is Undecidable. The reduction is from Team Private Constraint Logic (TPCL); see Appendix A.1 for a definition. We use many of the ideas and constructions from Section 3, but various modifications are needed to deal with the additional player and the model of player knowledge. Recall in this model we have three players on two different teams, each controlling a single robot.

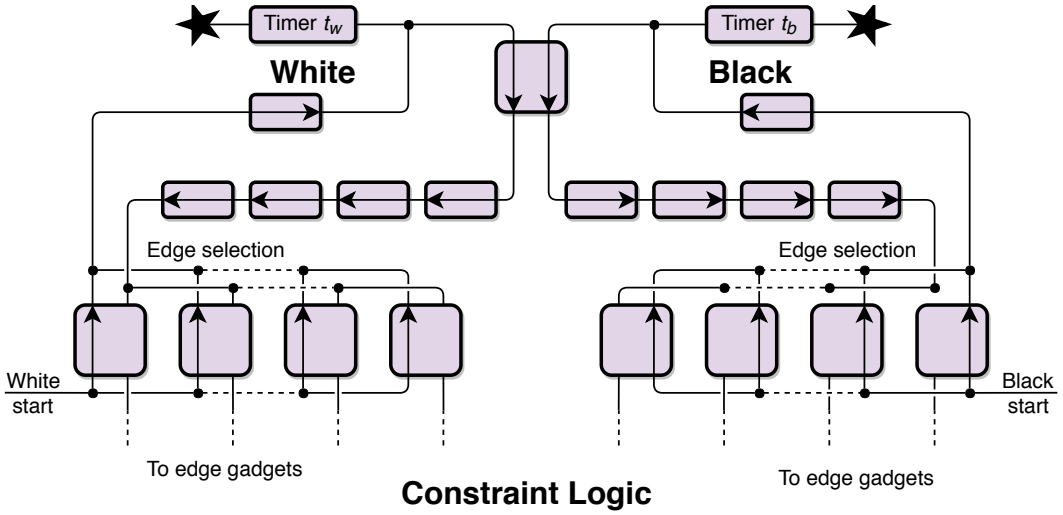


Figure 22 The overall structure and turn enforcement gadget. Each player's edge selection area has a L2T for each edge that player can flip; four are shown for each player. The bottom line from each such L2T connects to the corresponding edge gadget. The timers are as shown in Figure 20, with t_w and t_b repetitions. The inside connection to each timer is connected to its access line, and the outside connection (to a win gadget) is at B in Figure 20. The goal location past each timer is for the player whose side it is on.

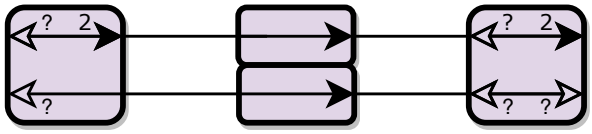


Figure 23 Another state of the construction shown in Figure 5. The leftmost gadget is in state 1, and the rightmost gadget is in state 3.

All players start knowing the configuration of the entire game; however, after that point players can only observe the states of the gadgets that their robots can reach via the connection graph. Adaptations for the planar version and the complexity of such games with noninteracting-tunnel gadgets remains open as in Section 3.

▶ **Lemma 14.** *Team motion planning with any set of gadgets is in RE (recursively enumerable).*

Proof. Suppose the White team has a forced win on some system of gadgets, and consider the tree of possible positions when White follows their winning strategy. The branches in the tree correspond to choices the Black team might make. Since White forces a win, every branch of the tree is finite. Since Black has finitely many choices at each turn, the tree is finitely branching, so by Kőnig's infinity lemma [14], the tree is finite. In particular, there is a finite bound on the number of turns it takes for White to win, so the winning strategy can be described in a finite amount of space. So there are countably many potential winning strategies, and we can sort them lexicographically.

Given a potential winning strategy, the problem of determining whether it is actually a winning strategy is decidable: an algorithm can explore every choice Black might make, and see whether White always wins. There are only finitely many choices to check because the strategy only describes a finite number of turns.

We use the following algorithm to determine whether White has a forced win. For each potential winning strategy in lexicographic order, check whether it is a winning strategy. If it is, accept. This algorithm accepts whenever White has a forced win, and runs forever otherwise, so it recognizes the games in which White has a forced win. ◀

Although [7] only mentions undecidability and not RE-completeness, it follows that TPCL is RE-complete. Containment in RE is given by an argument nearly identical to the proof of Lemma 14. The proof of undecidability is ultimately by a reduction from acceptance of a Turing machine on an empty input, which is RE-complete, implying that TPCL is RE-hard.

▶ **Theorem 15.** *Team motion planning with the locking 2-toggle gadget is RE-complete (and thus undecidable).*

Proof. Containment in RE is given by Lemma 14. For RE-hardness, we use a reduction from TPCL, with a similar construction as in the proof of Theorem 12. The overall construction is shown in Figure 24. Capital letters label L2Ts, and lowercase letters label lengths of delay lines. The two tunnels in the same L2T are labelled the same, instead of being positioned next to each other. The three players B, W_1, and W_2 each have their own region. Each region contains an edge selection area with k edges initially active, access to the constraint logic, and some additional gadgets. We need to ensure the following:

1. Turn order is enforced. That is, the players take turns in the order B, W_1, W_2, and neither team can gain anything by deviating from this. We use L_1 and L_2 to prevent B from being in the constraint logic area at the same time as W_1 or W_2, and appropriate delays to ensure each player is ready for their turn. The timer in W_2's region forces B to eventually pass the turn to W_1.
2. Each player can flip up to k edges each turn. If k edges are initially accessible for each player, the edge selection area allows them to select any k of their edges, and a player must end their turn in order to change their selection.
3. The White players have the correct information about the state of the game. Each of them has a visibility area, which allows them to see the orientation of the appropriate constraint logic edges. We must not allow W_1 and W_2 to both access the same L2T, as they could then use it to communicate. So we need a more complicated mechanism to prevent both White players from being to the constraint logic area at the same time.

For visibility, we modify the edge gadget as shown in Figure 25. The appropriate line is connected to each White player's visibility area if they should be able to see that edge.

A round of normal play proceeds as follows:

- B begins their turn by passing down through L_1 and L_2. W_1 waits next to V, and W_2 walks through their timer.
- B flips some edges, and returns, passing V. When W_1 sees this happen, they go to their visibility area, and then select k edges. W_2 continues in the timer.
- B finishes exiting through the delay b. Once B has passed L_1, W_1 enters the constraint logic area. W_2 reaches the end of the timer, finds S to be closed, and comes back.
- B is stuck on the side of L_1 away from the constraint logic area, and can select edges. W_1 flips edges and returns to just below L_1. W_2 goes to their visibility area, and then selects edges.
- After a number of turns large enough that both White players are definitely ready, W_1 exits L_1. The same round, W_2 enters L_2, passing the turns from W_1 to W_2.
- W_2 takes their turn. B waits just to the right of L_2, and W_1 waits above X.
- W_2 exits L_2 and goes to the timer. B passes through L_2 to take their turn, and W_1 waits.

We place each player's starting location to be at the end of a chain of 1-toggles leading to their region, so they arrive after an appropriate delay. We can set B to have no delay and W_1 and W_2 to have $2k$ delay, so B has time to select edges before the White players arrive. The first turn has slightly strange timing since W_2 starts the timer later than normal, but this is not important.

We consider ways in which player might deviate from normal play, and see that in each case they do not gain anything by deviating.

B enters the constraint logic through L_2 as soon as W_2 passes L_2 on their way out, at which point W_2 enters the timer. B need to be able to take a full turn and go back through S before W_2 reaches the end of the timer; this takes up to $2(b+c+2)+2+11k$ turns, since flipping each edge now takes up to 11 turns. So we need $t > 2(b+c+2)+2+11k$. The timer forces B to return through S within $t+2$ turns, since otherwise W_2 wins.

The gadget V lets W_1 know when B is done, since W_1 can see whether B is past V while waiting at L_1. Specifically, W_1 waits until they see B stay past V for $2c$ turns, and then return. For B to be unable to flip edges after this, we need $4c > t$. Then W_1 goes to visibility and sees the current configuration, selects k edges for their next turn, and waits at L_1 again. For W_1 to have time to do this before B gets out, we need $b > 2k+2$.

Once B exits L_1, W_1 goes in and flips edges. The delay d ensures that if W_1 (or W_2) flips any edges, then B will be ready for their next turn; we need $2d > 2k+4$. W_2 returns through the timer, checks visibility, and selects edges. If W_2 enters constraint logic before W_1 leaves, B can win through X and Y, so W_2 must wait until W_1 leaves. The White players coordinate using the fact that the length of an entire round is bounded, so they can wait long enough to ensure that they are both ready, and then W_1 exits X immediately before W_2 enters Y. Since W_1 was past L_1, B is locked outside of L_1, so W_2 can get past L_2; the W_1 can safely pass the turn to W_2.

While W_1 is past X, B might try going through Z and X, trapping W_1. In this case, W_2 can win through Z, so B will only go through Z if both X and Y are traversable.

During W_2's turn in the constraint logic, W_1 must not be past X to prevent B from winning through X and Y. So B can go through L_1, and go through L_2 as soon as W_2 exits. That is, W_2 cannot pass the turn back to W_1.

W_2 might try to stay in the timer, forcing B to stay out of the constraint logic to prevent W_2 from winning through S. Then W_1 might be able to take extra turns in the constraint logic. If the White team attempts this, B will win through P and Q. If B goes through R and P when Q is not traversable in order to trap W_1, W_2 will win through R; these three L2Ts are analogous to X, Y, and Z.

Assuming the constraints mentioned are satisfied, no player or team can usefully deviate from normal play, and normal play simulates the TPCL game. Thus White has a forced win in the team motion planning game if and only if they have a forced win in the TPCL game.

We can satisfy all the constraints, e.g by $b = 2k+3$, $c = 8k+7$, $d = k+3$, and $t = 31k+27$ (the constraints are not tight, but they suffice). The number of L2Ts in the resulting system of gadgets is only linear in the number of edges in the constraint logic graph. ◀

▶ **Theorem 16.** *Team motion planning with any interacting-k-tunnel reversible deterministic gadget is RE-complete.*

Proof. Containment in RE is given by Lemma 14. For RE-hardness, we adapt the TPCL reduction in Theorem 15 to work for the arbitrary gadget. As in the 2-player case of Theorem 3, it is almost sufficient to replace each L2T with the simulation in Theorem 3. We examine the L2Ts that are shared between two players.

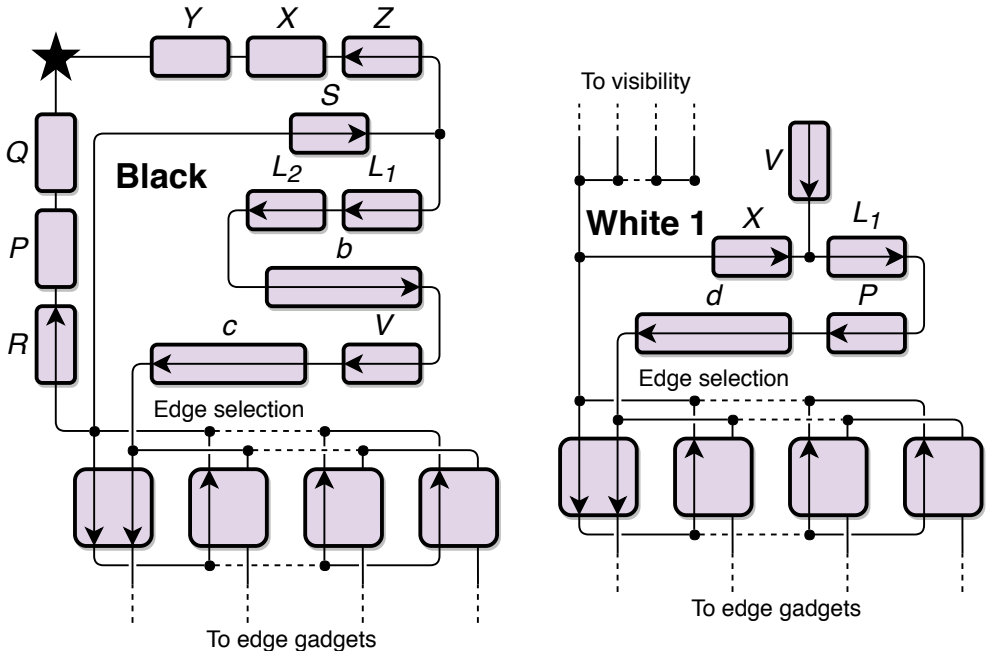


Figure 24 The turn enforcement gadget for the team game. Each player has their own region which contains an edge selection area, a path to the edge gadgets they can control, and some other constructions. Each White player has a visibility area which allows them to see the state of some edge gadgets in constant time. There is no good layout for the whole gadget, so we use pairs of 1-toggles that share a (capital) label to represent L2T. Long boxes with lowercase labels represent chains of 1-toggles with length given by the label. The win gadgets are for the obvious players, and the tunnels currently not traversable (P, Q, R, S, X, Y, and Z) will directed toward the win gadget when they become traversable.

First, L_1 and L_2 are analogous to the central L2T in Theorem 3: if two player are racing to enter, the player who should win is at least 6 turns ahead, and if one player exits and another enters, is works correctly.

For S, P, Q, R, X, Y, and Z, we use a single copy of the arbitrary gadget with 5 extra gadgets for delay, instead of the simulation. Considering the gadget as in Figure 2, we use state 1, and put the bottom edge in the position next to a win gadget. For S, Q, Y, R, and Z, if the bottom edge is traversed from state 2, the game is over, so the gadget is never in a state other than 1 or 2 while the game is going. For P and X, we know that B cannot safely wait past those gadgets, so the game must be about to end in Black victory if they ever reach state 3.

For V and the visibility gadgets on edges, we use the construction in Figure 26. B has three paths to choose from in the process of crossing the bottommost 1-toggle, and always two of them are align with that 1-toggle, so B has two options. The White player, say W_1 can see the state of a gadget in all three paths, and thus determine the orientation. If W_1 goes through one of these gadgets, B will use the other path. If there were only one path, W_1 could go through the gadget, forcing B to either not flip that edge or get a gadget into an unknown state (for L2Ts, we used the fact that W_1 could never traverse that tunnel in one direction). This visibility gadget allows W_1 to see the orientation of a constraint logic edge or V without being able to interfere.

Once we make these replacements, the new maze with the arbitrary gadget has a forced win by White if and only if the maze with L2Ts did. ◀

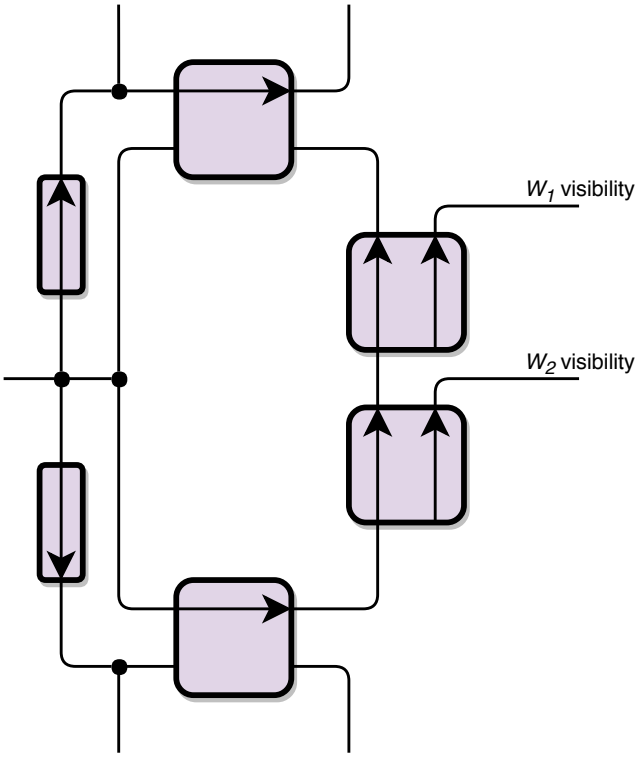


Figure 25 An edge gadget for the TPCL reduction. This is the same as a 2CL edge gadget, except two L2Ts have been added that allow W_1 or W_2 to see the state of the edge if it is connected to their visibility area, but they cannot make any transitions.

5 1-Player Bounded Motion Planning

In this section, we consider a broad class of gadgets which are naturally in NP and give a dichotomy classifying them as NP-complete or in NL. We examine all gadgets in tunnels whose state-transition graph forms a DAG. We will call these DAG gadgets for short. Our proof of hardness further applies to a larger class of gadgets, however a full classification of more general, simple to describe classes of gadgets will require more insight or much more case-work. Also, our constructions require the use of a crossover gadget.

The results in this section can be seen as similar to Viglietta's Metatheorem 1 about location traversal (being implemented by the interacting tunnels in gadgets) and single-use paths [19]. It also bears resemblance to Metatheorem 4 about pressure plates which only affect one door [19]. However, our proof goes through 3SAT rather than Hamiltonian Path, uses a different underlying model which makes different features salient, and gives generalizations in a different direction. Structurally the proof follows that used to show Mario as well as many other games are NP-hard [1].

▶ **Lemma 17.** *All DAG gadgets contain a single-use transition unless they are a transitionless gadget.*

Proof. First, find a node which only has transitions to **terminal states**, ones with no possible further transitions. To find one, begin by removing all terminal states from the graph. Of the remaining nodes, all of them which are now terminal states must have pointed to at least one terminal state in the original graph or it would have been removed, and it

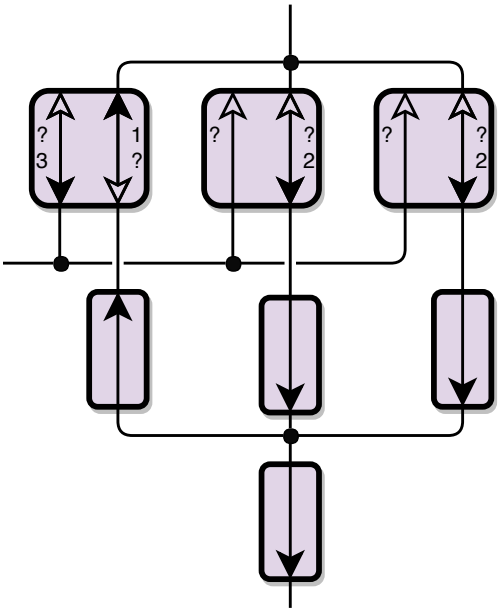


Figure 26 A visibility gadget for the TPCL reduction. The Black player can travel between the top and bottom, and a White player can enter the side to see which direction was traversed most recently.

must have only pointed to terminal states or it would not be a terminal node. Terminal nodes have no transitions. Thus the node we discovered has an available transition which closes all tunnels in the new state. The gadget starting from that state is a single-use gadget. ◀

In a system of gadgets, each DAG gadget can only be traversed polynomially many times. This is the core reason that motion planning involving these gadgets is always in NP.

▶ **Lemma 18.** *1-player motion planning with any set of DAG gadgets is in NP.*

Proof. If a gadget and its state is a sink in the state-transition graph, then no transitions are available from that state. Each time a gadget is traversed the state of the gadget is moved down the graph. All paths from any vertex to a leaf are of polynomial length and thus each gadget can only be traversed polynomially many times before it no longer has any open tunnels. Thus we can give a polynomial size witness consisting of the order in which gadgets are visited, as well as the transition made at each gadget. To verify this certificate we check that each specified transition is legal, and that the location after each transition is connected to the location before the next transition in the witness. ◀

Recall from Theorem 2 that all gadgets without interacting tunnels are in NL. Thus one might hope to show that all interacting-k-tunnel DAG gadgets are NP-complete. This is true for deterministic gadgets but false in general; nondeterministic gadgets require a more careful categorization. We will define two behaviors a DAG gadget might have, "distant opening" and "forced distant closing", and show that either behavior guarantees NP-hardness, while having neither one puts the gadget in NL.

A **distant opening** in a DAG gadget is a transition in some state across a tunnel which opens a different tunnel.

▶ **Lemma 19.** *1-player motion planning with any k-tunnel DAG gadget with a distant opening is NP-hard.*

Proof. We show this problem is hard by a standard reduction from 3SAT. See Appendix A.2 for a definition of 3SAT.

We construct our reduction as follows. We use the tunnel which is traversed in the distant opening and one of the tunnels it opens. Each literal in a 3-CNF formula will be represented by those two tunnels in a single gadget, in the state of the distance opening. Each variable x_i is represented by a connection to two different paths, one which goes through the opening transitions for the x_i literals, and one for the $\neg x_i$ literals. We place a single-use gadget at the start and end of each branch of each variable to ensure only one side of the variable is traversed. The single-use gadget prevents the agent from returning on the same branch, and if the agent returns via the other branch, they will not be able to proceed to the next variable.

Each clause contains connections between the openable tunnels for each of its literals. All variable gadgets are laid out in series followed by the clause gadgets, with the goal location at the end of the clause gadgets. Each clause gadget can only be traversed if at least one of its corresponding variable gadgets has been traversed, allowing at least one passage to be open. The agent can reach the goal location exactly when it has a path through the variable gadgets which makes each clause gadget traversable, which corresponds to a satisfying assignment of the 3-CNF formula. ◀

When a transition across a tunnel closes another tunnel, the situation is more complicated, since the agent may be able to cross the same tunnel through a different transition, choosing not to close the other tunnel. For distant openings, the agent always chooses to open the other tunnel. We will now consider only **monotonically closing** DAG gadgets, which are DAG gadgets with no distant openings. We clarify some terminology regarding k-tunnel DAG gadgets. A **transition** is an edge in the transition graph, which is a legal move between locations which changes the state of the gadget. A **traversal** in a state is an orientation of a tunnel which is open in that state. A traversal may correspond to multiple transitions; a gadget being deterministic is equivalent to each traversal having only one transition. An **orientation** of a set of tunnels in a state contains, for each tunnel in the set, a single traversal of the tunnel the state.

For NP-completeness one might suggest there exist a traversal such that all of its transitions close some other traversal. However, this fails in a simple two tunnel case where one transition closes one direction of the other tunnel and the other transition closes the other direction. This leads us to a more complex definition. A **forced distant closing** in a state of a DAG gadget is a traversal across a tunnel in that state and an orientation of some other tunnels in the state such that, for each transition corresponding to the traversal, the transition closes some traversal in the orientation. The **size** of a forced distant closing is the number of traversals in the orientation.

▶ **Lemma 20.** *1-player motion planning with any monotonic k-tunnel DAG gadget with a forced distant closing is NP-hard.*

Proof. Consider all states which have forced distant closings, and let s be such a state that is minimal in the state-transition DAG, so that after making a transition from state s there are no forced distant closings. We will use a forced distant closing in s with smallest size; say this forced distant closing traverses tunnel t and has size i. We chain the i tunnels in the orientation for the forced distant closing, in the directions specified by the orientation, to make what is effectively a single long tunnel r. We will use the tunnels t and r in a reduction from 3SAT, and they have two important properties:

- If the agent traverses t, it cannot later traverse r: since we are using a forced distant closing, after traversing t at least one (oriented) tunnel in r is not traversable. Since there are no distant openings, this tunnel cannot become traversable again.
- The agent can traverse r from state s: in state s, each tunnel in r is open. The agent begins by traversing the first tunnel in r. This cannot be a forced distant closing for the remaining $i-1$ tunnels, since we assume the smallest forced distant closing has size i. So the agent can choose a transition which leaves the remaining tunnels in r open. After this first traversal, there are no more forced distant closings, so the robot can always choose a transition which leaves the remaining tunnels in r open.

We can now describe the reduction, which is very similar to the reduction in the proof of Lemma 19. Each literal in a 3-CNF formula is represented by a gadget in state s, with the tunnels r chained together. Each variable x_i is represented by a connection to two different paths, one which goes through t for the x_i literals, and one for the $\neg x_i$ literals. We place a single-use gadget at the start and end of each branch of each variable to ensure only one side of the variable is traversed. The single-use gadget prevents the agent from returning on the same branch, and if the agent returns via the other branch, they will not be able to proceed to the next variable.

When the agent goes through the x_i (resp $\neg x_i$) path of a variable, it closes r in the gadget for each literal x_i ($\neg x_i$), which corresponds to assigning x_i to false (true). This is reversed from the reduction for gadgets with distant openings.

Each clause contains connections between the r for each of its literals. All variable gadgets are laid out in series followed by the clause gadgets, with the goal location at the end of the clause gadgets. Each clause gadget can only be traversed if at least one of its corresponding variable gadgets has *not* been traversed, leaving at least one passage r open. The agent can reach the goal location exactly when it has a path through the variable gadgets which leaves each clause gadget traversable, which corresponds to a satisfying assignment of the 3-CNF formula. ◀

▶ **Lemma 21.** *1-player motion planning with any monotonic k-tunnel DAG gadget with no forced distant closing is in NL.*

Proof. The proof follows that of Theorem 2, though we must be more careful to account for optional distant closings. As in Theorem 2, if a system of gadgets has a solution, then a solution of minimal length does not intersect itself. This only requires that the gadget has no distant openings, since then making transitions can never increase traversability, and the shortcutting argument applies.

We locally convert the system of gadgets into a directed graph, and show a path in the graph from the start location to the goal location corresponds to a solution to the system of gadgets which does not intersect itself. Given a (not self-intersecting) path in the graph, we follow the corresponding path through the system of gadgets. When we make a traversal, we must pick a transition to avoid closing tunnels we will need later. This is always possible because there are no forced distant closings; we can always choose a transition which does not close any traversal in the orientation consisting of the traversals the path will later take. By doing this, we ensure that every traversal we need is available when we get to it, so the system of gadgets is solvable.

Suppose there is a solution to the system of gadgets that does not intersect itself. Since it uses each tunnel at most once, and the gadget has no distant openings, the traversability of each tunnel does not change before the solution uses it. Thus the solution is also a path in the directed graph.

So the system of gadgets has a solution iff there is a path from the start location to the end location in the directed graph. Since we can locally convert the system of gadgets to the graph in logarithmic space and solve reachability in NL, the motion planning problem is in NL. ◀

Combining Lemmas 17, 18, 19, 20, and 21, we have our dichotomy:

▶ **Theorem 22.** *1-player motion planning with a k-tunnel DAG gadget is NP-complete if the gadget has a distant opening or forced distant closing, and otherwise is in NL.*

It is natural to wonder whether this condition for hardness can be checked in polynomial time. That is, is there a polynomial-time algorithm which determines whether 1-player motion planning with a given DAG gadget is NP-complete? For all of our other dichotomies, the question of whether a gadget of the appropriate type satisfies the condition for hardness is clearly in P; in fact, in L. But a forced distant closing involves an orientation of the tunnels in the gadget, so there may be exponentially many potential forced distant closings to check. We will show that whenever it is necessary to search through each potential forced distant closing, the number of states of the gadget is exponential in the number of tunnels, so the search takes time polynomial in the number of states.

First, it is easy to determine whether a DAG gadget has a distant opening in polynomial time, since we can iterate through the transitions and see whether each one opens another tunnel. So we consider gadgets with no distant openings, and wish to determine whether they have a forced distant closing.

▶ **Lemma 23.** *Suppose a monotonic DAG gadget has a state s with k open tunnels, and there are no forced distant closings from states reachable from s. Then the gadget has at least 2^k states reachable from s.*

Proof. For each subset of the open tunnels in s, we will find a state that has exactly those tunnels open. Since there are 2^k such subsets, this implies there are at least 2^k states. Assume without loss of generality that each tunnel is traversable from left to right in state s.

Given a subset X of the open tunnels, we perform transitions starting from s as follows. For each tunnel not in X, traverse the tunnel repeatedly until it is closed in both directions; this must happen eventually because the gadget is a DAG. At each traversal, choose a transition which does not close any other tunnel from left to right. If there were no such choice of transition, that traversal with all other tunnels oriented from left to right would be a forced distant closing, which does not exist by assumption.

After making these transitions, we have closed each tunnel not in X without closing any tunnels in X. Since the gadget is monotonic, we have not reopened any tunnel. So the final state has exactly the tunnels in X open. ◀

▶ **Theorem 24.** *Deciding whether a 1-player motion planning with a k-tunnel DAG gadget is NP-complete can be done in polynomial time.*

Proof. The following algorithm checks in polynomial time whether 1-player motion planning with a given a DAG gadget is NP-complete.

- For each transition, see whether it is a distant opening. If it is, accept.
- Iterate through the states of the gadget in reverse order; i.e. check each state reachable from s before checking s. For each state, and for each traversal from that state:

- Suppose the state has k open tunnels other than the tunnel of the traversal. If every transition corresponding to the traversal leaves fewer than k of these tunnels open, accept.
- Enumerate the 2^k orientations of these k open tunnels, and check for each orientation whether it is a forced distant closing with the traversal. If it is, accept.

- Reject.

If the gadget has a distant opening, the algorithm notices it in the first step. Otherwise, we check for each state and traversal whether it has a forced distant closing. If every transition for a traversal reduces the number of other open tunnels, than any orientation of the other tunnels gives a forced distant closing. Otherwise, we check for each orientation whether it gives a forced distant closing. So the algorithm accepts exactly when the gadget has a distant opening or a forced distant closing, which is when 1-player motion planning with the gadget is NP-complete by Theorem 22.

The only step of the algorithm which does not obviously take polynomial time is running through all 2^k orientations of tunnels. Suppose the algorithm reaches this step for some state and traversal. Then there are no forced distant closings after making a transition from this state, since we would have accept already if there were. Also, there is some transition corresponding to the traversal which leaves all k other open tunnels open. By Lemma 23, there are at least 2^k states reachable after making this transition. In particular, the gadget has more than 2^k states, so enumerating the 2^k orientations takes time polynomial in the number of states. Thus the algorithm runs in polynomial time. ◀

6 2-Player Bounded Motion Planning

In this section, we show that it is PSPACE-complete to decide who wins in a 2-player race with any nontrivial DAG gadget (having at least one transition). To do so we give a construction that shows hardness for single-use paths and single-use one-way gadgets by a reduction from QBF. A simpler construction is possible, but this construction is more easily adapted to the team game in Section 7. This gives us a nice example of the 2-player local motion planning problem fitting into the canonical complexity class for two-player bounded games. It is also of interest because of how incredibly simple this gadget is. Two-location gadgets trivially do not have interacting tunnels (there is no other tunnel to interact with) and thus the 1-player version of these problems are contained in NL by Theorem 2.

▶ **Lemma 25.** *2-player motion planning with any set of DAG gadgets is in PSPACE.*

Proof. Since each gadget can undergo only a polynomial number of transitions, the length of the game is polynomially bounded. An alternating Turing machine which uses $\forall$ states to pick Black's moves and $\exists$ state to pick White's moves can simulate the game in polynomial time, so the motion planning problem is in AP = PSPACE. ◀

▶ **Lemma 26.** *2-player motion planning with the single-use bidirectional gadget is PSPACE-complete.*

Proof. Containment in PSPACE follows from Lemma 25. For PSPACE-hardness, we reduce from quantified boolean formulas (QBF). See Appendix A.2 for a definition of QBF.

We begin by describing the gadgets used in the reduction. The variable gadget is shown in Figure 27. Most of the gadget is two branches, corresponding to a variable and its negation. Each branch has a series of forks separated by single-use paths. There will be a number of

forks depending on the number of occurrences of a literal in the formula; two forks are shown. Each side of each fork has two single-use paths in series. The game will be constructed so that White always prefers the top side of a fork to be traversable, and Black prefers them to be not traversable; the top of a fork will be used later in evaluating the formula.

During the game, both players will pass through each variable gadget, with one player taking each of the two branches. White will take the bottom side of each fork on their branch, and Black will take the top side. Afterwards, only the branch which White took will have forks whose top sides are traversable. Thus we consider the assignment of the variable to be the literal corresponding to the branch White takes.

Suppose both players are at the left end of a variable gadget, and it is Player 1's (who may be White or Black) turn. Player 1 picks a branch, and Player 2 must walk down the other branch. Player 1 arrives at the right end of the branches immediately before Player 2. If Player 1 proceeds along the bottom path, Player 2 wins, so Player 1 must take the top path, which takes one turn longer. After traversing the variable gadget, both players are at the right end, and it is Player 2's turn, so the other player gets to choose a branch in the next variable gadget.

The clause gadget is shown in Figure 28. There are three paths from the left end to the right end, corresponding to the literals in a clause. Each path goes through a fork in a variable gadget. After variables are assigned, the single-use paths on each end of the fork are used, as are either those on the top or those on the bottom of each fork. If the top single-use paths are used, that path through the clause gadget is blocked, and if the bottom paths are used, that path is open. White will ultimately win by traversing each clause gadget, so White prefers to use the bottom side of a fork, and Black prefers to use the top side.

Each path has a large amount of delay (gadgets in series) before and after the fork, so that trying to use the clause gadget during variable assignment results in losing before reaching the end of the delay.

The race gadget is shown in Figure 29. It ensures both players proceed though variable gadgets as fast as possible. Let Player 1 be the player who reaches the race gadget first in this situation, immediately before Player 2; they are also the player who did not pick the assignment of the last variable. If Player 1 takes the bottom path, Player 2 will win, so Player 1 takes the top path. Then Player 2 takes the bottom path, and now the two players have been separated.

If Player 1 arrives more than a turn ahead of Player 2, they can take the bottom path. The next turn, before Player two can do anything at the race gadget, Player 1 wins. If Player 2 reaches the race gadget first, they can take the top path and win.

Given a quantified boolean formula with V variables and C clauses, we construct a system of gadgets as follows. We assume the QBF has alternating quantifiers beginning with $\exists$. There is a series of variable gadgets connected end-to-end corresponding to the variables of the formula, in the order of quantification. The goal location inside each variable gadget is a win for alternating players, beginning with Black. The branches of the variable gadget corresponding to x correspond to the literals x and $\neg x$. Each branch of that variable gadget has enough forks that each instance of a x or $\neg x$ in the formula corresponds to a fork, and the two branches have the same number of forks.

There is a clause gadget for each clause in the formula, connected in series. The three branches of a clause gadget correspond to the three literals in the clause. Each branch goes through the fork in the appropriate variable gadget corresponding to that instance of the literal. The delay before and after each fork consists of $9C + 3V$ single-use paths. The right end of the last clause is connected to a White goal location.

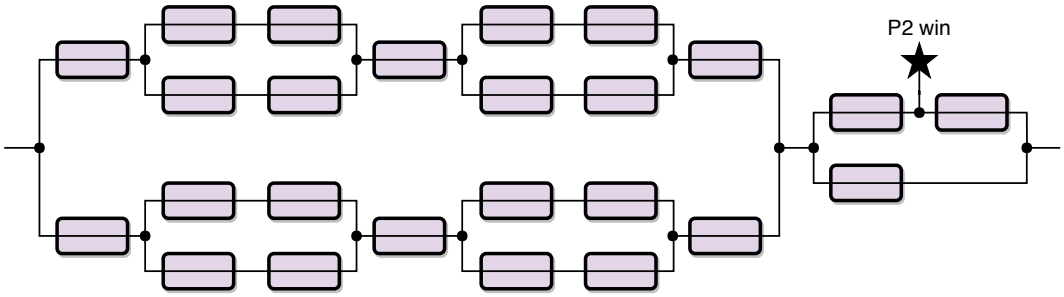


Figure 27 A variable gadget. The players arrive at the left, each take one path across, and exit at the right.

A race gadget is connected to the right end of the last variable gadget, with the goal locations such that Player 1 is the player with a win gadget inside the last variable gadget. The path with a White win gadget, which Black will walk down, is followed by $C(18C+6V+1)+2$ of single-use paths in series leading to a Black win gadget. The other path, which White will walk down, is connected to the first clause gadget.

Both players begin at the left end of the first variable gadget, and White goes first.

The game begins with White choosing a branch of the first variable gadget, corresponding to a choice of variable, and Black taking the other branch. Then Black chooses a branch of the second variable gadget, choosing the assignment of the variable based on the path White is forced to take. The players continue to take turns assigning variables. If either player deviates from this, such as by going into the delay in a clause gadget or by going backwards along another path, the other player will reach the race gadget first and win; the delay in clause gadgets is long enough to ensure that they do not have time to get through the clause gadget before losing. Otherwise both players arrive at the race gadget, and are sent down different branches.

White then proceed through each clause in series. Each branch of a clause is traversable if and only if the corresponding literal is true (since White took the bottom side and Black took the top side of each clause). The single-use paths between forks ensure that White cannot do anything other than progress through each clause gadget. If the formula is satisfied, White has a path through the clauses, and wins after $C(18C + 6V + 1)$ turns. If the formula is not satisfied, Black, who is walking down their long path, wins after slightly longer. Thus White has a forced win if and only if the quantified formula is true. ◀

▶ **Lemma 27.** *2-player motion planning with the single-use one-way gadget is PSPACE-complete.*

Proof. We again reduce from QBF. In the reduction in Lemma 26, neither player ever has to move through a single-use gadget to the left. Thus we can replace each bidirectional single-use gadget with a one-way single-use gadget pointing to the right, and the reduction still works. ◀

▶ **Corollary 28.** *2-player motion planning with any nontrivial DAG gadget is PSPACE-complete.*

Proof. As noted in Section 5 all DAG gadgets contain a single-use transition. This can be bidirectional or one-way, which are both shown to be PSPACE-hard in Lemmas 26 and 27. Containment in PSPACE is given by Lemma 25. ◀

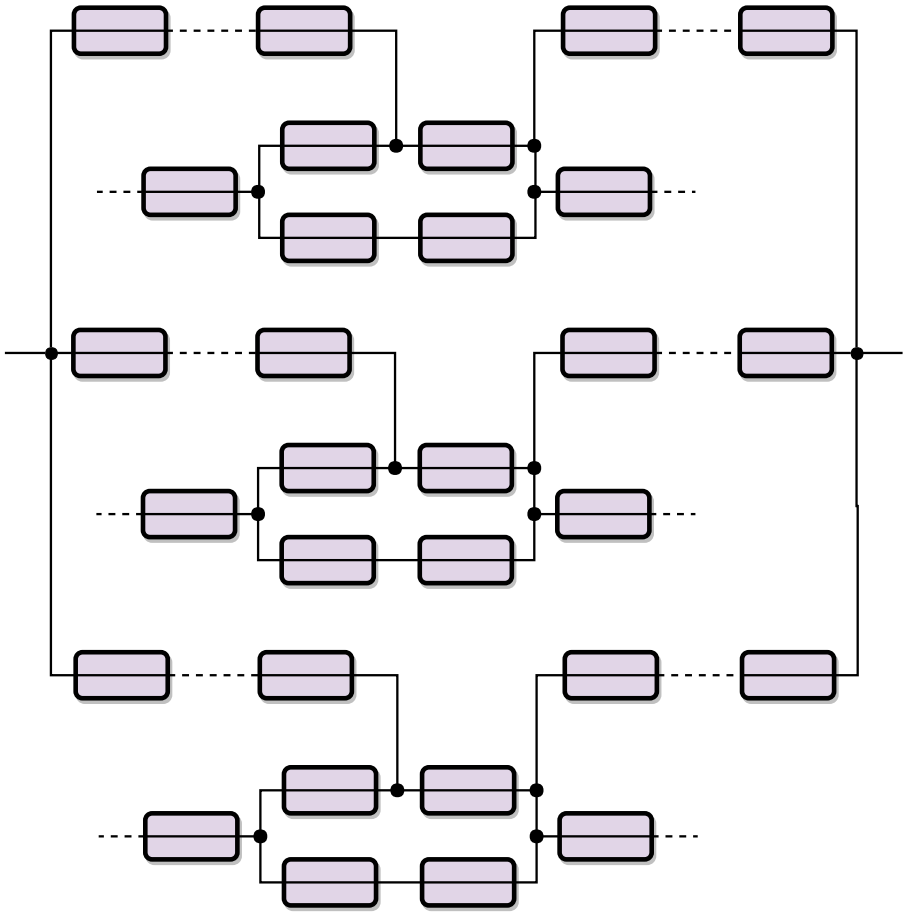


Figure 28 A clause gadget. Each literal is also part of a variable gadget. Each branch has a long series of gadgets so that it takes a large amount of time to traverse.

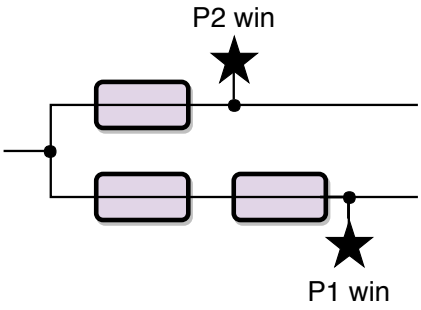


Figure 29 A race gadget. If Player 1 arrives at the left immediately before Player 2, each player ends up on one of the right exits. Otherwise, the player who arrives first wins.

7 Team Bounded Motion Planning

In this section we characterize the complexity of team imperfect information motion planning games with DAG gadgets. Since DAG gadgets are inherently bounded, the problem is in NEXPTIME, shown in Lemma 29. We go on to show in Lemma 30 that any nontrivial DAG gadget is NEXPTIME-complete by first giving a reduction from dependency quantified boolean formula (DQBF) for the single-use gadget. We then show that this proof adapts for single-use one-way gadgets. Since all DAG gadgets with at least one transition contain at least one of these, we achieve hardness for all such DAG gadgets.

▶ **Lemma 29.** *Team motion planning with any set of DAG gadgets is in NEXPTIME.*

Proof. A **partial history** for a player is the sequence of visible gadget states and moves made by that player, up to some point in the game. A **strategy** is a family of functions, one for each White player, that assign to each possible partial history a legal move from the position at the end of the partial history.

Since the gadget is a DAG, the game lasts a polynomial number of turns. Each player has polynomially many choices for each move, so there are only exponentially many possible sequences of moves, and only exponentially many possible partial histories for each player. Thus a strategy can be written in an exponential amount of space.

To determine whether White has a forced win in the team game, first nondeterministically pick a strategy. Then, for each possible sequence of moves the Black players could make, simulate the game with the White players following the strategy. If Black ever wins, reject; if White always wins, accept. This nondeterministic algorithm accepts if and only if there is some strategy White can use to force a win. The algorithm runs in exponential time because there are exponentially many sequences of moves the Black players might make, and the game for each such sequence takes a polynomial amount of time to simulate. Thus the algorithm decides the team game on systems of the gadget in NEXPTIME. ◀

▶ **Lemma 30.** *Team motion planning with the single-use bidirectional gadget is NEXPTIME-complete.*

Proof. Containment in NEXPTIME follows from Lemma 29. For NEXPTIME-completeness, we reduce from dependency quantified boolean formulas (DQBF). See Appendix A.2 for a definition of DQBF. In this reduction White represents the existential variables and Black represents the universal variables.

The reduction uses the same gadgets as that in Lemma 26, except that the clause gadget is modified as in Figure 30. This allows the White player checking the formula to try each literal, and return to the start of the clause gadget if the literal is false. This is necessary because the White player cannot see the state of the literals until arriving at them. For variable gadgets, we do not include the portion with a win gadget for Player 2 (the rightmost quarter or so in Figure 27), since we no longer want players to alternate choosing variables.

We construct the system of gadgets as follows. The overall structure is shown in Figure 31. For each set of variables $\vec{x}_1$, $\vec{x}_1$, $\vec{y}_1$, and $\vec{y}_2$, there is a corresponding set of variable gadgets (without the win gadget component) connected in series, followed by a race gadget. For simplicity, we will put C forks in each branch of each variable, where the formula has C clauses, though usually we need much fewer. Then each variable gadget takes $k = 3C + 1$ turns to traverse. We call the top path of a race gadget the **fast exit** and the bottom path the **slow exit**, since (in normal play) the first (second) player to arrive leaves through the fast (slow) exit. It will become clear which player each win gadget in a race gadget is for.

The turn order will be B, then W_1, then W_2. Both B and W_1 start at the beginning of the variable gadgets for $\vec{x}_1$. W_2 starts next to a delay line of length d_1. The fast exit of the race gadget for $\vec{x}_1$ and the end of this delay line both connect to the beginning of the $\vec{x}_2$ variable gadgets. The slow exit connects to a delay line of length d_2. The end of this delay line and the fast exit of the $\vec{x}_2$ race gadget connect to the beginning of the $\vec{y}_1$ variable gadgets, and the slow exit connects to a delay line of length d_3. The end of this delay line is connected to the *slow* exit of the $\vec{y}_1$ race gadget and the beginning of the $\vec{y}_2$ variable gadgets. The fast exit of the $\vec{y}_1$ race gadget is connected to yet another delay line of length d_4. The slow exit of the $\vec{y}_2$ race gadget is connected to a long delay line of length d_5 followed by a win gadget for B, and the fast exit is connected to a longer delay line of length $d_5 + 3$.

This all serves to accomplish the following. First, B chooses the assignment for $\vec{x}_1$ accompanied by W_1, so W_1 learns the assignment. Then B and W_1 are separated, and B assigns $\vec{x}_2$ accompanied by W_2. Next, W_1 chooses $\vec{y}_1$ accompanied by B, and finally W_2 chooses $\vec{y}_2$ accompanied by B. The delays d_1 through d_4 are chosen so that the White players arrive at exactly the right time; we have $d_1 = |\vec{x}_1|k + 1$, $d_2 = |\vec{x}_2|k - 1$, $d_3 = |\vec{y}_1|k$, and $d_4 = |\vec{y}_2|$. If a player deviates during variable assignment, they will arrive at their next race gadget too late, and lose.

The end of the final delay line for W_1, of length d_4, is connected to the first clause gadget, and the clause gadgets are connected in series corresponding to the clauses of the formula. The delay lines in each branch of each clause gadget have length Vk, where V is the number of variables; this ensures that if a player enters one of the delay lines during variable selection, an opponent will reach a race gadget and win before they accomplish anything. The end of the last clause gadget is connected to a win gadget for W_1. When W_1 reaches each clause gadget, they try the literals one at a time. When they cross the delay line to the fork, if the fork is traversable, they move on to the next clause. Otherwise they return through the other delay line and try the next literal. Each clause takes up to $6Vk + 1$ turns to cross.

If the formula is satisfied, W_1 eventually gets through all the clauses and wins. Otherwise, B wins after walking through their delay line of length d_5, which we can set to $C(6Vk+1)+1$.

We have seen that no player or team can benefit by deviating from normal play, and normal play is equivalent to the game corresponding to the DQBF. Thus White has a forced win if and only if the DQBF is true. ◀

▶ **Lemma 31.** *Team motion planning with the single-use one-way gadget is NEXPTIME-complete.*

Proof. The reduction in Lemma 30 still works when we replace each single-use bidirectional gadget with a one-way bidirectional gadget. We have to be a bit more careful than in Lemma 27: of the two paths in a clause gadget from the beginning to a fork, we need one path to point to the right and the other to point to the left, allowing W_1 to return from that fork. All other gadgets point to the right. ◀

▶ **Corollary 32.** *Team motion planning with any nontrivial DAG gadget is NEXPTIME-complete.*

Proof. Every DAG gadget has a single-use transition, which may be either bidirectional or one-way. Both cases are shown to be NEXPTIME-hard in Lemmas 30 and 31. Containment in NEXPTIME is Lemma 29. ◀

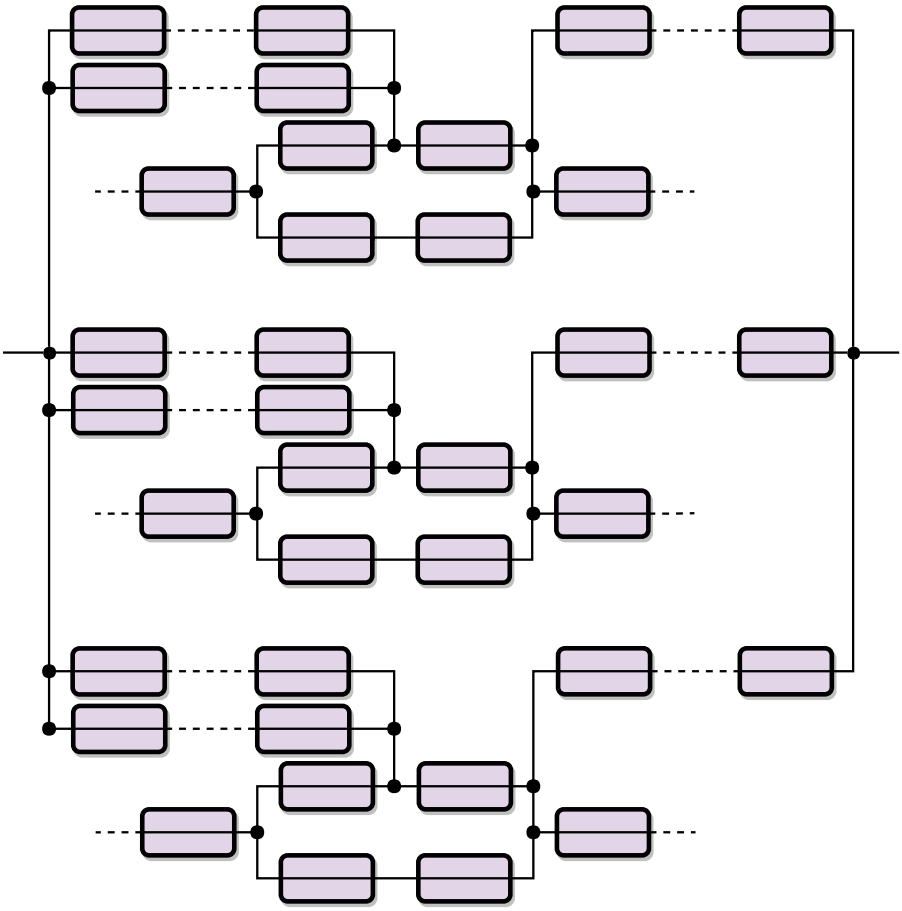


Figure 30 A clause gadget for team games. There are now two paths from the entrance of the clause to each fork, so the White player traversing the clause can return if they discover the fork is not traversable.

8 Applications

In this section we give examples of some known hard problems whose proofs can be simplified by using this motion planning framework.

8.1 PushPull-1F

In this section, we use the results of this paper to provide a simple proof that a Sokoban variant called PushPull-1F is PSPACE-hard, by reducing from motion planning in planar systems of locking 2-toggles (Section 2.3). This problem, and many related problems, were considered in [5] and were shown to be PSPACE-complete in [15] by a reduction from nondeterministic constraint logic; our reduction is much more straightforward using the infrastructure of the gadget framework.

▶ **Definition 33.** *In* ***PushPull-1F****, there is a square grid containing movable blocks, fixed blocks, an agent, and a goal location. The agent can freely move through empty squares, but can't move through blocks. The agent can push or pull one movable block at a time. The agent wins by reaching the goal location. The corresponding decision problem is whether a given instance of PushPull-1F is winnable.*

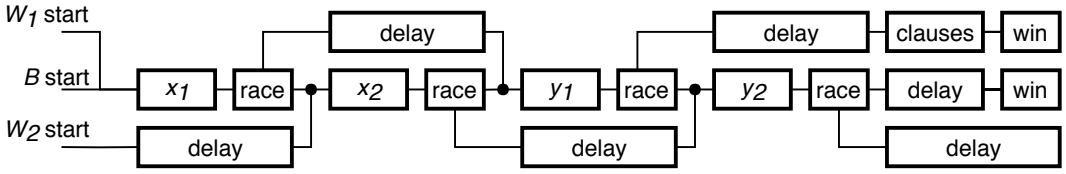


Figure 31 The high-level structure of the DQBF reduction.

In the notation "PushPull-1F", "PushPull" indicates that the agent can both push and pull, "1" indicates the number of blocks which can be moved at a time, and "F" indicates the existence of fixed blocks [5].

▶ **Theorem 34** ([15]). *PushPull-kF is PSPACE-hard for* $k \geq 1$.

Proof. We reduce from 1-player planar motion planning with locking 2-toggles, shown PSPACE-complete in Theorem 10. The (planar) connection graph is implemented using tunnels built with fixed blocks, and the agent and target location are placed appropriately. It suffices to build a gadget which behaves as a locking 2-toggle.

Such a gadget is shown in Figure 32. The two tunnels, currently both traversable, go from top to left and right to bottom. They interact in the center, where traversing either tunnel requires pushing a block into the middle square, which blocks the other tunnel. This is surrounded by four 1-toggles, which prevent additional traversals which aren't possible in a locking 2-toggle. Each 1-toggle is a room with 3 blocks, which can only be entered on one side. Upon entry, the agent can move the blocks to reveal the other exit, but doing so requires blocking the entrance taken, which flips the 1-toggle. ◀

8.2 Mario Kart

Mario Kart is a popular Nintendo racing game whose computational complexity was considered in [3] which showed NP-completeness for 1 player races and PSPACE-completeness for 2 player races with reductions from 3SAT and QSAT respectively. Using results from this paper, the 2 player proof now only needs a single, simple gadget, reducing a several page proof to a paragraph.

▶ **Theorem 35.** *Deciding if a player can force a win in two player Mario Kart is PSPACE-hard.*

Proof. A single-use one-way gadget can be constructed from a ramp and Dash Mushroom in Mario Kart. We place a ramp before a gap in the track long enough that a racer going at the normal maximum speed will not be able to make the jump and will fall onto another track that will take a long time to reach the finish line, ensuring they lose. However, this gap is small enough that, if the player uses a Dash Mushroom before, the increase in speed will allow them to make the jump. We put a single Dash Mushroom power-up before each ramp, ensuring the first racer to arrive can pick up the item and use it to cross the gap. To ensure a racer does not pick up the item and then keep it for later use, we precede the mushroom and ramp with a one-way gadget implemented by a long-fall. Along with the trivial existence of crossovers and the finish line as a location based win condition, Mario Kart is PSPACE-hard by Theorem 27. ◀

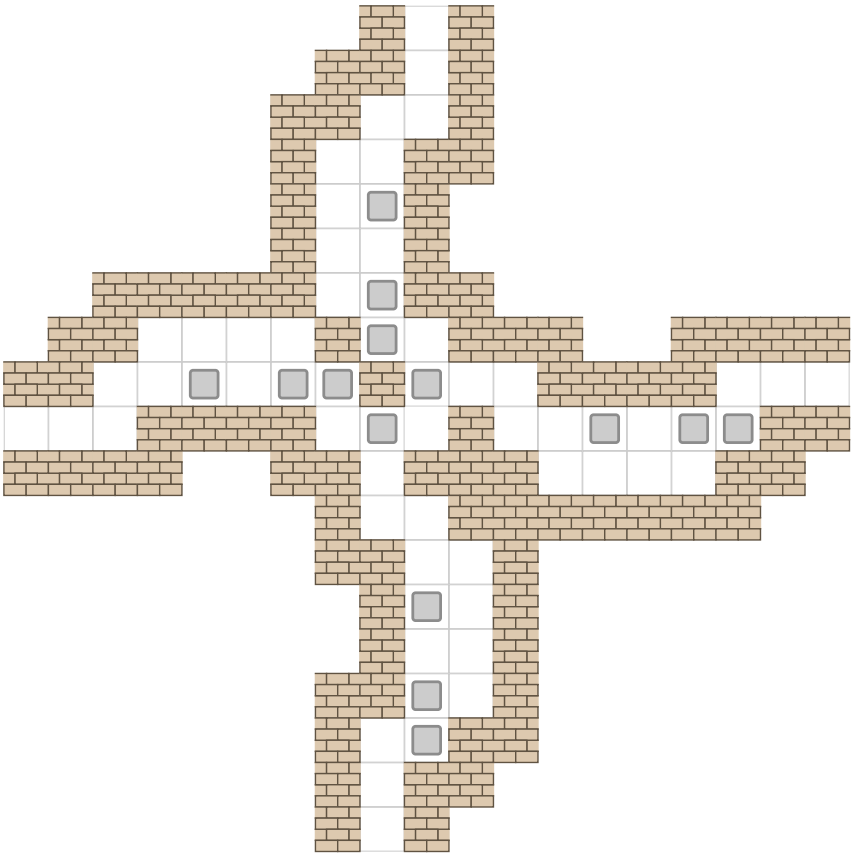


Figure 32 A locking 2-toggle in PushPull-1F.

9 Open Problems

This paper characterizes the complexity of two large classes of gadgets (DAG gadgets and reversible deterministic gadgets). Ideally, we could fully characterize the complexity of motion planning for every gadget type (and set of gadgets) as being easy or hard. There are many specific steps we might take towards this grand goal:

1. Is 2-player motion planning with 1-toggles EXPTIME-complete? This would complete our characterization for 2-player games with k-tunnel reversible deterministic gadgets. As an easier target, we could prove PSPACE-hardness, perhaps by adapting the 2-player proof for one-way closing gadgets.
2. Can we extend our characterizations of k-tunnel reversible deterministic gadgets to remove one of these restrictions? Specifically, non-tunnel gadgets, non-reversible gadgets, and nondeterministic gadgets are all interesting (and challenging) goals.
3. Which motion planning problems remain hard on planar systems of gadgets, like we proved for 1-player reversible deterministic? Are there any examples of gadgets where the planar version of the motion planning problem has a different complexity?

While we focused in this paper on general theory building, we can also explore the application of this motion planning framework to analyze the complexity of specific problems of interest. We conjecture that the results of this paper simplify many past hardness proofs, which can now be reduced to one or two figures showing how to build any hard gadget

according to our characterization, and how to connect gadgets together. See the hardness surveys [8, 11, 12, 4] for a large family of candidate problems. Of course, we also hope that this framework will enable the solution of many open problems in this space.

References

1 Greg Aloupis, Erik D. Demaine, Alan Guo, and Giovanni Viglietta. Classic Nintendo Games are (NP-)Hard. In *Proceedings of the 7th International Conference on Fun with Algorithms (FUN 2014)*, Lipari Island, Italy, July 2014.

2 Greg Aloupis, Erik D. Demaine, Alan Guo, and Giovanni Viglietta. Classic Nintendo Games are (Computationally) Hard. *Theoretical Computer Science*, 586:135–160, 2015. Originally at FUN 2014.

3 Jeffrey Bosboom, Erik D Demaine, Adam Hesterberg, Jayson Lynch, and Erik Waingarten. Mario kart is hard. In *Japanese Conference on Discrete and Computational Geometry and Graphs*, pages 49–59. Springer, 2015.

4 Erik D. Demaine. 6.890: Algorithmic Lower Bounds: Fun with Hardness Proofs. MIT lecture notes and videos, 2014. URL: `http://courses.csail.mit.edu/6.890/fall14/`.

5 Erik D. Demaine, Isaac Grosof, and Jayson Lynch. Push-Pull Block Puzzles are Hard. In *Proceedings of the 10th International Conference on Algorithms and Complexity*, pages 177–195, Athens, Greece, May 2017. `doi:10.1007/978-3-319-57586-5_16`.

6 Erik D. Demaine, Isaac Grosof, Jayson Lynch, and Mikhail Rudoy. Computational Complexity of Motion Planning of a Robot through Simple Gadgets. In *Proceedings of the 9th International Conference on Fun with Algorithms (FUN 2018)*, pages 18:1–18:21, La Maddalena, Italy, June 13–15 2018.

7 Erik D. Demaine and Robert A. Hearn. Constraint Logic: A uniform framework for modeling computation as games. In *Proceedings of the 23rd Annual IEEE Conference on Computational Complexity*, pages 149–162, June 2008.

8 Erik D. Demaine and Robert A. Hearn. Playing Games with Algorithms: Algorithmic Combinatorial Game Theory. In Michael H. Albert and Richard J. Nowakowski, editors, *Games of No Chance 3*, volume 56 of *Mathematical Sciences Research Institute Publications*, pages 3–56. Cambridge University Press, 2009.

9 Michal Forisek. Computational Complexity of Two-Dimensional Platform Games. In *Proceedings International Conference on Fun with Algorithms (FUN 2010)*, pages 214–227, 2010.

10 Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.

11 Robert A. Hearn and Erik D. Demaine. *Games, Puzzles, and Computation*. A. K. Peters, Ltd., Natick, MA, USA, 2009.

12 Graham Kendall, Andrew J. Parkes, and Kristian Spoerer. A Survey of NP-Complete Puzzles. *ICGA Journal*, 31(1):13–34, 2008. URL: `http://www.bibsonomy.org/bibtex/25cc2e9e313951aad1b2127959e6b2269/dblp`.

13 Sanjeev Khanna, Madhu Sudan, Luca Trevisan, and David P. Williamson. The Approximability of Constraint Satisfaction Problems. *SIAM Journal on Computing*, 30(6):1863–1920, 2000. `doi:10.1137/S0097539799349948`.

14 Dénes Kőnig. Über eine Schlussweise aus dem Endlichen ins Unendliche. *Acta Sci. Math. (Szeged)*, 3(2–3):121–130, 1927.

15 André G. Pereira, Marcus Ritt, and Luciana S. Buriol. Pull and PushPull are PSPACE-complete. *Theoretical Computer Science*, 628:50–61, 2016. `doi:10.1016/j.tcs.2016.03.012`.

16 Gary L. Peterson and John H. Reif. Multiple-person Alternation. In *Proceedings of the 20th Annual Symposium on Foundations of Computer Science*, SFCS '79, pages 348–363, Washington, DC, USA, 1979. IEEE Computer Society. `doi:10.1109/SFCS.1979.25`.

17 Thomas J. Schaefer. The complexity of satisfiability problems. In *Proceedings of the 10th Annual ACM Symposium on Theory of Computing*, pages 216–226, San Diego, California, May 1978.

18 Tom C Van Der Zanden and Hans L Bodlaender. PSPACE-completeness of Bloxorz and of games with 2-buttons. In *International Conference on Algorithms and Complexity*, pages 403–415. Springer, 2015.

19 Giovanni Viglietta. Gaming is a hard job, but someone has to do it! *Theory of Computing Systems*, 54(4):595–621, 2014.

20 Giovanni Viglietta. Lemmings is PSPACE-complete. *Theoretical Computer Science*, 586:120–134, 2015.

A Problem Definitions

In this appendix, we give formal definitions for the known hard problems used in this paper. In the paper we use single player, 2-player, and team imperfect information versions of Constraint Logic and Boolean Formula Games. The exact problems are specified in the following sections.

A.1 Constraint Logic

Constraint Logic [7, 11] is a uniform family of games – one-player, two-player, or team, with both bounded and unbounded variants – with the appropriate complexity in each case (as in Table 1). We will only describe the unbounded variants of Constraint Logic, as we use formula games for our bounded reductions. We also do not describe zero-player Constraint Logic, as we do not need it here.

In general, a **constraint graph** is an undirected maximum-degree-3 graph, where each edge has a weight of 1 (called a red edge) or 2 (called a blue edge). A **legal configuration** of a constraint graph is an orientation of the edges such that, at every vertex, the total incoming weight is at least 2. A **legal move** in a legal configuration of a constraint graph is a reversal of a single edge that results in another legal configuration.

In **1-player** Constraint Logic (also called **Nondeterministic Constraint Logic or NCL**), we are given a legal configuration of a constraint graph and a target edge e, and we want to know whether there is a sequence of legal moves ending with the reversal of target edge e. In this game, two types of vertices suffice for PSPACE-completeness: an OR vertex has exactly three incident blue edges, and an AND vertex has exactly one incident blue edge and exactly two incident red edges. We can also assume that each OR vertex can be assigned two "input" edges, and the overall construction is designed to guarantee that at most one input edge is incoming at any time; thus, we only need a "Protected OR" gadget which does not handle the case of two incoming inputs. Furthermore, the problem remains PSPACE-complete for planar constraint graphs.

In **2-player Constraint Logic (2CL)**, each edge of a constraint graph is also colored either black or white, and two players named Black and White alternate making valid moves where each player can only reverse an edge of their color. Given a legal configuration of a constraint graph, a target white edge for White, and a target black edge for Black, the goal is to determine whether White has a forced win, i.e., a strategy for reversing their target edge before Black can possibly reverse their target edge. In this game, six types of vertices suffice for EXPTIME-completeness: AND and OR vertices where all edges are white, AND vertices where all edges are black, AND vertices where the blue edge is white and one or both of the red edges are black, and degree-2 vertices where exactly one edge is black.

In **Team Private Constraint Logic (TPCL)**, there are two players on the White team and one player on the Black team, who play in round-robin fashion. In each move, the player can reverse up to a constant number k of edges of their color. Each player has a target edge to reverse, and can see the orientation of a specified set of edges, including edges of their own color and edges incident to those edges. Given a legal configuration of a constraint graph, the goal is to determine whether the White team has a forced win; i.e., whether one of the White players can reverse their target edge before Black can. In this game, all possible black/white colorings of AND and OR vertices suffice for RE-completeness. (Only undecidability has been claimed before, but RE-completeness follows by the same arguments.)

A.2 Formula Games

A **3-CNF formula** is a boolean formula φ of the form $C_1 \wedge \cdots \wedge C_k$, where each **clause** C_i is the disjunction of up to three **literals**, which are variables or their negations. An **assignment** for such a formula specifies a truth value for each variable, and is **satisfying** if the formula is true under the assignment.

In **3SAT**, we are given a 3-CNF formula, and we want to know whether it has a satisfying assignment. 3SAT is NP-complete [10].

A **partially quantified boolean formula** is a formula of the form $Q_1 x_1 : \cdots : Q_n x_n : \varphi$, where Q_i is one of the quantifiers $\forall$ or $\exists$, x_i is a (distinct) variable, and φ is a 3-CNF formula. An **assignment** for a partially quantified boolean formula specifies a truth value for each variable in φ that is not any x_i, called **free** variables. For a partially quantified boolean formula $\psi = Q_1 x_1 : \cdots : Q_n x_n : \varphi$ with $n > 0$, let $\psi' = Q_2 x_2 : \cdots : Q_n x_n : \varphi$. Given an assignment S for ψ, define assignments $S + x_1$ and $S + \neg x_1$ for ψ' which assign the same truth value as S to each free variable of φ and assign "true" and "false" to x_1, respectively. The truth value of ψ under S is defined recursively as follows:

- If $n = 0$ (so $\psi = \varphi$), ψ is true under S if and only if φ is true under S.
- If $n > 0$ and $Q_1 = \forall$, ψ is true under S if and only if ψ' is true under both $S + x_1$ and $S + \neg x_1$.
- If $n > 0$ and $Q_1 = \exists$, ψ is true under S if and only if ψ' is true under at least one of $S + x_1$ and $S + \neg x_1$.

A **quantified boolean formula** is a partially quantified boolean formula with no free variables. A quantified boolean formula has only one assignment (which is empty), so we say it is true if it is true under this unique assignment.

The truth value of a quantified boolean formula $\psi = Q_1 x_1 : \cdots : Q_n x_n : \varphi$ is equivalent to whether $\exists$ has a forced win in the following game: two players $\exists$ and $\forall$ choose an assignment for φ by assigning variables in the order they are quantified, with player Q_i choosing the truth value of x_i. $\exists$ wins if the assignment satisfies φ.

In **QBF**, we are given a (fully) quantified boolean formula, and we want to know whether it is true. QBF is PSPACE-complete, even if we restrict to formulas with alternating quantifiers beginning with $\exists$. This restriction is equivalent to that $\exists$ and $\forall$ take alternating turns, with $\exists$ going first [10].

A **dependency quantified boolean formula** is a formula of the form $\forall x_1 : \cdots : \forall x_m : \exists y_1(s_1) : \cdots : \exists y_n(s_n) : \varphi$, where x_i and y_j are (distinct) variables, φ is a 3-CNF formula, and s_j is a subset of $\{x_i \mid i \leq m\}$. We also require that every variable in φ is some x_i or y_j (φ has no free variables). A **strategy** for a dependency quantified boolean formula is a collection of functions $f_j : \{\text{true}, \text{false}\}^{s_j} \to \{\text{true}, \text{false}\}$ for $j = 1, \ldots, n$. A strategy **solves**

a dependency quantified boolean formula if for every map $S : \{x_i \mid i \leq m\} \to \{\text{true}, \text{false}\}$, the assignment given by $x_i \mapsto S(x_i)$ and $y_j \mapsto f_j(S|_{s_j})$ satisfies φ. Intuitively, y_j is only allowed to depend on the variables in s_j. A quantified boolean formula is a special case of a dependency quantified boolean formula, where each $s_j = \{x_i \mid i < k\}$ for some k. A dependency quantified boolean formula is **true** if there is a strategy that solves it.

The truth value of a dependency quantified boolean formula $\forall x_1 : \cdots : \forall x_m : \exists y_1(s_1) : \cdots : \exists y_n(s_n) : \varphi$ is equivalent to whether the $\exists$ team has a forced win in the following game, which puts a team of one player $\forall$ against a team of players $\exists_j$ for $j = 1, \ldots, n$: $\forall$ picks a truth value for each x_i. $\exists_j$ sees the truth value for each element of s_j (and nothing else) and picks a truth value for y_j. The $\exists$ team wins if the resulting assignment satisfies φ.

In the **DQBF** problem, we are given a dependency quantified boolean formula, and we want to know whether it is true. DQBF is NEXPTIME-complete even if we restrict to formulas of the form $\forall \vec{x}_1 : \forall \vec{x}_2 : \exists \vec{y}_1(\vec{x}_1) : \exists \vec{y}_2(\vec{x}_2) : \varphi$, where $\vec{x}_i$ and $\vec{y}_i$ may contain multiple variables, and each variable in $\vec{y}_i$ can depend on all the variables in $\vec{x}_i$. This restriction is equivalent to requiring that the $\exists$ team has two players who each choose multiple variables, and they see disjoint exhaustive subsets of the variables $\forall$ picks [16].

Computational Pseudorandomness, the Wormhole Growth Paradox, and Constraints on the AdS/CFT Duality

Adam Bouland
Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, 617 Soda Hall, Berkeley, CA 94720, U.S.A.
abouland@berkeley.edu

Bill Fefferman
Department of Computer Science, University of Chicago, 5730 S Ellis Ave, Chicago, IL 60637, U.S.A.
wjf@uchicago.edu

Umesh Vazirani
Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, 671 Soda Hall, Berkeley, CA 94720, U.S.A.
vazirani@cs.berkeley.edu

Abstract

The AdS/CFT correspondence is central to efforts to reconcile gravity and quantum mechanics, a fundamental goal of physics. It posits a duality between a gravitational theory in Anti de Sitter (AdS) space and a quantum mechanical conformal field theory (CFT), embodied in a map known as the AdS/CFT dictionary mapping states to states and operators to operators. This dictionary map is not well understood and has only been computed on special, structured instances. In this work we introduce cryptographic ideas to the study of AdS/CFT, and provide evidence that either the dictionary must be exponentially hard to compute, or else the quantum Extended Church-Turing thesis must be false in quantum gravity.

Our argument has its origins in a fundamental paradox in the AdS/CFT correspondence known as the wormhole growth paradox. The paradox is that the CFT is believed to be "scrambling" – i.e. the expectation value of local operators equilibrates in polynomial time – whereas the gravity theory is not, because the interiors of certain black holes known as "wormholes" do not equilibrate and instead their volume grows at a linear rate for at least an exponential amount of time. So what could be the CFT dual to wormhole volume? Susskind's proposed resolution was to equate the wormhole volume with the quantum circuit complexity of the CFT state. From a computer science perspective, circuit complexity seems like an unusual choice because it should be difficult to compute, in contrast to physical quantities such as wormhole volume.

We show how to create pseudorandom quantum states in the CFT, thereby arguing that their quantum circuit complexity is not "feelable", in the sense that it cannot be approximated by any efficient experiment. This requires a specialized construction inspired by symmetric block ciphers such as DES and AES, since unfortunately existing constructions based on quantum-resistant one way functions cannot be used in the context of the wormhole growth paradox as only very restricted operations are allowed in the CFT. By contrast we argue that the wormhole volume is "feelable" in some general but non-physical sense. The duality between a "feelable" quantity and an "unfeelable" quantity implies that some aspect of this duality must have exponential complexity. More precisely, it implies that either the dictionary is exponentially complex, or else the quantum gravity theory is exponentially difficult to simulate on a quantum computer.

While at first sight this might seem to justify the discomfort of complexity theorists with equating computational complexity with a physical quantity, a further examination of our arguments shows that any resolution of the wormhole growth paradox must equate wormhole volume to an "unfeelable" quantity, leading to the same conclusions. In other words this discomfort is an inevitable consequence of the paradox.

11th Innovations in Theoretical Computer Science Conference (ITCS 2020).
Editor: Thomas Vidick; Article No. 63; pp. 63:1–63:2

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

2012 ACM Subject Classification Theory of computation → Quantum computation theory; Theory of computation → Pseudorandomness and derandomization; Theory of computation → Quantum complexity theory

Keywords and phrases Quantum complexity theory, pseudorandomness, AdS/CFT correspondence

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.63

Category Abstract

Related Version A full version of the paper is available at `https://arxiv.org/abs/1910.14646v1`.

Funding *Adam Bouland*: A.B. was supported in part by ARO Grant W911NF-12-1-0541, NSF Grant CCF-1410022, and a Vannevar Bush faculty fellowship.
Bill Fefferman: B.F. acknowledges support from AFOSR YIP number FA9550-18-1-0148.
Umesh Vazirani: U.V. was supported in part by ARO Grant W911NF-12-1-0541, NSF Grant CCF-1410022, a Vannevar Bush faculty fellowship, and the Miller Institute at U.C. Berkeley through a Miller Professorship.

Acknowledgements We thank Scott Aaronson, Adam Brown, John Preskill, Douglas Stanford, Lenny Susskind, and Brian Swingle for detailed comments. We also thank Chris Akers, Dorit Aharonov, Ahmed Almheiri, Ning Bao, Raphael Bousso, Matt DeCrosse, Helia Kamal, Dan Harlow, Patrick Hayden, Juan Maldacena, Saeed Mehraban, Dominik Neuenfeld, Sepehr Nezami, Fabio Sanches, Jonah Sherman, Jalex Stark, Vincent Su, Michael Walter, and Ying Zhao for helpful discussions.

New Query Lower Bounds for Submodular Function Minimization

Andrei Graur
Department of Mathematics, Princeton University, Princeton, NJ, USA
agraur@princeton.edu

Tristan Pollner
Department of Mathematics, Princeton University, Princeton, NJ, USA
tpollner@princeton.edu

Vidhya Ramaswamy
Department of Computer Science, Princeton University, Princeton, NJ, USA
vidhyar@cs.princeton.edu

S. Matthew Weinberg
Department of Computer Science, Princeton University, Princeton, NJ, USA
smweinberg@princeton.edu

Abstract

We consider submodular function minimization in the oracle model: given black-box access to a submodular set function $f : 2^{[n]} \to \mathbb{R}$, find an element of $\arg\min_S\{f(S)\}$ using as few queries to $f(\cdot)$ as possible. State-of-the-art algorithms succeed with $\tilde{O}(n^2)$ queries [13], yet the best-known lower bound has never been improved beyond n [6].

We provide a query lower bound of $2n$ for submodular function minimization, a $3n/2 - 2$ query lower bound for the non-trivial minimizer of a symmetric submodular function, and a $\binom{n}{2}$ query lower bound for the non-trivial minimizer of an asymmetric submodular function.

Our $3n/2 - 2$ lower bound results from a connection between SFM lower bounds and a novel concept we term the *cut dimension* of a graph. Interestingly, this yields a $3n/2 - 2$ cut-query lower bound for finding the global mincut in an undirected, weighted graph, but we also prove it cannot yield a lower bound better than $n + 1$ for s-t mincut, even in a directed, weighted graph.

2012 ACM Subject Classification Theory of computation → Submodular optimization and polymatroids

Keywords and phrases submodular functions, query lower bounds, min cut

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.64

Funding *S. Matthew Weinberg*: Supported by NSF CCF-1717899.

1 Introduction

Submodular function minimization (SFM) is a classic algorithmic problem with numerous applications (e.g. [1, 11, 12, 14]): given black-box access to a submodular[1] function $f : 2^{[n]} \to \mathbb{R}$, find an element of $\arg\min\{f(S)\}$. Due to its ubiquity within TCS and without, the problem has received substantial attention over the past four decades within various communities. Seminal work of Grötschel, Lovasz, and Schrijver first established that a minimizer can be found in poly-time [5], and after a long series of improvements the state-of-the-art now requires $\tilde{O}(n^2)$ value queries to $f(\cdot)$ and $\tilde{O}(n^3)$ additional overhead.

[1] $f(\cdot)$ is submodular if $f(X \cup Y) + f(X \cap Y) \leq f(X) + f(Y)$ for all sets X, Y. This is equivalent to $f(S \cup T \cup \{i\}) - f(S \cup T) \leq f(S \cup \{i\}) - f(S)$ for all S, T, i (called *diminishing marginal returns*).

Despite remarkable progress on the algorithmic front, shockingly few *lower bounds* on submodular function minimization are known. It is perhaps unsurprising that computational lower bounds are elusive, but *even query lower bounds are virtually non-existent.* Indeed, state-of-the-art query lower bounds for SFM have remained stagnant at exactly n for the past decade [6]. Our main results are new query lower bounds for three variants of SFM. We briefly provide the formal problem statements and our main results below, followed by an overview of context and related work.

▶ **Definition 1** (Query Complexity of Submodular Function Minimization). *Given as input black-box access to a submodular function $f(\cdot)$ over n elements, output an element of* $\arg\min_S\{f(S)\}$*, along with* $\min_S\{f(S)\}$*. The* query complexity *of SFM is equal to the minimum $q(\cdot)$ such that a deterministic algorithm solves SFM on all instances of n elements with at most $q(n)$ queries.*

- *If $f(\cdot)$ is further assumed* symmetric*, i.e. $f(S) = f([n] \setminus S)$ for all S, this is* Symmetric SFM*.*
- *If we ask for* $\arg\min_{S \notin \{\emptyset,[n]\}}\{f(S)\}$*, this is* Non-Trivial SFM *(we will also use both qualifiers).*[2]

As a representative problem to have in mind, imagine a graph on n nodes with positive edge weights and define $f(S)$ to be the weight of all edges leaving set S (the value of cut S). Then $f(\cdot)$ is submodular, and non-trivial symmetric SFM would count the number of *cut queries* needed to find the mincut. If you seek the minimum s-t cut (and adjust notation so that $f(S)$ is equal to the weight of all edges leaving $S \cup \{s\}$), then this is standard SFM (because it is valid to output $\emptyset$, which implies that the mincut is s). If you seek the global mincut in a directed graph, then this is an instance of Non-Trivial SFM (because it is now invalid to output $\emptyset$). If you seek the global mincut in an undirected graph, then this is an instance of Non-Trivial Symmetric SFM (because it doesn't matter which side of the cut is sending versus receiving). Our main results are below.

▶ **Theorem 2** (Main Results). *The following lower bound the query complexity of SFM:*

- *The query complexity of SFM is at least $2n$.*
- *The query complexity of Non-Trivial Symmetric SFM is at least $3n/2 - 2$.*
- *The query complexity of Non-Trivial SFM is at least $\binom{n}{2}$.*

1.1 New Technique: The Cut Dimension

Our SFM and Non-Trivial SFM lower bounds are direct constructions, and we defer all related intuition and technical details to the corresponding sections. Our Non-Trivial Symmetric SFM lower bound, however, derives from a new framework based on the *cut dimension* of graphs.

▶ **Definition 3** (Global Cut Dimension, special case of Definition 11). *Let G be a directed graph with m edges, and let S be a subset of nodes. Define $\vec{v}^S$ be the vector in $\mathbb{R}^m$ with $v_e^S = 1$ iff the edge e has left endpoint in S and right endpoint not in S (and $v_e^S = 0$ otherwise). Then the* cut dimension *of G is the dimension of span($\{\vec{v}^S,\ S$ is a global mincut$\}$). We also consider the following variants:*

[2] Indeed, note that Symmetric SFM (without the Non-Trivial qualifier) is trivial, as $\emptyset$ (or $[n]$) is always a solution.

- *If G is undirected, then $v_e^S = 1$ iff the edge e has one endpoint in S and the other not in S.*
- *If we seek the min s-t cut, then $v_e^S = 1$ iff the edge e has left endpoint in $S \cup \{s\}$ and right endpoint not in $S \cup \{s\}$. We also take the dimension over min s-t cuts instead of global mincuts. In this case, we call this the s-t Cut Dimension.*

Our main result concerning the cut dimension connects it to SFM lower bounds:

▶ **Theorem 4** (Special case of Theorem 15). *If an undirected graph exists with Global Cut Dimension d, then the query complexity of Non-Trivial Symmetric SFM is at least d.*

If a graph exists with s-t Cut Dimension d, then the query complexity of SFM is at least d.

If a directed graph exists with Global Cut Dimension d, then the query complexity of Non-Trivial SFM is at least d.

Interestingly, we also establish that the Cut Dimension is a *equivalent* to the best achievable lower bounds based on graphs via a canonical perturbation approach. Our $3n/2 - 2$ lower bound for Non-Trivial Symmetric SFM follows immediately from Theorem 4 and the construction of an undirected graph with Global Cut Dimension $3n/2 - 2$. We also establish that every graph has s-t Cut Dimension at most $n + 1$, meaning this approach is useful for Non-Trivial SFM but not SFM.

1.2 Related Work

The first poly-time (and strongly poly-time) algorithms for SFM were given by [5] using the Lovasz extension and the Ellipsoid algorithm [10]. A substantial series of improvements followed over the subsequent four decades [3, 18, 4, 8, 7, 19, 15, 9] The state-of-the-art is an $\tilde{O}(n^2)$ upper bound on the query complexity of SFM [13], an $O(n^3)$ upper bound on the query complexity of Non-Trivial Symmetric SFM [16], and an $\tilde{O}(n^3)$ upper bound on the query complexity of Non-Trivial SFM [13].[3]

Despite this substantial progress on upper bounds, the only unconditional query lower bound is just n (which is surprisingly non-trivial to establish) [6]. [2] give a construction which requires $\Omega(n)$ *queries to the Lovasz extension* (if the algorithm can only query the Lovasz extension), but one can find the minimizer in their construction by simply querying all n singletons.

Our Non-Trivial Symmetric SFM lower bound uses the cut function in graphs. Recent work of [17] establishes that this particular instance of Non-Trivial Symmetric SFM (in unweighted graphs) can be solved by a randomized algorithm in $\tilde{O}(n)$ queries, but our techniques are unrelated.

1.3 Roadmap

Section 2 provides our $2n$ lower bound for SFM, which is a direct construction. Section 3 proves (a generalization of) Theorem 4 and provides a graph with Global Cut Dimension $3n/2 - 2$, yielding our $3n/2 - 2$ lower bound for Non-Trivial Symmetric SFM. Section 4 provides our $\binom{n}{2}$ lower bound for Non-Trivial SFM, which is also a direct construction.

Appendix B contains auxiliary claims concerning cut queries in graphs (i.e., it is impossible to learn precisely a directed graph using cut queries, what can you learn?) which are not necessary for our lower bounds, but likely useful for future work. Appendix A contains one omitted proof.

[3] The final bound follows by a reduction from Non-Trivial SFM to SFM incurring a blowup of $2n$ (for all elements i, run SFM only over sets containing i and not containing $i + 1$, and then only over sets containing i and not $i - 1$).

2 A $2n$ Query Lower Bound for SFM

This section proves our lower bound on SFM.

▶ **Theorem 5.** *The query complexity of SFM is at least* $2n$.

Let us first provide intuition for our construction. We start with an arbitrary permutation σ on $[n]$, and define the *important sets* R_i for $0 \leq i \leq n$ by $R_i := \{\sigma(1), \sigma(2), ..., \sigma(i)\}$ for each $i \in [n] \cup \{0\}$. Observe that there is exactly one important set of each size from 0 to n. These important sets will be the potential minimizers. Intuitively, we will define our function such that: (a) any algorithm must query at least $n-1$ unimportant sets to learn the important sets, and (b) any algorithm must query all $n+1$ important sets to learn the minimizer. In detail:

- Let σ be an arbitrary permutation on $[n]$.
- Define $R_i := \{\sigma(1), \ldots, \sigma(i)\}$ for each $i \in [n] \cup \{0\}$.
- For each $i \in [n] \cup \{0\}$, let $c_i \in \{0, 1\}$.
- Define the function $f^{\vec{c}}_{\sigma}(\cdot)$ such that (below, $j(S)$ denotes the maximum j such that $R_j \subseteq S$):

$$f^{\vec{c}}_{\sigma}(S) = \begin{cases} -c_i & \text{if } S = R_i \text{ for some } 0 \leq i \leq n \\ (|S| - j(S)) \cdot (n + 2 - j(S)) & \text{else} \end{cases}$$

That is, $f^{\vec{c}}_{\sigma}(\cdot)$ is defined to be non-negative on the unimportant sets, and non-positive on the important sets. Intuitively, queries to unimportant sets give information regarding σ, and queries to important sets give information regarding $\vec{c}$. It is not obvious, but straightforward to establish that $f^{\vec{c}}_{\sigma}(\cdot)$ is submodular for all $\sigma, \vec{c}$. The proof of Lemma 6 appears in Appendix A.

▶ **Lemma 6.** *For all* $\sigma, \vec{c}$, $f^{\vec{c}}_{\sigma}$ *is submodular.*

We now provide a complete proof that deterministic algorithms must make $2n$ queries for functions of the form $f^{\vec{c}}_{\sigma}(\cdot)$. We define an adversary which adaptively sets $\sigma, \vec{c}$ as queries are made:

- Initialize $\sigma, \vec{c}$ to be undefined.
- Let i denote the maximum j such that $\sigma(j)$ is defined (so initially $i = 0$).
- When a new query, S, is made:
  1. If $S = R_j$, for some $j \leq i$, answer 0 and set $c_j = 0$. Call this an *important query*.
  2. If $R_i \not\subseteq S$, $j(S)$ is defined. Answer $(|S| - j(S)) \cdot (n + 2 - j(S))$. Call this a *useless query*.
  3. If $R_i \subset S$, $j(S)$ is not yet defined. Pick any $j \notin S$ (such a j must exist as $S \neq R_n$) and set $\sigma(i+1) = j$.[4] Now $j(S) := i$, so answer $(|S| - i) \cdot (n + 2 - i)$. Call this a *decoy query*.
- If the algorithm terminates after $n+1$ (distinct) important queries, σ and $\vec{c}$ are fully defined.
- If the algorithm has made fewer than $n+1$ (distinct) important queries, let x denote the algorithm's guess for the minimum value.
  1. If $x = 0$, set all undefined $c_i := 1$ and complete σ arbitrarily (if necessary).
  2. If $x = -1$, set all undefined $c_i := 0$ and complete σ arbitrarily (if necessary).
  3. If $x \notin \{0, -1\}$, complete $\vec{c}, \sigma$ arbitrarily.

[4] Observe also that $j \neq \sigma(\ell)$ for any $\ell \leq i$, as $R_i \subset S$.

Theorem 5 will follow by proving that the above adversary is consistent and that the adversary has the power to make multiple (distinct) minima unless the algorithm has made at least $n-1$ decoy queries and $n+1$ important queries.

▶ **Observation 7.** *The adversary answers all queries in a way that is consistent with some* $f^{\vec{c}}_{\sigma}(\cdot)$.

Proof. Observe that the adversary answers all queries to R_j with 0, so this is always consistent. Further observe that whenever an unimportant set is queried, either the answer is already determined by σ (and therefore consistent), or one new output of σ is fixed so that the answer is now determined by σ (and therefore consistent now and forever). The precise definition of $f^{\vec{c}}_{\sigma}(\cdot)$ is important for the final claim: as soon as we know that $\sigma(i+1) \notin S$, this fixes the value of $f^{\vec{c}}_{\sigma}(\cdot)$.

Finally, observe that the completion step is also consistent with all previous queries, as they are completely defined by the partial definition of $\sigma, \vec{c}$. ◀

▶ **Lemma 8.** *Algorithms cannnot make* $n+1$ *distinct important queries without* $n-1$ *decoy queries.*

Proof. Observe that each decoy query increases i by one. Observe that the only distinct important queries that can be made are $\emptyset, [n]$, and $R_1, \ldots, R_i$, for a total of $i+2$. If $i < n-1$, then the distinct possible important queries are also $< n+1$. ◀

▶ **Lemma 9.** *Any algorithm making* $< n+1$ *distinct important queries is wrong.*

Proof. If the guess is $\notin \{0, -1\}$, then the guess is clearly wrong. If the guess is 0, then the completion step makes it so that the minimum is -1, so the guess is wrong. If the guess is -1, then the completion step makes it so that the minimum is 0, so the guess is wrong. ◀

Proof of Theorem 5. Lemmas 9 and 8 together assert that the algorithm must make $n-1$ decoy queries and $n+1$ important queries in order to correctly solve SFM on instances of the form $f^{\vec{c}}_{\sigma}(\cdot)$ against the prescribed adversary. Therefore, a total of $2n$ queries must be made. ◀

We conclude this section by noting that our construction witnesses a lower bound of exactly $2n$ (and no better).

▶ **Proposition 10.** *An SFM algorithm exists making* $2n$ *queries for any function of the form* $f^{\vec{c}}_{\sigma}(\cdot)$.

Proof. First, query the $n-1$ sets $[n] \setminus \{i\}$ for all $i \neq 1$. Observe that $f^{\vec{c}}_{\sigma}([n] \setminus \{i\}) \leq 0$ if and only if $\sigma(n) = i$. Similarly, as $[n] \setminus \{i\}$ is missing only a single element (i), this means that if $\sigma(n) \neq i$ then $f^{\vec{c}}_{\sigma}([n] \setminus \{i\}) = (n - \sigma^{-1}(i)) \cdot (n + 3 - \sigma^{-1}(i))$. So the query to $[n] \setminus \{i\}$ reveals $\sigma^{-1}(i)$, for any i. Therefore, these $n-1$ queries completely reveal σ (because σ is a permutation).

After σ is fully revealed, simply query the $n+1$ important sets to find the minimizer. ◀

3 A $3n/2 - 2$ Query Lower Bound for Non-Trivial Symmetric SFM

We begin this section by providing a generalization the cut dimension, first by providing a class of submodular functions which generalize mincuts in graphs.

3.1 Defining the Generalized Cut Dimension

Consider a ground set of n elements, and a disjoint set of m *hyperedges*. We associate with each $S \subseteq [n]$ (including $S = \emptyset$) a set $h(S) \subseteq [m]$ of hyperedges that are active for S. For example, to capture mincuts in an undirected graph we might have the hyperedges simply be the edges of that graph, and $h(S)$ would denote the edges with one endpoint in S and the other not in S.

To each $i \in [m]$, associate a non-negative weight w_i, and define the function $f(\cdot)$ so that $f(S) := \sum_{i \in h(S)} w_i$. If the active sets $h(\cdot)$ satisfy the following inequality, then it is easy to see that $f(\cdot)$ is submodular (below, $X \overline{\cup} Y$ denotes the multiset union of X and Y, which contains two copies of every element in $X \cap Y$):

$$h(S \cap T) \overline{\cup} h(S \cup T) \subseteq h(S) \overline{\cup} h(T).$$

We call such functions *weight-based*. It is easy to see that cuts in graphs or hypergraphs are weight-based. For such functions, there is a meaningful notion of "dimension" associated with the set of minimizers. For every $S \subseteq [n]$, define the vector $\vec{v}^S \in \mathbb{R}^m$ so that $v_i^S = 1$ if and only if $i \in h(S)$ *and* $w_i > 0$, and $v_i^S = 0$ otherwise. For a set $\mathcal{S}$ of subsets of $[n]$, let $\dim(\mathcal{S})$ denote the dimension of the span (over $\mathbb{R}^m$) of the vectors $\{\vec{v}^S\}_{S \in \mathcal{S}}$. We now define the Generalized Cut Dimension:

▶ **Definition 11** (Generalized Cut Dimension)**.** *Let $f(\cdot)$ be weight-based. Then the* Generalized Cut Dimension *of $f(\cdot)$ is equal to dim*$(\arg\min_S\{f(S)\})$*. The* Generalized Non-Trivial Cut Dimension *of $f(\cdot)$ is dim*$(\arg\min_{S \notin \{\emptyset,[n]\}}\{f(S)\})$*.*

We will call a weight-based function symmetric if $h(S) = h([n] \setminus S)$ – it is clear that the resulting submodular function is symmetric.

3.2 Connecting Generalized Cut Dimension to Query Complexity

In this section, we establish the *equivalence* of Generalized Cut Dimension to a canonical "perturbation" approach for lower bounding the query complexity.

▶ **Definition 12** (Perturbation Bound)**.** *Starting from a (symmetric, if desired) weight-based submodular function $f(\cdot)$ with weights $\vec{w}$ and (non-trivial, if desired) minimizers $\mathcal{M}_f$, pick a sufficiently small $\varepsilon > 0$ so that every $\vec{w}'$ with $w'_i \in [(1-\varepsilon)w_i, (1+\varepsilon)w_i]$ induces a (symmetric, if desired) weight-based submodular function $g(\cdot)$ with (non-trivial, if desired) minimizers $\mathcal{M}_g \subseteq \mathcal{M}_f$. Let $\mathcal{G}(f)$ denote the set of all (symmetric, if desired) weight-based functions with $w'_i \in [(1-\varepsilon)w_i, (1+\varepsilon)w_i]$.*

If it is the case that, for any set of $q-1$ queries to f, there exists a $g \in \mathcal{G}(f)$ consistent with those queries such that $\min_S\{g(S)\} \neq \min_S\{f(S)\}$, we say that f witnesses a (Symmetric) Perturbation Bound *of q (if there is a $g(\cdot) \in \mathcal{G}(f)$ with $\min_{S \notin \{\emptyset,[n]\}}\{g(S)\} \neq \min_{S \notin \{\emptyset,[n]\}}\{f(S)\}$, we say that f witnesses a Non-Trivial Perturbation Bound of q).*

We refer to the *Perturbation Bound of f as the maximum possible q such that f witnesses a Peturbation Bound of q (and similarly can define* the *Non-Trivial Perturbation Bound).*

Intuitively, the perturbation bound captures the following natural way to obtain query lower bounds: start from some function $f(\cdot)$ with minimizers $\mathcal{M}_f$. There is some non-zero gap δ between the minimizers and the rest, so there exists a sufficiently small ε such that perturbing weights by ε can tie-break among minimizers, but not yield a new minimizer.

▶ **Observation 13.** *If there exists an $f(\cdot)$ witnessing a (Symmetric, Non-Trivial) Perturbation Bound of q, then the query complexity of (Symmetric, Non-Trivial) SFM is at least q.*

Proof. Assume for contradiction that an algorithm correctly outputs the minimum value, x, after $q-1$ queries that are consistent with $f(\cdot)$. If $x \neq \min_S\{f(S)\}$, then all queries are consistent with $f(\cdot)$, so the algorithm could be wrong because the function is $f(\cdot)$. if $x = \min_S\{f(S)\}$, then because $f(\cdot)$ witnesses a Perturbation Bound of q, there exists a $g(\cdot)$ consistent with all $q-1$ queries with $\min_S\{g(S)\} \neq x$, so the algorithm could be wrong because the function is $g(\cdot)$.

The same proof holds verbatim if $f(\cdot)$ is assumed to be symmetric, or if we replace absolute minimizers with non-trivial minimizers. ◀

We now establish that the perturbation bound approach yields exactly the same lower bound as the generalized cut dimension. Our proof will make use of the following observation.

▶ **Observation 14.** *For any $g \in \mathcal{G}(f)$, $g(S) = \vec{v}^S \cdot \vec{w}'$ (where $\vec{w}'$ is the weight vector defining $g(\cdot)$).*

Proof. First, recall that $g(S) := \sum_{i \in h(S)} w'_i$. Recall further that $v_i^S = 1$ whenever $w_i \neq 0$ and $i \in h(S)$. Importantly, note that $w_i = 0 \Leftrightarrow w'_i = 0$ for all $g(\cdot) \in \mathcal{G}(f)$, so in fact $v_i^S = 1$ whenever $w'_i \neq 0$ and $i \in h(S)$ (and 0 otherwise). This immediately implies that $\vec{v}^S \cdot \vec{w}' = \sum_{i \in h(S)} w'_i = g(S)$. ◀

▶ **Theorem 15.** *Let $f(\cdot)$ be (symmetric) weight-based. Then the (Symmetric, Non-Trivial) Perturbation Bound of $f(\cdot)$ is exactly equal to the Generalized (Non-Trivial) Cut Dimension of $f(\cdot)$.*

Proof. We break the proof down into two lemmas, one establishing that the Perturbation Bound is at most the Generalized Cut Dimension, and one establishing that the Perturbation Bound is at least the Generalized Cut Dimension. We first establish the easy direction, that if $f(\cdot)$ has Generalized (Non-Trivial) Cut Dimension d, it witnesses a (Symmetric, Non-Trivial) Perturbation Bound of at most d. For simplicity of notation throughout the proof, we explicitly prove the standard case, but the claims for Symmetric and Non-Trivial follow verbatim.

▶ **Lemma 16.** *For all (Symmetric) weight-based $f(\cdot)$, the (Symmetric, Non-Trivial) Perturbation Bound is at most the Generalized (Non-Trivial) Cut Dimension.*

Proof. Say that $f(\cdot)$ has Generalized Cut Dimension d, and let $S_1, \ldots, S_d$ be such that $v^{S_1}, \ldots, v^{S_d}$ form a basis for the span of $\{\vec{v}^S\}_{S \in \mathcal{M}_f}$. We claim that queries to $S_1, \ldots, S_d$ completely determine $g(S) = f(S)$ for all $S \in \mathcal{M}_f$ and all $g(\cdot) \in \mathcal{G}(f)$. If true, this establishes a set of q queries for which there does not exist a $g(\cdot) \in \mathcal{G}(f)$ consistent with these queries for which $\min_S\{g(S)\} \neq \min_S\{f(S)\}$ (and therefore the Perturbation Bound for $f(\cdot)$ is at most d).

Consider any $S \in \mathcal{M}_f$. Then we know by definition of the Generalized Cut Dimension that $\vec{v}^S = \sum_i c_i \vec{v}^{S_i}$ for some $c_1, \ldots, c_d \in \mathbb{R}$. We claim that this implies that $g(S) = \sum_i c_i g(S_i)$ *for any* $g(\cdot) \in \mathcal{G}(f)$. This follows from the following equalities, which make use of Observation 14.

$$\begin{aligned} g(S) &= \vec{v}^S \cdot \vec{w}' \\ &= \sum_i c_i \vec{v}^{S_i} \cdot \vec{w}' \\ &= \sum_i c_i \cdot g(S_i). \end{aligned}$$

Therefore, if we query $S_1, \ldots, S_d$ and learn that $g(S_i) = f(S_i)$, this fully determines $g(S) = f(S)$ for all $S \in \mathcal{M}_f$, and therefore establishes that the Perturbation Bound for $f(\cdot)$ is at most d. ◀

We now show the hard direction: Perturbation Bound is at least the Generalized Cut Dimension.

▶ **Lemma 17.** *Any (Symmetric) weight-based $f(\cdot)$ witnesses a (Symmetric, Non-Trivial) Perturbation Bound of d, the (Non-Trivial) Generalized Cut Dimension.*

Proof. Let $T_1, \ldots, T_{d-1}$ be any $d-1$ sets queried. Let X denote the subspace of vectors $\vec{y}$ which satisfy the linear equations $\vec{v}^{T_i} \cdot \vec{y} = 0$ for all $1 \leq i \leq d-1$. Observe that X has dimension at least $m - d + 1$. Let Y denote the subspace of vectors spanned by $\{\vec{v}^S\}_{S \in \mathcal{M}_f}$. Observe that Y has dimension d.

The dimension of X ($\geq m - d + 1$) and the dimension of Y (d) sum to $> m$. This means that there exists a non-zero vector, $\vec{z} \in X \cap Y$.[5] Because $\vec{z} \in X$, we can add $\varepsilon\vec{z}$ to $\vec{w}$ for any ε and arrive at a $\vec{w}'$ which is consistent with the queries so far. Because $\vec{z} \in Y$, we must have $z_i = 0$ whenever $w_i = 0$ (because all $\vec{v}^S$ have $v_i = 0$ when $w_i = 0$). Therefore, there exists a sufficiently small ε such that $\vec{w} + \varepsilon\vec{z}$ results in a $g(\cdot)$ which is consistent with all $d-1$ queries so far, and is in $\mathcal{G}(f)$, and also $\vec{w} - \varepsilon\vec{z}$ results in such a $g(\cdot)$ as well.

Consider now $\vec{z} \cdot \vec{v}^S$ for any $S \in \mathcal{M}_f$. If $\vec{z} \cdot \vec{v}^S > 0$, then when $\vec{w}' := \vec{w} - \varepsilon\vec{z}$, we have $g(S) < f(S)$, and therefore $\min_S\{g(S)\} < \min_S\{f(S)\}$, meaning that we have found the desired Perturbation Bound $g(\cdot)$ for these $d-1$ queries. If $\vec{z} \cdot \vec{v}^S < 0$ we can instead use $\vec{w}' := \vec{w} + \varepsilon\vec{z}$. So if these $d-1$ queries have no witness, it must be that $\vec{z} \cdot \vec{v}^S = 0$ for all $S \in \mathcal{M}_f$. In particular that this holds for the basis $\vec{v}^{S_1}, \ldots, \vec{v}^{S_d}$ of Y. To summarize this paragraph: unless $\vec{v}^{S_i} \cdot \vec{z} = 0$ for all i (note that these S_i were not necessarily queried), then these $d-1$ queries have a witness for the Perturbation Bound.

We will now establish that we can't have $\vec{z} \cdot \vec{v}^{S_i} = 0$ for all i, which will establish that in fact there is a witness for these $d-1$ queries (and all $d-1$ queries, since they were arbitrary). Consider that because $\vec{z} \in Y$, we can write $\vec{z} = \sum_i \beta_i \vec{v}^{S_i}$ for some $\vec{\beta}$ which is not $\vec{0}$. If $\vec{z} \cdot \vec{v}^{S_i} = 0$ for all i we have

$$\sum_i \beta_i \vec{v}^{S_i} \cdot \vec{v}^{S_j} = 0, \quad \forall j.$$

[5] To see why this is the case, write a basis $\mathcal{B}_X = \{v_1, v_2, \ldots, v_{m-q}\}$ of X and a basis $\mathcal{B}_Y = \{w_1, w_2, \ldots, w_d\}$ of Y. If $\mathcal{B}_X \cap \mathcal{B}_Y$ is non-empty then we are of course done, otherwise $\mathcal{B}_X \cup \mathcal{B}_Y$ is a set of strictly more than m vectors in $\mathbb{R}^m$. Hence they must be linearly dependent, implying we can write $\alpha_1 v_1 + \ldots \alpha_{m-q} v_{m-q} + \beta_1 w_1 + \ldots + \beta_d w_d = 0$ for some coefficients $\{\alpha_i\}$, $\{\beta_j\}$ that are not all zero. Then note that $\beta_1 w_1 + \ldots \beta_d w_d$ is clearly in both X and Y, and cannot be zero as otherwise all coefficients would be zero (because both $\mathcal{B}_X$ and $\mathcal{B}_Y$ are linearly independent).

Therefore, if we let A denote the $d \times m$ matrix whose rows are the vectors $\vec{v}^{S_i}$, we get that:

$$(A \cdot A^T) \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_d \end{bmatrix} = 0$$

Because $\vec{v}^{S_1}, \ldots, \vec{v}^{S_d}$ form a basis for Y we know A, A^T, and $A \cdot A^T$ have rank d.[6] But $\vec{\beta}$ is a non-zero vector in the kernel of the $d \times d$ matrix $A \cdot A^T$, which is a contradiction. Therefore, we must have $\vec{z} \cdot \vec{v}^{S_i} \neq 0$ for some i, implying that there exists the desired $g(\cdot)$ for any set of $d-1$ queries, and the Perturbation Bound is hence at least d. ◀

The proof of the theorem now follows directly from Lemmas 16 and 17. ◀

Theorem 15 lets us now restrict attention to the study of generalized cut dimension if we aim to prove lower bounds through the canonical perturbation approach. The subsequent sections establish that this is fruitful for symmetric, non-trivial SFM, but not for standard SFM.

3.3 An Undirected Graph with Global Cut Dimension $3n/2 - 2$

In this section, we provide an explicit undirected graph G on n vertices which has global cut dimension $3n/2 - 2$. This establishes the following theorem:

▶ **Theorem 18.** *The query complexity of Symmetric Non-Trivial SFM is at least* $3n/2 - 2$.

Proof. First, let n be odd and $n \geq 3$. Then $n = 2a + 1$ for some $a \geq 1$, so label the vertices of G as $\{v, w_1, w_1', w_2, w_2', \ldots, w_a, w_a'\}$. For edges (all undirected), put an edge between v and all other nodes, and an edge between w_i and w_i' for all i (and no other edges). It is easy to see that every cut in G has value at least 2, and that the mincuts indeed have value 2. The mincuts either separate w_i from the rest of the graph, w_i' from the rest of the graph, or $\{w_i, w_i'\}$ from the rest of the graph.

For any $i \in [a]$, let i_1, i_2, i_3 denote the three positions in indicator vectors corresponding to the three edges (v, w_i), (v, w_i'), (w_i, w_i'). Restricted to these positions, the indicator vectors for the three minimum cuts $\{w_i\}$, $\{w_i'\}$, $\{w_i, w_i'\}$ are $(1, 0, 1)$, $(0, 1, 1)$, and $(1, 1, 0)$, which span a subspace of dimension three. As these three cuts have zeroes for all other entries, the full vectors also span a subspace of dimension three. For each $i \in [a]$, the set of three indices referenced above are distinct, which means that taking all these indicator vectors together has rank $3a$.

As $n = 2a + 1$, $3a = 3(n-1)/2$. So the claim holds when n is odd. If n is even, use exactly the same construction on $n - 1$ nodes, and connect the remaining node to v with an edge of weight two. Now there is one additional mincut (separating the extra node from the rest), so the dimension is $3a + 1 = 3(n-2)/2 + 1 = 3n/2 - 2$. ◀

[6] A short proof of this is through the singular value decomposition (SVD) of A. Write $A = U\Sigma V$ where $U \in \mathbb{R}^{d \times d}$, $\Sigma \in \mathbb{R}^{d \times m}$, and $V \in \mathbb{R}^{m \times m}$ (where U and V satisfy $UU^T = I$ and $VV^T = I$, and Σ is diagonal with d non-zero entries). Note then that $A \cdot A^T = U\Sigma VV^T\Sigma^T U^T = U(\Sigma\Sigma^T)U^T$. As $\Sigma\Sigma^T$ is diagonal of rank d and $UU^T = I$, this is a singular value decomposition of $A \cdot A^T$, and directly implies that its rank is d.

3.4 Generalized Cut Dimension is at most $n + 1$

The previous section establishes that the Non-Trivial Symmetric Cut Dimension can be much larger than n, which leads to novel lower bounds. In this section, we establish that this approach will not yield novel lower bounds for standard SFM (and by Theorem 15, neither will the canonical perturbation argument for weight-based functions).

▶ **Theorem 19.** *The Generalized Cut Dimension of any weight-based function is at most* $n + 1$.

Proof. First, recall that for any submodular $f(\cdot)$, the set of minimizes $\mathcal{M}_f$ is closed under union and intersection.[7] For each $i \in [n]$, define $S_i := \cap_{S \in \mathcal{M}_f, i \in S} S$ (if there exists a minimizer containing i, otherwise let S_i be null). If S_i is not null, then $S_i \in \mathcal{M}_f$, because $\mathcal{M}_f$ is closed under intersection. Our goal will be to show that these S_i (and $\emptyset$, if $\emptyset \in \mathcal{M}_f$) span $\mathcal{M}_f$ through a sequence of lemmas.

Define the *base sets* $\mathcal{B}$ of $\mathcal{M}_f$ to be the set of all non-null S_i, together with $\emptyset$ (if $\emptyset \in \mathcal{M}_f$). It is clear that $\mathcal{B}$ has size at most $n + 1$. It is also the case that every minimizer $S \in \mathcal{M}_f$ can be written as the union of elements in $\mathcal{B}$:

▶ **Lemma 20.** *For all* $S \in \mathcal{M}_f$, $S = \cup_{i \in S} S_i$.

Proof. For any $i \in S$, we know that $i \in S_i$ which means that $S \subseteq \cup_{i \in S} S_i$. We need only show that for any $i \in S$, $S_i \subseteq S$. This is true by definition of S_i because S is a minimizer containing i. ◀

We next show that the base sets "cover" $\mathcal{M}_f$ in the following sense. Say that a set S is covered by $\mathcal{B}$ if either: (a) $S \in \mathcal{B}$, or (b) S can be written as the union of two sets which are covered by $\mathcal{B}$.

▶ **Lemma 21.** *Every set in* $\mathcal{M}_f$ *is covered by* $\mathcal{B}$.

Proof. Assume for contradiction that there is some $S \in \mathcal{M}_f$ which is not covered by $\mathcal{B}$. Take the S which minimizes $|S|$. Then clearly $S \notin \mathcal{B}$. So pick an arbitrary $i \in S$ and we can write $S = \cup_{j \in S} S_j = S_i \cup (\cup_{j \in S \setminus S_i} S_j)$. S_i is clearly non-empty, and also because $S \notin \mathcal{B}$, it is not equal to S. Similarly, $\cup_{j \notin S_i} S_j$ is non-empty (or else S_i would equal S), and is also not equal to S (because it does not contain i). Because $|\cup_{j \in S \setminus S_i} S_j| < |S|$, it is covered. We have just written S as the union of two covered sets, so therefore S is also covered, a contradiction. ◀

Now, we are ready for the last step. We will argue that for all $g \in \mathcal{G}(f)$, knowledge of $g(S), g(T)$, and $g(S \cap T)$ suffices to deduce $g(S \cup T)$. We will then deduce that knowledge of $g(S)$ for all $S \in \mathcal{B}$ suffices to deduce $g(S)$ for all $S \in \mathcal{M}_f$.

▶ **Lemma 22.** *Let* $S, T \in \mathcal{M}_f$. *Then* $\vec{v}^{S \cup T} = \vec{v}^S + \vec{v}^T - \vec{v}^{S \cap T}$.

Proof. Recall from the definition of weight-based that $h(S \cap T) \overline{\cup} h(S \cup T) \subseteq h(S) \overline{\cup} h(T)$. But recall also that $\sum_{i \in h(S \cap T) \overline{\cup} h(S \cup T)} w_i = \sum_{i \in h(S) \overline{\cup} h(T)} w_i$ because all of $S, T, S \cap T, S \cup T$ are minimizers. As all w_i are non-negative, this means that the only possible i which are counted fewer times on the LHS than the RHS must have $w_i = 0$. This immediately means that $\vec{v}^{S \cap T} + \vec{v}^{S \cup T} = \vec{v}^S + \vec{v}^T$. ◀

[7] To see this, recall that $f(S \cup T) + f(S \cap T) \leq f(S) + f(T)$. If both of the sets on the RHS are minimizers, both sets on the LHS must be as well.

▶ **Corollary 23.** *Every $S \in \mathcal{M}_f$ has $\vec{v}^S \in span(\{\vec{v}^T\}_{T\in\mathcal{B}})$.*

Proof. Lemma 22 establishes that if S, T, and $S \cap T$ are in span$(\{\vec{v}^T\}_{T\in\mathcal{B}})$, then so is $S \cup T$. Assume for contradiction that some $S \in \mathcal{M}_f$, $\vec{v}^S$ is not in span$(\{\vec{v}^T\}_{T\in\mathcal{B}})$, and take the one of minimal $|S|$. Then S is covered by $\mathcal{B}$ by Lemma 21, so we can write $S = A \cup B$, where $|A|, |B| < |S|$. $\vec{v}^A, \vec{v}^B$, and $\vec{v}^{A\cap B}$ are therefore in span$(\{\vec{v}^T\}_{T\in\mathcal{B}})$. By Lemma 22, so then is $\vec{v}^S$, a contradiction. ◀

The proof is now concluded: we have argued that $|\mathcal{B}| \leq n+1$, and Corollary 23 establishes that all of $\mathcal{M}_f$ is in the span of $\mathcal{B}$, so the Generalized Cut Dimension is at most $n+1$.

Importantly, observe that this proof fails to hold for the Generalized Non-Trivial Cut Dimension (it must, as we previously demonstrated an example with Generalized Non-Trivial Cut Dimension $3n/2-2$). The point of failure is that the set of Non-Trivial minimizers is not closed under intersection or union (if either the intersection is empty or the union is $[n]$). ◀

4 A $\binom{n}{2}$ Query Lower Bound for Non-Trivial SFM

In this section, we establish our lower bound for Non-Trivial SFM. The class of functions we consider will be the following:

▶ **Definition 24.** *A function $f(\cdot)$ is* cost-based *if there exists a cost function $c : 2^{[n]} \to \mathbb{R}_+$ with $c(T) = 0$ whenever $|T| \leq 1$ such that $f(S) = \sum_{i\in S} f(\{i\}) - \sum_{T\subseteq S} c(T)$.*

▶ **Proposition 25.** *Every cost-based function is submodular.*

Proof. We will establish that $f(S \cup \{i\}) - f(S) \leq f(T \cup \{i\}) - f(T)$ whenever $T \subseteq S$ and $i \notin S$. Observe that for any $X \subseteq [n]$ with $i \notin X$ we have $f(X \cup \{i\}) - f(X) = f(\{i\}) - \sum_{U\subseteq X} c(U \cup \{i\})$. $f(\{i\})$ is independent of X, and the second term is clearly at least as large for $X = S$ than $X = T$ (as $c(U) \geq 0$ for every U). Therefore, $f(\cdot)$ has diminishing marginal returns and is submodular. ◀

▶ **Theorem 26.** *The query complexity of Non-Trivial SFM is at least $\binom{n}{2}$.*

Proof. Consider the following $\binom{n}{2}+1$ cost functions. Define $c(\cdot)$ so that $c(S) = 0$ if $|S| \leq n-2$, $c([n] \setminus \{i\}) = n-1$ for all i, and $c([n]) = 2n$. Call the associated function $f(\cdot)$ where we set $f(\{i\}) = 1$ for all i. Then $f(S) = |S|$ when $|S| \leq n-2$, $f([n] \setminus \{i\}) = 0$, and $f([n]) = -n^2$.

For every $1 \leq i < j \leq n$ define $c_{ij}(\cdot)$ so that $c_{ij}(S) = 0$ if $|S| \leq n-3$. For sets of size $n-2$, set $c_{ij}([n] \setminus \{i, j\}) = n-1$, and $c_{ij}([n] \setminus \{k, \ell\}) = 0$ for all other $\{k, \ell\} \neq \{i, j\}$. For sets of size $n-1$, set $c_{ij}([n] \setminus \{i\}) = c_{ij}([n] \setminus \{j\}) = 0$, and $c_{ij}([n] \setminus \{k\}) = n-1$ for all $k \notin \{i, j\}$. Finally, set $c_{ij}([n]) = 3n-1$. Call the associated function $f_{ij}(\cdot)$ where we set $f_{ij}(\{\ell\}) = 1$ for all ℓ. Observe that $f_{ij}(S) = |S|$ when $|S| \leq n-3$, $f_{ij}(S) = n-2$ if $|S| = n-2$ and $S \neq [n] \setminus \{i, j\}$, $f_{ij}([n] \setminus \{i, j\}) = -1$, $f_{ij}([n] \setminus \{\ell\}) = 0$ for all ℓ, and $f_{ij}([n]) = -n^2$.

Observe, importantly, that the non-trivial minimum of $f(\cdot)$ is 0, while the non-trivial minimum of $f_{ij}(\cdot)$ is -1 for all i, j. Observe also that $f(\cdot)$ and $f_{ij}(\cdot)$ *differ only on their evaluation for* $[n] \setminus \{i, j\}$. Therefore, an adversary could answer any query S with $f(S)$. If the algorithm terminates with fewer than $\binom{n}{2}$ queries, then there is some $[n] \setminus \{i, j\}$ that has not been queried. Therefore, the adversary is free to decide that the function is either $f(\cdot)$ or $f_{ij}(\cdot)$. As the value of the minima for these two functions are distinct, the algorithm cannot be correct. Therefore, any correct algorithm for Non-Trivial SFM must make at least $\binom{n}{2}$ queries. ◀

5 Conclusions and Open Questions

We establish the first query lower bounds exceeding n for SFM ($2n$), Non-Trivial Symmetric SFM ($3n/2 - 2$), and Non-Trivial SFM ($\binom{n}{2}$). Our asymmetric lower bounds are from direct constructions. Our symmetric lower bound arises from the novel *cut dimension*.

Our work leaves open a clear direction for future work: what is the maximum possible Global Cut Dimension for an undirected graph? Or more generally, what is the maximum possible Non-Trivial Symmetric Generalized Cut Dimension of a weight-based function?

It is also of course generally important to further improve query complexity lower bounds for SFM variants (and also develop better algorithms).

References

1 Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast Approximate Energy Minimization via Graph Cuts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(11):1222–1239, 2001. doi:10.1109/34.969114.

2 Deeparnab Chakrabarty, Yin Tat Lee, Aaron Sidford, and Sam Chiu-wai Wong. Subquadratic Submodular Function Minimization. *CoRR*, abs/1610.09800, 2016. arXiv:1610.09800.

3 William H. Cunningham. On submodular function minimization. *Combinatorica*, 5(3):185–192, 1985. doi:10.1007/BF02579361.

4 Lisa Fleischer and Satoru Iwata. Improved algorithms for submodular function minimization and submodular flow. In *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing, May 21-23, 2000, Portland, OR, USA*, pages 107–116, 2000. doi:10.1145/335305.335318.

5 Martin Grötschel, László Lovász, and Alexander Schrijver. The Ellipsoid Method and its Consequences in Combinatorial Optimization. *Combinatorica*, 1(2):169–197, 1981.

6 Nicholas J. A. Harvey. *Matchings, matroids and submodular functions.* PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 2008. URL: http://hdl.handle.net/1721.1/44416.

7 Satoru Iwata. A Faster Scaling Algorithm for Minimizing Submodular Functions. *SIAM J. Comput.*, 32(4):833–840, 2003. doi:10.1137/S0097539701397813.

8 Satoru Iwata, Lisa Fleischer, and Satoru Fujishige. A combinatorial strongly polynomial algorithm for minimizing submodular functions. *J. ACM*, 48(4):761–777, 2001. doi:10.1145/502090.502096.

9 Satoru Iwata and James B. Orlin. A simple combinatorial algorithm for submodular function minimization. In *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2009, New York, NY, USA, January 4-6, 2009*, pages 1230–1237, 2009. URL: http://dl.acm.org/citation.cfm?id=1496770.1496903.

10 Leonid G. Khachiyan. A Polynomial Algorithm in Linear Programming. *Soviet Mathematics Doklady*, 20(1):191–194, 1979.

11 Pushmeet Kohli, M. Pawan Kumar, and Philip H. S. Torr. P3 & Beyond: Move Making Algorithms for Solving Higher Order Functions. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(9):1645–1656, 2009. doi:10.1109/TPAMI.2008.217.

12 Pushmeet Kohli and Philip HS Torr. Dynamic graph cuts and their applications in computer vision. In *Computer Vision*, pages 51–108. Springer, 2010.

13 Yin Tat Lee, Aaron Sidford, and Sam Chiu-wai Wong. A Faster Cutting Plane Method and its Implications for Combinatorial and Convex Optimization. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 1049–1065, 2015. doi:10.1109/FOCS.2015.68.

14 Hui Lin and Jeff A. Bilmes. Optimal Selection of Limited Vocabulary Speech Corpora. In *INTERSPEECH 2011, 12th Annual Conference of the International Speech Communication Association, Florence, Italy, August 27-31, 2011*, pages 1489–1492, 2011. URL: http://www.isca-speech.org/archive/interspeech_2011/i11_1489.html.

15 James B. Orlin. A Faster Strongly Polynomial Time Algorithm for Submodular Function Minimization. In *Integer Programming and Combinatorial Optimization, 12th International IPCO Conference, Ithaca, NY, USA, June 25-27, 2007, Proceedings*, pages 240–251, 2007. doi:10.1007/978-3-540-72792-7_19.

16 Maurice Queyranne. Minimizing symmetric submodular functions. *Math. Program.*, 82:3–12, 1998. doi:10.1007/BF01585863.

17 Aviad Rubinstein, Tselil Schramm, and S. Matthew Weinberg. Computing Exact Minimum Cuts Without Knowing the Graph. In *9th Innovations in Theoretical Computer Science Conference, ITCS 2018, January 11-14, 2018, Cambridge, MA, USA*, pages 39:1–39:16, 2018. doi:10.4230/LIPIcs.ITCS.2018.39.

18 Alexander Schrijver. A Combinatorial Algorithm Minimizing Submodular Functions in Strongly Polynomial Time. *J. Comb. Theory, Ser. B*, 80(2):346–355, 2000. doi:10.1006/jctb.2000.1989.

19 Jens Vygen. A note on Schrijver's submodular function minimization algorithm. *J. Comb. Theory, Ser. B*, 88(2):399–402, 2003. doi:10.1016/S0095-8956(02)00047-3.

A Omitted Proofs from Section 2

We will make use of the following technical lemma:

▶ **Lemma 27.** *If $R_i \subseteq S$, then $f^{\vec{c}}_{\sigma}(S) \leq (|S| - i) \cdot (n + 2 - i)$.*

Proof. If $S = R_{j(S)}$, then $f^{\vec{c}}_{\sigma}(S) = -c_j \leq 0 \leq (|S| - i) \cdot (n + 2 - i)$, as desired. Otherwise because $j(S) \geq i$ (by definition of $j(S)$), we know $|S| - j(S) \leq |S| - i$ and $n + 2 - j(S) \leq n + 2 - i$, which implies that $f^{\vec{c}}_{\sigma}(S) \leq (|S| - i) \cdot (n + 2 - i)$ as desired. ◀

Proof of Lemma 6. Let X, Y be any two subsets of $[n]$; we will show that

$$f^{\vec{c}}_{\sigma}(X) + f^{\vec{c}}_{\sigma}(Y) \geq f^{\vec{c}}_{\sigma}(X \cup Y) + f^{\vec{c}}_{\sigma}(X \cap Y).$$

Note that if $X \subseteq Y$, the inequality is trivially satisfied, as $X \cap Y = X$ and $X \cup Y = Y$; the inequality is also trivially satisfied if $Y \subseteq X$. Hence, we will assume that neither set is contained in the other; note that this means neither set could equal $\emptyset$ or $[n]$. From here we consider two separate cases.

In the first case, assume neither X nor Y is an important set. Let R_i be the largest important set that is a subset of X and R_j be the largest important set that is a subset of Y. Without loss of generality, let's assume that $i \geq j$. Let $A := (X \setminus R_i) \setminus (Y \setminus R_i)$, $B := (Y \setminus R_i) \setminus (X \setminus R_i)$, $C := (X \setminus R_i) \cap (Y \setminus R_i)$, and $D := Y \cap (R_i \setminus R_j)$ and for convenience define $a := |A|$, $b := |B|$, $c := |C|$, and $d := |D|$. From the definition of f we have that

$$f^{\vec{c}}_{\sigma}(X) = (a + c) \cdot (n + 2 - i)$$

and

$$f^{\vec{c}}_{\sigma}(Y) = (b + c + d) \cdot (n + 2 - j).$$

First, we prove the inequality when $i = j$. In this case, $D = \emptyset$, but A and B are both non-empty (as otherwise either $X \subseteq Y$ or $Y \subseteq X$). As $X \cap Y$ contains R_i we have

$$f^{\vec{c}}_{\sigma}(X \cap Y) \leq c(n + 2 - i)$$

from Lemma A.1. As $X \cup Y$ contains R_i we similarly have

$$f^{\vec{c}}_{\sigma}(X \cup Y) \leq (a + b + c) \cdot (n + 2 - i).$$

Hence,

$$\begin{aligned} f^{\vec{c}}_{\sigma}(X \cup Y) + f^{\vec{c}}_{\sigma}(X \cap Y) &\leq (a+b+2c)(n+2-i) \\ &= (a+c)(n+2-i) + (b+c)(n+2-i) \\ &= f^{\vec{c}}_{\sigma}(X) + f^{\vec{c}}_{\sigma}(Y) \end{aligned}$$

which proves the inequality as $d = 0$. We'll next prove the inequality assuming $i > j$. In that case, we again have that R_j is a subset of $X \cap Y$ so

$$f^{\vec{c}}_{\sigma}(X \cap Y) \leq (c+d)(n+2-j)$$

by Lemma A.1. Similarly, R_i is a subset of $X \cup Y$ so

$$f^{\vec{c}}_{\sigma}(X \cup Y) \leq (a+b+c)(n+2-i).$$

Hence it is sufficient to prove

$$(a+c) \cdot (n+2-i) + (b+c+d) \cdot (n+2-j) \geq (c+d)(n+2-j) + (a+b+c)(n+2-i)$$

which reduces to $b(n+2-j) \geq b(n+2-i)$, which is true as $b \geq 0$ and $i > j$. Hence we have completed our analysis for the first case.

Our second case is when X is an important set and Y is not. Say $X = R_i$ for some $i \geq 1$ and let R_j be the largest important set contained in Y. Clearly, $i > j$ (as otherwise we would have $X \subseteq Y$). Let $A := Y \setminus R_i$, $B := (Y \setminus R_j) \cap R_i$, and for convenience define $a := |A|$ and $b := |B|$. Note that $a > 0$ (as otherwise $Y \subseteq X$) and that there are exactly $a + b$ elements of Y not in R_j. Because $X \cap Y$ contains R_j but not R_{j+1} we know

$$f^{\vec{c}}_{\sigma}(X \cap Y) = b \cdot (n+2-j).$$

Also, as $X \cup Y$ contains R_i we know that

$$f^{\vec{c}}_{\sigma}(X \cup Y) \leq a \cdot (n+2-i).$$

Hence it is sufficient to prove

$$-c_i + (a+b) \cdot (n+2-j) \geq b \cdot (n+2-j) + a \cdot (n+2-i).$$

This inequality is equivalent to $-c_i + a(i-j) \geq 0$, which is true since $a \geq 1, i - j \geq 1$ and $c_i \leq 1$, hence concluding the second case.

By symmetry, we now also have the same conclusion if Y is an important set and X is not. Moreover, we need not consider the case where both are important sets as then one must be a subset of the other. Hence we have the result. ◀

B On Cut Queries in Directed Graphs

In this section, we examine the limits of cut queries when learning the edges of a graph. This appendix is not directly relevant to our main results, but may be of interest for future work.

▷ Claim 28. A directed weighted graph can be learned via cut queries up to directed cycles. That is, with cut queries one can learn a graph G' that is equivalent to the true graph G up to adding/deleting directed cycles. Moreover, no set of queries can determine the weight of a directed cycle.

Proof. We first work over unweighted graphs and show that a graph can be learned up to the direction of directed cycles. Let $G(V, E)$ be a directed, unweighted graph. First, we note that for any two vertices u and v in V, we can learn if

- both edges (u, v) and (v, u) exist,
- exactly one of the edges (u, v) and (v, u) exist (but not which one),
- or neither of these edges exist.

To see this, consider creating a new function $f'(\cdot)$ which outputs $f(S) + f(V \setminus S)$. Then $f'(\cdot)$ corresponds to an undirected *weighted* graph G' where the weight between two nodes is zero, one, or two (and equal to the number of edges between them in G). As G' is undirected, we can learn G' exactly using cut queries [17].[8]

Moreover, for every vertex u, by querying $\{u\}$ and $V \setminus \{u\}$, we know the in degree and the out degree of u.

Now, suppose two graphs G and G' both satisfy all the queries made so far. We claim that G can be converted to G' by flipping the direction of certain cycles. Let $e = (u, v)$ be an edge in G that does not exist in G'. We know that (v, u) must be an edge in G'. Suppose we flip edge (v, u) in G'. Now, the in degree of u in G' is one less than the in degree of u in G', while the out degree of u in G' is one more than that of G. Hence, there is an edge (u, v') in G', and an edge (v', u) in G. We flip this edge in G'. Continuing this way, we flip all edges in a path, until we reach v. If G' is the same as G, we are done, else, we pick another edge and repeat this procedure. Hence, G' and G are the same, up to the directions of directed cycles.

Moreover, for any cut queried, every directed cycle either does not contribute anything to the cut or adds exactly 1 to the cut (irrespective of the direction). Hence, the direction of cycles in a directed graph cannot be learned by cut queries.

This argument can be extended to weighted graphs as well. For weighted graphs, we can learn the sum of the weights of edges (u, v) and (v, u) for all edges, as well as the in degree and out degree of every vertex.

Suppose we have two graphs G and G' which satisfy all the queries made to learn the above. Similar to the unweighted case, we claim that G' can be converted to G by changing the weights of certain directed cycles. Let (u, v) be an edge which has weight w in G and w' in G'. We change the weight of (u, v) in G' to w, and add $w' - w$ to the weight of (v, u) in G'. Now, the in degree of u has increased by $w' - w$. Since the in degree of u is the same in both G and G' initially, this implies that there are edges incident on u in G whose weights differ from those in G' by exactly $w' - w$. We change the weights of each of those edges to match the ones in G, continuing till we complete each of these cycles. Hence, G and G' are identical up to the weights of certain directed cycles.

Note that each time we decrease the weight of a cycle in one direction by α, we increase the weight of the cycle in the other direction by exactly α. That is, the sum of the weights of the cycle in both directions remains constant. Let's assume that a cut query cuts k edges of this cycle. This implies that the cut query also cuts k edges of the cycle in the opposite direction. Hence, we can only determine the sum of the weights of these two cycles, irrespective of the number of queries made. ◁

▷ Claim 29. For a weighted undirected graph, when making s-t cut queries, the weight of edge (s, u) cannot be learned for any vertex u. Similarly, the weight of edge (u, t) cannot be learned for any vertex u.

[8] To see this, observe that $f'(\{u\}) + f'(\{v\}) - f'(\{u, v\})$ is exactly twice the weight of the edge between u and v in G'.

Proof. We first note that we can always learn the weight of edges of the form (u, v), with $u, v \neq s \neq t$. This can be done by querying $\{s\}$, $\{s, u\}$, $\{s, v\}$ and $\{s, u, v\}$. Hence, we can learn all edges except for those of the form (s, u) and (u, t) (because these queries are redundant or invalid in that case). After learning these weights, every query $S \cup \{s\}$ can be viewed as the sum of n weights (the weights of edges from S to t, and the weights of edges from $V \setminus S$ to s). Let us denote the weight of edge (s, u) as w_u, and the weight of edge (u, t) as w'_u. Every query $S \cup \{s\}$ can be written as a linear equation

$$\sum_{u \in S} w'_u + \sum_{u \in V \setminus S} w_u = c_S$$

Let α_S denote a $2n$ dimensional vector with the coefficients of the above equation. Let $\mathbf{w}$ be a $2n$ dimensional vector with w_u and the w'_u, for all u. The above equation can be written as

$$\langle \alpha_S, \mathbf{w} \rangle = c_S$$

Let us consider the subspace Π spanned by $\{\alpha_S \colon S \subseteq V \setminus \{s, t\}\}$. Let e_u denote the vector with a 1 in the position of edge (s, u) and zeros elsewhere. If $e_u \in \Pi$, we can compute the value of w_u. We show that $e_u \notin \Pi$, for all u.

To show this, it is enough to describe a vector in the kernel of Π, whose dot product with e_u is non-zero. Consider the vector β with 1 in the position of edges (s, u) and (u, t), and $-\frac{1}{n}$ elsewhere.

$$\langle e_u, \beta \rangle = 1$$

However, for any S,

$$\langle \alpha_S, \beta \rangle = 0$$

Hence, we cannot compute the weight of edge (s, u) for any vertex u. The same argument can also be made for the edge (u, t). ◁

Computation-Aware Data Aggregation

Bernhard Haeupler
Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA
haeupler@cs.cmu.edu

D. Ellis Hershkowitz
Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA
dhershko@cs.cmu.edu

Anson Kahng
Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA
akahng@cs.cmu.edu

Ariel D. Procaccia
Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA
arielpro@cs.cmu.edu

Abstract

Data aggregation is a fundamental primitive in distributed computing wherein a network computes a function of every nodes' input. However, while compute time is non-negligible in modern systems, standard models of distributed computing do not take compute time into account. Rather, most distributed models of computation only explicitly consider communication time.

In this paper, we introduce a model of distributed computation that considers *both* computation and communication so as to give a theoretical treatment of data aggregation. We study both the structure of and how to compute the fastest data aggregation schedule in this model. As our first result, we give a polynomial-time algorithm that computes the optimal schedule when the input network is a complete graph. Moreover, since one may want to aggregate data over a pre-existing network, we also study data aggregation scheduling on arbitrary graphs. We demonstrate that this problem on arbitrary graphs is hard to approximate within a multiplicative 1.5 factor. Finally, we give an $O(\log n \cdot \log \frac{\mathrm{OPT}}{t_m})$-approximation algorithm for this problem on arbitrary graphs, where n is the number of nodes and OPT is the length of the optimal schedule.

2012 ACM Subject Classification Theory of computation → Approximation algorithms analysis; Theory of computation → Scheduling algorithms

Keywords and phrases Data aggregation, distributed algorithm scheduling, approximation algorithms

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.65

Funding *Bernhard Haeupler*: Supported in part by NSF grants CCF-1527110, CCF-1618280, CCF-1814603, CCF-1910588, NSF CAREER award CCF-1750808 and a Sloan Research Fellowship.
D. Ellis Hershkowitz: Supported in part by NSF grants CCF-1527110, CCF-1618280, CCF-1814603, CCF-1910588, NSF CAREER award CCF-1750808 and a Sloan Research Fellowship.
Anson Kahng: Supported in part by NSF grants IIS-1350598, IIS-1714140, CCF-1525932, and CCF-1733556; by ONR grants N00014-16-1-3075 and N00014-17-1-2428; and by a Sloan Research Fellowship and a Guggenheim Fellowship.
Ariel D. Procaccia: Supported in part by NSF grants IIS-1350598, IIS-1714140, CCF-1525932, and CCF-1733556; by ONR grants N00014-16-1-3075 and N00014-17-1-2428; and by a Sloan Research Fellowship and a Guggenheim Fellowship.

1 Introduction

Distributed systems drive much of the modern computing revolution. However, these systems are only as powerful as the abstractions which enable programmers to make use of them. A key such abstraction is data aggregation, wherein a network computes a function of every node's input. For example, if every node stored an integer value, a programmer could run data aggregation to compute the sum or the largest value of every node in the network. Indeed, the well-studied and widely-used AllReduce abstraction [29, 16] consists of a data aggregation step followed by a broadcast step.

The utility of modern systems is their ability to perform massive computations and so, applications of data aggregation often consist of a function which is computationally-intensive to compute. A rigorous theoretical study of data aggregation, then, must take the cost of computation into account. At the same time, one cannot omit the cost of communication, as many applications of data aggregation operate on large datasets which take time to transmit over a network.

However, to our knowledge, all existing models of distributed computation – e.g., the CONGEST [28], SINR [4], (noisy) radio network [21, 9, 7, 8], congested clique [12], dual graph [6], store-and-forward [22, 31], LOCAL [23], and telephone broadcast models [30, 20, 17] – all only consider the cost of communication. Relatedly, while there has been significant applied research on communication-efficient data aggregation algorithms, there has been relatively little work that explicitly considers the cost of computation, and even less work that considers how to design a network to efficiently perform data aggregation [25, 19, 26, 27, 18]. In this way, there do not seem to exist theoretical results for efficient data aggregation scheduling algorithms that consider both the cost of communication and computation.

Thus, we aim to provide answers to two theoretical questions in settings where both computation and communication are non-negligible:

1. *How should one structure a network to efficiently perform data aggregation?*
2. *How can one coordinate a fixed network to efficiently perform data aggregation?*

1.1 Our Model and Problem

The Token Network Model

So as to give formal answers to these questions we introduce the following simple distributed model, the Token Network Model. A Token Network is given by an undirected graph $G = (V, E)$, $|V| = n$, with parameters $t_c, t_m \in \mathbb{N}$ which describe the time it takes nodes to do computation and communication, respectively.[1]

Time proceeds in synchronous rounds during which nodes can compute on or communicate atomic tokens. Specifically, in any given round a node is busy or not busy. If a node is not busy and has at least one token it can *communicate*: any node that does so is busy for the next t_m rounds, at the end of which it passes one of its tokens to a neighbor in G. If a node is not busy and has two or more tokens, it can *compute*: any node that does so is busy for the next t_c rounds, at the end of which it combines (a.k.a. aggregates) two of its tokens into a single new token.[2] At a high level, this means that communication takes t_m rounds and computation takes t_c rounds.

[1] We assume $t_c, t_m = \text{poly}(n)$ throughout this paper.

[2] Throughout this paper, we assume for ease of exposition that the smaller of t_c and t_m evenly divides the larger of t_c and t_m, or equivalently that either t_c or t_m is 1.

The Token Computation Problem

We use our Token Network model to give a formal treatment of data aggregation scheduling. In particular, we study the Token Computation problem. Given an input Token Network, an algorithm for the Token Computation problem must output a schedule S which directs each node when to compute and when and with whom to communicate. A schedule is valid if after the schedule is run on the input Token Network where every node begins with a single token, there is one remaining token in the entire network; i.e., there is one node that has aggregated all the information in the network. We use $|S|$ to notate the length of S – i.e., the number of rounds S takes – and measure the quality of an algorithm by the length of the schedule that it outputs. For completeness, we give a more technical and formal definition in Appendix A.

Discussion of Modeling Choices

Our Token Network model and the Token Computation problem are designed to formally capture the challenges of scheduling distributed computations where both computation and communication are at play. In particular, combining tokens can be understood as applying some commutative, associative function to the private input of all nodes in a network. For instance, summing up private inputs, taking a minimum of private inputs, or computing the intersection of input sets can all be cast as instances of the Token Computation problem. We assume that the computation time is the same for every operation and that the output of a computation is the same size as each of the inputs as a simplifying assumption. We allow nodes to receive information from multiple neighbors as this sort of communication is possible in practice.

Lastly, our model should be seen as a so-called "broadcast" model [21] of communication. In particular, it is easy to see that our assumption that a node can send its token to only a single neighbor rather than multiple copies of its token to multiple neighbors is without loss of generality: One can easily modify a schedule in which nodes send multiple copies to one of equal length in which a node only ever sends one token per round. An interesting followup question could be to consider our problem in a non-broadcast setting.

1.2 Our Results

We now give a high-level description of our technical results.

Optimal Algorithm on Complete Graphs (Section 3)

We begin by considering how to construct the optimal data aggregation schedule in the Token Network model for complete graphs for given values of t_c and t_m. The principal challenge in constructing such a schedule is formalizing how to optimally pipeline computation and communication and showing that any valid schedule needs at least as many rounds as one's constructed schedule. We overcome this challenge by showing how to modify a given optimal schedule into an efficiently computable one in a way that preserves its pipelining structure. Specifically, we show that one can always modify a valid optimal schedule into another valid optimal schedule with a well-behaved recursive form. We show that this well-behaved schedule can be computed in polynomial time. Stronger yet, we show that the edges over which communication takes place in this schedule induce a tree. It is important to emphasize that this result has implications beyond producing the optimal schedule for a complete graph; it shows one optimal way to *construct* a network for data aggregation (if one had the freedom to include any edge), thereby suggesting an answer to the first of our two research questions.

Hardness and Approximation on Arbitrary Graphs (Section 4)

We next consider the hardness of producing good schedules efficiently for arbitrary graphs and given values of t_c and t_m. We first show that no polynomial-time algorithm can produce a schedule of length within a multiplicative 1.5 factor of the optimal schedule unless P = NP. This result implies that one can only *coordinate* data aggregation over a pre-existing network so well.

Given that an approximation algorithm is the best one can hope for, we next give an algorithm which in polynomial time produces an approximately-optimal Token Computation schedule. Our algorithm is based on the simple observation that after $O(\log n)$ repetitions of pairing off nodes with tokens, having one node in each pair route a token to the other node in the pair, and then having every node compute, there will be a single token in the network. The difficulty in this approach lies in showing that one can route pairs of tokens in a way that is competitive with the length of the optimal schedule. We show that by considering the paths in G traced out by tokens sent by the optimal schedule, we can get a concrete hold on the optimal schedule. Specifically, we show that a polynomial-time algorithm based on our observation produces a valid schedule of length $O(\mathrm{OPT} \cdot \log n \cdot \log \frac{\mathrm{OPT}}{t_m})$ with high probability,[3] where OPT is the length of the optimal schedule. Using an easy bound on OPT, this can be roughly interpreted as an $O(\log^2 n)$-approximation algorithm. This result shows that data aggregation over a pre-existing network can be *coordinated* fairly well.

Furthermore, it is not hard to see that when $t_c = 0$ and $t_m > 0$, or when $t_c > 0$ and $t_m = 0$, our problem is trivially solvable in polynomial time. However, we show hardness for the case where $t_c, t_m > 0$, which gives a formal sense in which computation and communication cannot be considered in isolation, as assumed in previous models of distributed computation.

1.3 Terminology

For the remainder of this paper we use the following terminology. A token a *contains* token a' if $a = a'$ or a was created by combining two tokens, one of which contains a'. For shorthand we write $a' \in a$ to mean that a contains a'. A *singleton* token is a token that only contains itself; i.e., it is a token with which a node started. We let a_v be the singleton token with which vertex v starts and refer to a_v as v's singleton token. The *size* of a token is the number of singleton tokens it contains. Finally, let a_f be the last token of a valid schedule S; the *terminus* of S is the node at which a_f is formed by a computation.

2 Related Work

Cornejo et al. [10] study a form of data aggregation in networks that change over time, where the goal is to collect tokens at as few nodes as possible after a certain time. However, they do not consider computation time and they measure the quality of their solutions with respect to the optimal offline algorithm. Awerbuch et al. [5] consider computation and communication in a setting where jobs arrive online at nodes, and nodes can decide whether or not to complete the job or pass the job to a neighbor. However, they study the problem of job scheduling, not data aggregation, and, again, they approach the problem from the perspective of competitive analysis with respect to the optimal offline algorithm.

[3] Meaning at least $1 - 1/\mathrm{poly}(n)$ henceforth.

Another line of theoretical work related to our own is a line of work in centralized algorithms for scheduling information dissemination [30, 20, 17]. In this problem, an algorithm is given a graph and a model of distributed communication, and must output a schedule that instructs nodes how to communicate in order to spread some information. For instance, in one setting an algorithm must produce a schedule which, when run, broadcasts a message from one node to all other nodes in the graph. The fact that these problems consider spreading information is complementary to the way in which we consider consolidating it. However, we note that computation plays no role in these problems, in contrast to our Token Computation problem.

Of these prior models of communication, the model which is most similar to our own is the telephone broadcast model. In this model in each round a node can "call" another node to transmit information or receive a call from a single neighbor. Previous results have given a hardness of approximation of 3 [13] for broadcasting in this model and logarithmic as well as sublogarithmic approximation algorithms for broadcasting [14]. The two notable differences between this model and our own are (1) in our model nodes can receive information from multiple neighbors in a single round[4] and (2) again, in our model computation takes a non-negligible amount of time. Note, then, that even in the special case when $t_c = 0$, our model does not generalize the telephone broadcast model; as such we do not immediately inherit prior hardness results from the telephone broadcast problem. Furthermore, (1) and especially (2) preclude the possibility of an easy reduction from our problem to the telephone broadcast problem.

There is also a great deal of related applied work; additional details are in Appendix B.

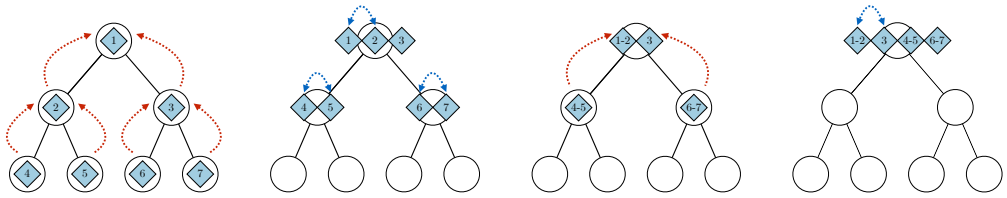
3 Optimal Algorithm for Complete Graphs

In this section we provide an optimal polynomial-time algorithm for the Token Computation problem on a complete graph. The schedule output by our algorithm ultimately only uses the edges of a particular tree, and so, although we reason about our algorithm in a fully connected graph, in reality our algorithm works equally well on said tree. This result, then, informs the design of an optimal network.

3.1 Binary Trees (Warmup)

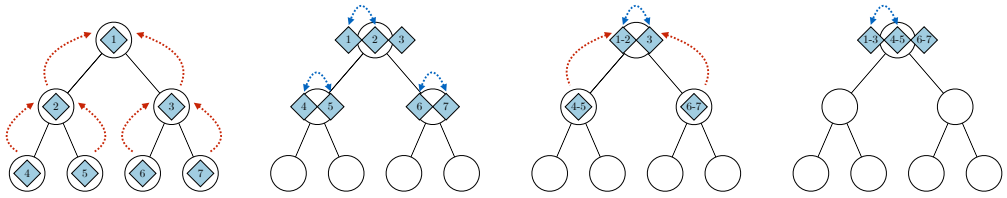
We build intuition by considering a natural solution to Token Computation on the complete graph: naïve aggregation on a rooted binary tree. In this schedule, nodes do computations and communications in lock-step. In particular, consider the schedule S which alternates the following two operations until only a single node with tokens remains on a fixed binary tree: (1) every non-root node that has a token sends its token to its parent in the binary tree; (2) every v with at least two tokens performs one computation. Once only one node has any tokens, that node performs computation until only a single token remains. After $\log n$ iterations of this schedule, the root of the binary tree is the only node with any tokens, and thereafter only performs computation for the remainder of S. However, S does not efficiently pipeline communication and computation: after each iteration of (1) and (2), the root of the tree gains an extra token. Therefore, after $\log n$ repetitions of this schedule, the root has $\log n$ tokens. In total, then, this schedule aggregates all tokens after essentially $\log n(t_c + t_m) + \log n \cdot t_c$ rounds. See Figure 1.

[4] See above for the justification of this assumption.



(a) Round 1. **(b)** Round 2. **(c)** Round 3. **(d)** Round 4.

Figure 1 The naïve aggregation schedule on a binary tree for $t_c = t_m = 1$ and $n = 7$ after 4 rounds. tokens are represented by blue diamonds; a red arrow from node u to node v means that u sends to v; and a double-ended blue arrow between two tokens a and b means that a and b are combined at the node. Notice that the root gains an extra token every 2 rounds.



(a) Round 1. **(b)** Round 2. **(c)** Round 3. **(d)** Round 4.

Figure 2 The aggregation schedule on a binary tree for $t_c = t_m = 1$ and $n = 7$ after 4 rounds where the root pipelines its computations. Again, tokens are represented by blue diamonds; a red arrow from node u to node v means that u sends to v; and a double-ended blue arrow between two tokens a and b means that a and b are combined at the node. Notice that the root will never have more than 3 tokens when this schedule is run.

For certain values of t_c and t_m, we can speed up naïve aggregation on the binary tree by pipelining the computations of the root with the communications of other nodes in the network. In particular, consider the schedule S' for a fixed binary tree for the case when $t_c = t_m$ in which every non-root node behaves exactly as it does in S but the root always computes. Since the root always computes in S', even as other nodes are sending, it does not build up a surplus of tokens as in S. Thus, this schedule aggregates all tokens after essentially $\log n(t_c + t_m)$ rounds when $t_c = t_m$, as shown in Figure 2.

However, as we will prove, binary trees are not optimal even when they pipeline computation at the root and $t_c = t_m$. In the remainder of this section, we generalize this pipelining intuition to arbitrary values of t_c and t_m and formalize how to show a schedule is optimal.

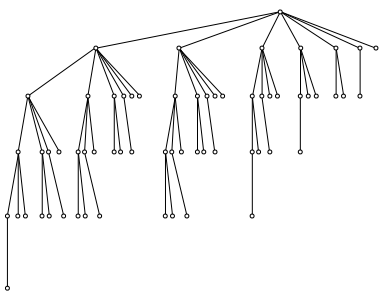


Figure 3 $T(16)$ for $t_c = 2$, $t_m = 1$.

3.2 Complete Graphs

We now describe our optimal polynomial-time algorithm for complete graphs. This algorithm produces a schedule which greedily aggregates on a particular tree, T_n^*. In order to describe this tree, we first introduce the tree $T(R, t_c, t_m)$. This tree can be thought of as the largest tree for which greedy aggregation aggregates all tokens in R rounds given computation cost t_c and communication cost t_m. We will overload notation and let $T(R)$ denote $T(R, t_c, t_m)$ for some fixed values of t_c and t_m. Let the root of a tree be the node in that tree with no parents. Also, given a tree T_1 with root r we define T_1 JOIN T_2 as T_1 but where r also has T_2 as an additional subtree. We define $T(R)$ as follows (see Figure 3 for an example):

$$T(R) \coloneqq \begin{cases} \text{A single leaf} & \text{if } R < t_m + t_c \\ T(R - t_c) \text{ JOIN } T(R - t_c - t_m) & \text{otherwise} \end{cases}$$

Since an input to the TOKEN COMPUTATION problem consists of n nodes, and not a desired number of rounds, we define $R^*(n, t_c, t_m)$ to be the minimum value such that $|T(R^*(n, t_c, t_m))| \geq n$. We again overload notation and let $R^*(n)$ denote $R^*(n, t_c, t_m)$. Formally,

$$R^*(n) \coloneqq \min\{R : |T(R)| \geq n\}.$$

We let T_n^* denote $T(R^*(n))$. For ease of presentation we assume that $|T_n^*| = n$.[5]

The schedule produced by our algorithm will simply perform greedy aggregation on T_n^*. We now formally define greedy aggregation and establish its runtime on the tree $T(R)$.

▶ **Definition 1** (Greedy Aggregation). Given an r-rooted tree, let the *greedy aggregation* schedule be defined as follows. In the first round, every node except for r sends its token to its parent. In subsequent rounds we do the following. If a node is not busy and has at least two tokens, it performs a computation. If a non-root node is not busy, has exactly one token, and has received a token from every child in previous rounds, it forwards its token to its parent.

▶ **Lemma 2.** *Greedy aggregation on $T(R)$ terminates in R rounds.*

Proof. We will show by induction on $k \geq 0$ that greedy aggregation results in the root of $T(k)$ having a token of size $|T(k)|$ after k rounds. The base cases of $k \in [0, t_m + t_c)$ are trivial, as nothing needs to be combined. For the inductive step, applying the inductive hypothesis and using the recursive structure of our graph tells us that the root of $T(k + t_c)$ has a token of size $|T(k)|$ at its root in k rounds, and the root of the child $T(k - t_m)$ has a token of size $|T(k - t_m)|$ at its root in $k - t_m$ rounds. Therefore, by the definition of greedy aggregation, the root of $T(k - t_m)$ sends its token of size $|T(k - t_m)|$ to the root of $T(k + t_c)$ at time $k - t_m$, which means the root of $T(k + t_c)$ can compute a token of size $|T(k - t_m)| + |T(k)| = |T(k + t_c)|$ by round $k + t_c$. ◀

To build intuition about how quickly T_n^* grows, see Figure 6 for an illustration of $|T_n^*|$ as a function of n for specific values of t_c and t_m. Furthermore, notice that $T(R)$ and T_n^* are constructed in such a way that greedy aggregation pipelines computation and communication. We can now formalize our optimal algorithm, which simply outputs the greedy aggregation schedule on T_n^*, as Algorithm 1. The following theorem is our main result for this section.

[5] If $|T_n^*| > n$, then we could always "hallucinate" extra nodes where appropriate.

Algorithm 1 OptComplete(t_c, t_m, n).

Input: t_c, t_m, n
Output: A schedule for Token Computation on K_n with parameters t_c and t_m
Arbitrarily embed T_n^* into K_n
return Greedy aggregation schedule on T_n^* embedded in K_n

▶ **Theorem 3.** *Given a complete graph K_n on n vertices and any $t_m, t_c \in \mathbb{Z}^+$,* OptComplete *optimally solves* Token Computation *on the* Token Network *(K_n, t_c, t_m) in polynomial time.*

To show that Theorem 3 holds, we first note that OptComplete trivially runs in polynomial time. Therefore, we focus on showing that greedy aggregation on T_n^* optimally solves the Token Computation problem on K_n. We demonstrate this claim by showing that, given R rounds, $|T(R)|$ is the size of the largest solvable graph. Specifically, we will let $N^*(R)$ be the size of the largest complete graph on which one can solve Token Computation in R rounds, and we will argue that $N^*(R)$ obeys the same recurrence as $|T(R)|$.

First notice that the base case of $N^*(R)$ is trivially 1.

▶ **Lemma 4.** *For $R \in \mathbb{Z}_0^+$ we have that $N^*(R) = 1$ for $R < t_c + t_m$.*

Proof. If $R < t_c + t_m$, there are not enough rounds to send and combine a token, and so the Token Computation problem can only be solved on a graph with one node. ◀

We now show that for the recursive case $N^*(R)$ is always at least as large as $N^*(R - t_c) + N^*(R - t_c - t_m)$, which is the recurrence that defines $|T(R)|$.

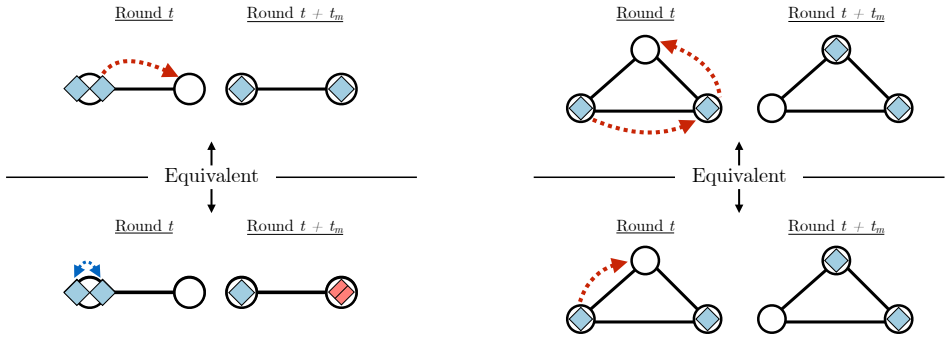
▶ **Lemma 5.** *For $R \in \mathbb{Z}_0^+$ we have that $N^*(R) \geq N^*(R-t_c)+N^*(R-t_c-t_m)$ for $R \geq t_c+t_m$.*

Proof. Suppose $R \geq t_c + t_m$. Let S_1 be the optimal schedule on the complete graph of $N^*(R-t_c)$ nodes with terminus v_{t1} and let S_2 be the optimal schedule on the complete graph of size $N^*(R-t_c-t_m)$ with corresponding terminus v_{t2}. Now consider the following solution on the complete graph of $N^*(R-t_c)+N^*(R-t_c-t_m)$ nodes. Run S_1 and S_2 in parallel on $N^*(R-t_c)$ and $N^*(R-t_c-t_m)$ nodes respectively, and once S_2 has completed, forward the token at v_{t2} to v_{t1} and, once it arrives, have v_{t1} perform one computation. This is a valid schedule which takes R rounds to solve Token Computation on $N^*(R-t_c)+N^*(R-t_c-t_m)$ nodes. Thus, we have that $N^*(R) \geq N^*(R-t_c)+N^*(R-t_c-t_m)$ for $R \geq t_c+t_m$. ◀

It remains to show that this bound on the recursion is tight. To do so, we case on whether $t_c \geq t_m$ or $t_c < t_m$. When $t_c \geq t_m$, we perform a straightforward case analysis to show that N^* follows the same recurrence as T_n^*. Specifically, we case on when the last token in the optimal schedule was created to show the following.

▶ **Lemma 6.** *When $t_c \geq t_m$ for $R \in \mathbb{Z}_0^+$ it holds that $N^*(R) = N^*(R-t_c)+N^*(R-t_c-t_m)$.*

Proof. Suppose that $R \geq t_c + t_m$. By Lemma 5, it is sufficient to show that $N^*(R) \leq N^*(R-t_c)+N^*(R-t_c-t_m)$. Consider the optimal solution given R rounds. The last action performed by any node must have been a computation that combines two tokens, a and b, at the terminus v_t because, in an optimal schedule, any further communication of the last token increases the length of the schedule. We now consider three cases.



(a) Combining insight. **(b)** Shortcutting insight.

Figure 4 An illustration of the shortcutting and combining insights. Here, tokens are denoted by blue diamonds, and hallucinated tokens are denoted by striped red diamonds. As before, a red arrow from node u to node v means that u sends to v, and a double-ended blue arrow between two tokens a and b means that a and b are combined at the node. Notice that which nodes have tokens and when nodes have tokens are the same under both modifications (though in the combining insight, a node is only hallucinating that it has a token).

- In the first case, a and b were both created at v_t. Because both of a or b could not have been created at time $R - t_c$, one of them must have been created at time $R - 2t_c$ at the latest. This means that $N^*(R) \leq N^*(R-t_c)+N^*(R-2t_c) \leq N^*(R-t_c)+N^*(R-t_c-t_m)$.
- In the second case, exactly one of a or b (without loss of generality, a) was created at v_t. This means that b must have been sent to v_t at latest at time $R - t_c - t_m$. It follows that $N^*(R) \leq N^*(R - t_c) + N^*(R - t_c - t_m)$.
- In the last case, neither a nor b was created at v_t. This means that both must have been sent to v_t at the latest at time $R - t_c - t_m$. We conclude that $N^*(R) \leq N^*(R - t_c - t_m) + N^*(R - t_c - t_m) \leq N^*(R - t_c) + N^*(R - t_c - t_m)$.

Thus, in all cases we have $N^*(R) \leq N^*(R - t_c) + N^*(R - t_c - t_m)$. ◀

We now consider the case in which communication is more expensive than computation, $t_c < t_m$. One might hope that the same case analysis used when $t_c \geq t_m$ would prove the desired result for when $t_c < t_m$. However, we must do significantly more work to show that $N^*(R) = N^*(R - t_c) + N^*(R - t_c - t_m)$ when $t_c < t_m$. We do this by establishing structure on the schedule which solves Token Computation on $K_{N^*(R)}$ in R rounds: we successively modify an optimal schedule in a way that does not affect its validity or length but which adds structure to the schedule.

Specifically, we leverage the following insights – illustrated in Figure 4 – to modify schedules. *Combining insight*: Suppose node v has two tokens in round t, a and b, and v sends a to node u in round t. Node v can just aggregate a and b, treat this aggregation as it treats b in the original schedule and u can just pretend that it receives a in round $t + t_m$. That is, u can "hallucinate" that it has token a. Note that this insight crucially leverages the fact that $t_c < t_m$, since otherwise the performed computation would not finish before round $t + t_m$. *Shortcutting insight*: Suppose node v sends a token to node u in round t and node u sends a token to node w in a round in $[t, t + t_m]$. Node v can "shortcut" node u and send to w directly and u can just not send.

Through modifications based on these insights we show that there exists an optimal schedule where the last node to perform a computation never communicates, and every computation performed by this node computes on the token with which this node started. This structure, in turn, allows us to establish the following lemma, which asserts that when $t_c < t_m$ we have that $N^*(R)$ and $|T(R)|$ follow the same recurrence.

▶ **Lemma 7.** *When $t_c < t_m$, for $R \in \mathbb{Z}_0^+$ it holds that $N^*(R) = N^*(R-t_c)+N^*(R-t_c-t_m)$.*

The proof of the lemma is relegated to Appendix D. We are now ready to prove the theorem.

Proof of Theorem 3. On a high level, we argue that the greedy aggregation schedule on $T(R)$ combines $N^*(R)$ nodes in R rounds and is therefore optimal. Combining Lemma 4, Lemma 6, and Lemma 7 we have the following recurrence on $N^*(R)$ for $R \in \mathbb{Z}_0^+$.

$$N^*(R) = \begin{cases} 1 & \text{if } R < t_c + t_m \\ N^*(R - t_c) + N^*(R - t_c - t_m) & \text{if } R \geq t_c + t_m \end{cases}$$

Notice that this is the recurrence which defines $|T(R)|$ so for $R \in \mathbb{Z}_0^+$ we have that $N^*(R) = |T(R)|$, and by Lemma 2, the greedy aggregation schedule on $T(R)$ terminates in R rounds.

Thus, the greedy aggregation schedule on $T(R)$ solves Token Computation on $K_{|T(R)|} = K_{N^*(R)}$ in R rounds, and therefore is an optimal solution for $K_{N^*(R)}$. Since T_n^* is the smallest $T(R)$ with at most n nodes, greedy aggregation on T_n^* is optimal for K_n and so OptComplete optimally solves Token Computation on K_n. Finally, the polynomial runtime is trivial. ◀

4 Hardness and Approximation for Arbitrary Graphs

We now consider the Token Computation problem on arbitrary graphs. Unlike in the case of complete graphs, the problem turns out to be computationally hard on arbitrary graphs. The challenge in demonstrating the hardness of Token Computation is that the optimal schedule for an arbitrary graph does not have a well-behaved structure. Our insight here is that by forcing a single node to do a great deal of computation we can impose structure on the optimal schedule in a way that makes it reflect the minimum dominating set of the graph. The following theorem formalizes this; its full proof is relegated to Appendix E.

▶ **Theorem 8.** Token Computation *cannot be approximated by a polynomial-time algorithm within $(1.5-\epsilon)$ for $\epsilon \geq \frac{1}{o(\log n)}$ unless $P = NP$.*

Therefore, our focus in this section is on designing an approximation algorithm. Specifically, we construct a polynomial-time algorithm, SolveTC, which produces a schedule that solves Token Computation on arbitrary graphs using at most $O(\log n \cdot \log \frac{\mathrm{OPT}}{t_m})$ multiplicatively more rounds than the optimal schedule, where OPT is the length of the optimal schedule. Define the diameter D of graph G as $\max_{v,u} d(u,v)$. Notice that OPT/t_m is at most $(n-1)t_c/t_m + D$ since $\mathrm{OPT} \leq (n-1)(t_c + D \cdot t_m)$: the schedule that picks a pair of nodes, routes one to the other then aggregates and repeats $n-1$ times is valid and takes $(n-1)(t_c + D \cdot t_m)$ rounds. Thus, our algorithm can roughly be understood as an $O(\log^2 n)$ approximation algorithm. Formally, our main result for this section is the following theorem whose lengthy proof we summarize in the rest of this section.

▶ **Theorem 9.** SolveTC *is a polynomial-time algorithm that gives an $O(\log n \cdot \log \frac{\mathrm{OPT}}{t_m})$-approximation for* Token Computation *with high probability.*

The rest of this section provides an overview of this theorem's lengthy proof. Our approximation algorithm, SolveTC, is given as Algorithm 2. SolveTC performs $O(\log n)$ repetitions of: designate some subset of nodes with tokens sinks and the rest of the nodes with tokens sources; route tokens at sources to sinks. If $t_c > t_m$, we will delay computations until tokens from sources arrive at sinks, and if $t_m \geq t_c$, we will immediately aggregate tokens that arrive at the same node.

4.1 Token Computation Extremes (Warmup)

Before moving on to a more technical overview of our algorithm, we build intuition by considering two extremes of Token Computation.

$t_m \ll t_c$

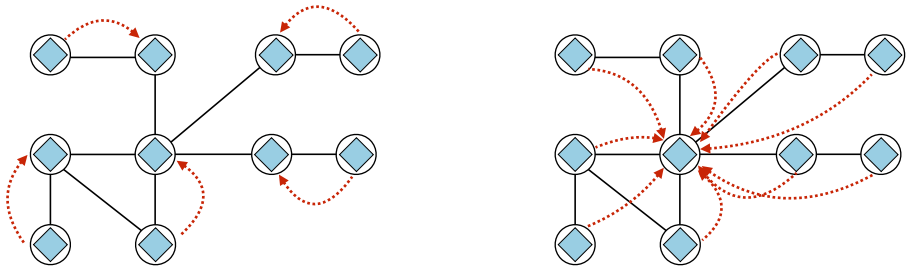
First, consider the case where $t_m \ll t_c$; that is, communication is very cheap compared to computation. As computation is the bottleneck here, we can achieve an essentially optimal schedule by parallelizing computation as much as possible. That is, consider a schedule consisting of $O(\log n)$ repetitions of: (1) each node with a token uniquely pairs off with another node with a token; (2) one node in each pair routes its token to the other node in its pair; (3) nodes that received a token perform one computation. This takes $O(t_c \cdot \log n)$ rounds to perform computations along with some amount of time to perform communications. But, any schedule takes at least $\Omega(t_c \cdot \log n)$ rounds, even if communication were free and computation were perfectly parallelized. Because the time to perform communications is negligible, this schedule is essentially optimal.

$t_c \ll t_m$

Now consider the case where $t_c \ll t_m$; that is, computation is very cheap compared to communication. In this setting, we can provide an essentially optimal schedule by minimizing the amount of communication that occurs. In particular, we pick a center c of the graph[6] and have all nodes send their tokens along the shortest path towards c. At any point during this schedule, it is always more time efficient for a node with multiple tokens to combine its tokens together before forwarding them since $t_c \ll t_m$. Thus, if at any point a node has multiple tokens, it combines these into one token and forwards the result towards c. Lastly, c aggregates all tokens it receives. This schedule takes $t_m \cdot r$ time to perform its communications, where r is the radius of the graph,[7] and some amount of time to perform its computations. However, because for every schedule there exists a token that must travel at least r hops, any schedule takes at least $\Omega(r \cdot t_m)$ rounds. Computations take a negligible amount of time since $t_c \ll t_m$, which means that this schedule is essentially optimal.

See Figure 5 for an illustration of these two schedules. Thus, in the case when $t_m \ll t_c$, we have that routing between pairs of nodes and delaying computations is essentially optimal, and in the case when $t_c \ll t_m$, we have that it is essentially optimal for nodes to greedily aggregate tokens before sending. These two observations will form the foundation of our approximation algorithm.

[6] The center of graph G is $\arg\min_v \max_u d(v,u)$ where $d(v,u)$ is the length of the shortest $u-v$ path.
[7] The radius of graph G is $\min_v \max_u d(v,u)$.



(a) $t_m \ll t_c$. **(b)** $t_c \ll t_m$.

Figure 5 An illustration of essentially optimal schedules for the extremes of the Token Computation problem. Dotted red arrows give the node towards which each node routes. In the case where $t_m \ll t_c$ one would repeat this sort of routing $O(\log n)$ times.

4.2 Approximation Algorithm

Recall that our approximation algorithm routes tokens from designated sources to designated sinks $O(\log n)$ times. Formally, the problem which our algorithm solves $O(\log n)$ times is as follows.

▶ **Definition 10** (Route and Compute Problem)**.** The input to the Route and Compute Problem consists of a set $U \subseteq V$ and a set of directed paths $\vec{\mathcal{P}_U} = \{\vec{P_u} : u \in U\}$ where: (1) $u \in U$ has a token and is the source of $\vec{P_u}$; (2) every sink of every path $\vec{P_u}$ has a token; (3) if u and t_u are the sources and sinks of $\vec{P_u} \in \vec{\mathcal{P}_U}$, respectively, then neither u nor t_u are endpoints of any $\vec{P_{u'}} \in \vec{\mathcal{P}_U}$ for $u' \neq u$. A solution of cost C is a schedule of length C which, when run, performs computations on a constant fraction of tokens belonging to nodes in U.

SolveTC repeatedly calls a subroutine, GetDirectedPaths, to get a set of paths for which it would like to solve the Route and Compute Problem. It then solves the Route and Compute Problem for these paths, using RoutePaths$_m$ if $t_c \leq t_m$ or RoutePaths$_c$ if $t_c > t_m$. Below we give an overview of these procedures. The proofs of the lemmas in this section, as well as further details regarding SolveTC, are relegated to Appendix F.

Algorithm 2 SolveTC.

Input: Token Computation instance given by graph $G = (V, E), t_c, t_m$
Output: A schedule for the input Token Computation problem
$W \leftarrow V$
for iteration $i \in O(\log n)$ **do**
  $\vec{\mathcal{P}_U} \leftarrow$ GetDirectedPaths(W, G)
  if $t_c > t_m$ **then** RoutePaths$_m(\vec{\mathcal{P}_U})$
  if $t_c \leq t_m$ **then** RoutePaths$_c(\vec{\mathcal{P}_U})$
  $W \leftarrow \{v : v$ has 1 token$\}$

4.2.1 Producing Paths on Which to Route

We now describe GetDirectedPaths. First, for a set of paths $\mathcal{P}$, we define the *vertex congestion* of $\mathcal{P}$ as $\text{con}(\mathcal{P}) = \max_v \sum_{P \in \mathcal{P}}(\#$ occurences of $v \in P)$, and the *dilation* of $\mathcal{P}$ as $\max_{P \in \mathcal{P}} |P|$.

Given that nodes in $W \subseteq V$ have tokens, GetDirectedPaths solves a flow LP which has a flow for each $w \in W$ whose sinks are $w' \in W$ such that $w' \neq w$. The objective of this flow LP is the vertex congestion. The flow for each $w \in W$ defines a probability

distribution over (undirected) paths with endpoints w and w' where $w' \neq w$ and $w' \in W$. Given these probability distributions, we repeatedly sample paths by taking random walks proportional to LP values of edges until we produce a set of paths – one for each $w \in W$ – with low vertex congestion. Lastly, given our undirected paths, we apply another subroutine to direct our paths and fix some subset of nodes $U \subset W$ as sources such that $|U|$ is within a constant fraction of $|W|$. The key property of the LP we use is that it has an optimal vertex congestion comparable to OPT, the length of the optimal Token Computation schedule. Using this fact and several additional lemmas we can prove the following properties of GetDirectedPaths.

▶ **Lemma 11.** *Given $W \subseteq V$,* GetDirectedPaths *is a randomized polynomial-time algorithm that returns a set of directed paths, $\vec{\mathcal{P}}_U = \{P_u : u \in U\}$ for $U \subseteq W$, such that with high probability at least $1/12$ of nodes in W are sources of paths in $\vec{\mathcal{P}}_U$ each with a unique sink in W. Moreover,*

$$con(\vec{\mathcal{P}}_U) \leq O\left(\frac{\mathrm{OPT}}{\min(t_c, t_m)} \log \frac{\mathrm{OPT}}{t_m}\right) \text{ and } dil(\vec{\mathcal{P}}_U) \leq \frac{8\mathrm{OPT}}{t_m}.$$

4.2.2 Routing Along Produced Paths

We now specify how we route along the paths produced by GetDirectedPaths. If $t_c > t_m$, we run RoutePaths$_m$ to delay computations until tokens from sources arrive at sinks, and if $t_m \geq t_c$, we run RoutePaths$_c$ to immediately aggregate tokens that arrive at the same node.

Case of $t_c > t_m$

RoutePaths$_m$ adapts the routing algorithm of Leighton et al. [22] – which was simplified by Rothvoß [31] – to efficiently route from sources to sinks.[8] We let OPTRoute be this adaptation of the algorithm of Leighton et al. [22].

▶ **Lemma 12.** *Given a set of directed paths $\vec{\mathcal{P}}_U$ with some subset of endpoints of paths in $\vec{\mathcal{P}}_U$ designated sources and the rest of the endpoints designated sinks,* OPTRoute *is a randomized polynomial-time algorithm that w.h.p. produces a* Token Network *schedule that sends from all sources to sinks in $O(con(\vec{\mathcal{P}}_U) + dil(\vec{\mathcal{P}}_U))$.*

Given $\vec{\mathcal{P}}_U$, RoutePaths$_m$ is as follows. Run OPTRoute and then perform a single computation. As mentioned earlier, this algorithm delays computation until all tokens have been routed.

▶ **Lemma 13.** RoutePaths$_m$ *is a polynomial-time algorithm that, given $\vec{\mathcal{P}}_U$, solves the* Route and Compute *Problem w.h.p. using $O(t_m(con(\vec{\mathcal{P}}_U) + dil(\vec{\mathcal{P}}_U)) + t_c)$ rounds.*

Case of $t_c \leq t_m$

Given directed paths $\vec{\mathcal{P}}_U$, RoutePaths$_c$ is as follows. Initially, every sink is *asleep* and every other node is *awake*. For $O(\mathrm{dil}(\vec{\mathcal{P}}_U) \cdot t_m)$ rounds we repeat the following: if a node is not currently sending and has exactly one token then it forwards this token along its path; if

[8] Our approach for the case when $t_c > t_m$ can be simplified using techniques from Srinivasan and Teo [32]. In fact, using their techniques we can even shave the $\frac{\log \mathrm{OPT}}{t_c}$ factor in our approximation. However, because these techniques do not take computation into account, they do not readily extend to the case when $t_c \leq t_m$. Thus, for the sake of a unified exposition, we omit the adaptation of their results.

a node is not currently sending and has two or more tokens then it sleeps for the remainder of the $O(\mathrm{dil}(\vec{\mathcal{P}_U}) \cdot t_m)$ rounds. Lastly, every node combines any tokens it has for $t_c \cdot \mathrm{con}(\vec{\mathcal{P}_U})$ rounds.

▶ **Lemma 14.** RoutePaths$_c$ *is a polynomial-time algorithm that, given* $\vec{\mathcal{P}_U}$*, solves the* Route and Compute *Problem w.h.p. using* $O(t_c \cdot con(\vec{\mathcal{P}_U}) + t_m \cdot dil(\vec{\mathcal{P}_U}))$ *rounds.*

By leveraging the foregoing results, we can prove Theorem 9; see Appendix F.3 for details.

5 Future Work

There are many promising directions for future work. First, as Section 4.1 illustrates, the extremes of our problem – when $t_c \ll t_m$ and when $t_m \ll t_c$ – are trivial to solve. However, our hardness reduction demonstrates that for $t_c = 1$ and t_m in a specific range, our problem is hard to approximate. Determining precisely what values of t_m and t_c make our problem hard to approximate is open.

Next, it is not always the case that there exists a centralized coordinator to produce a schedule. We hope to give an analysis of our problem in a distributed setting as no past work in this setting takes computation into account. Even more broadly, we hope to analyze formal models of distributed computation in which nodes are not assumed to have unbounded computational resources and computation takes a non-trivial amount of time.

We also note that there is a gap between our hardness of approximation and the approximation guarantee of our algorithm. The best possible approximation, then, is to be decided by future work.

Furthermore, we are interested in studying technical challenges similar to those studied in approximation algorithms for network design. For instance, we are interested in the problem in which each edge has a cost and one must build a network subject to budget constraints which has as efficient a Token Computation schedule as possible.

Lastly, there are many natural generalizations of our problem. For instance, consider the problem in which nodes can aggregate an arbitrary number of tokens together, but the time to aggregate multiple tokens is, e.g., a concave function of the number of tokens aggregated. These new directions offer not only compelling theoretical challenges but may be of practical interest.

References

1. I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. A survey on sensor networks. *IEEE Communications Magazine*, 40(8):102–114, 2002.
2. I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: a survey. *Computer Networks*, 38(4):393–422, 2002.
3. G. Anastasi, M. Conti, M. Di Francesco, and A. Passarella. Energy conservation in wireless sensor networks: a survey. *Ad Hoc Networks*, 7(3):537–568, 2009.
4. Matthew Andrews and Michael Dinitz. Maximizing capacity in arbitrary wireless networks in the SINR model: Complexity and game theory. In *IEEE INFOCOM 2009*, pages 1332–1340. IEEE, 2009.
5. Baruch Awerbuch, Shay Kutten, and David Peleg. Competitive distributed job scheduling. In *Proceedings of the twenty-fourth annual ACM symposium on Theory of computing*, pages 571–580. ACM, 1992.
6. Keren Censor-Hillel, Seth Gilbert, Fabian Kuhn, Nancy Lynch, and Calvin Newport. Structuring unreliable radio networks. *Distributed Computing*, 27(1):1–19, 2014.

7 Keren Censor-Hillel, Bernhard Haeupler, D Ellis Hershkowitz, and Goran Zuzic. Broadcasting in Noisy Radio Networks. In *Proceedings of the ACM Symposium on Principles of Distributed Computing*, pages 33–42. ACM, 2017.

8 Keren Censor-Hillel, Bernhard Haeupler, D Ellis Hershkowitz, and Goran Zuzic. Erasure Correction for Noisy Radio Networks, 2018. CoRR, Vol. abs/1805.04165. `arXiv:1805.04165`.

9 Imrich Chlamtac and Shay Kutten. On broadcasting in radio networks-problem analysis and protocol design. *IEEE Transactions on Communications*, 33(12):1240–1246, 1985.

10 Alejandro Cornejo, Seth Gilbert, and Calvin Newport. Aggregation in dynamic networks. In *Proceedings of the 2012 ACM symposium on Principles of distributed computing*, pages 195–204. ACM, 2012.

11 I. Dinur and D. Steurer. Analytical approach to parallel repetition. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing (STOC)*, pages 624–633, 2014.

12 Andrew Drucker, Fabian Kuhn, and Rotem Oshman. On the power of the congested clique model. In *Proceedings of the 2014 ACM symposium on Principles of distributed computing*, pages 367–376. ACM, 2014.

13 M. Elkin and G. Kortsarz. A combinatorial logarithmic approximation algorithm for the directed telephone broadcast problem. *SIAM journal on Computing*, 35(3):672–689, 2005.

14 M. Elkin and G. Kortsarz. Sublogarithmic approximation for telephone multicast. *Journal of Computer and System Sciences*, 72(4):648–659, 2006.

15 M. R. Garey and D. S. Johnson. *Computers and Intractability*. W. H. Freeman and Company, 1979.

16 A. Grama. *Introduction to parallel computing*. Pearson Education, 2003.

17 J. Iglesias, R. Rajaraman, R. Ravi, and R. Sundaram. Rumors across radio, wireless, telephone. In *Proceedings of the Leibniz International Proceedings in Informatics (LIPIcs)*, volume 45, 2015.

18 N. Jain, J. M. Lau, and L. Kale. Collectives on two-tier direct networks. In *Proceedings of the European MPI Users' Group Meeting*, pages 67–77, 2012.

19 B. Klenk, L. Oden, and H. Fröning. Analyzing communication models for distributed thread-collaborative processors in terms of energy and time. In *Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pages 318–327, 2015.

20 G. Kortsarz and D. Peleg. Approximation algorithms for minimum-time broadcast. *SIAM Journal on Discrete Mathematics*, 8(3):401–427, 1995.

21 Eyal Kushilevitz and Yishay Mansour. Computation in Noisy Radio Networks. In *SODA*, volume 98, pages 236–243, 1998.

22 F. T. Leighton, B. M. Maggs, and S. B. Rao. Packet routing and job-shop scheduling in O(congestion + dilation) steps. *Combinatorica*, 14(2):167–186, 1994.

23 Nathan Linial. Locality in distributed graph algorithms. *SIAM Journal on Computing*, 21(1):193–201, 1992.

24 L. Marchal, Y. Yang, H. Casanova, and Y. Robert. A realistic network/application model for scheduling divisible loads on large-scale platforms. In *Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 10–pp, 2005.

25 L. Oden, B. Klenk, and H. Fröning. Energy-efficient collective reduce and allreduce operations on distributed GPUs. In *Proceedings of the 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, pages 483–492, 2014.

26 P. Patarasuk and X. Yuan. Bandwidth efficient all-reduce operation on tree topologies. In *Proceedings of the 21st IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 1–8, 2007.

27 P. Patarasuk and X. Yuan. Bandwidth optimal all-reduce algorithms for clusters of workstations. *Journal of Parallel and Distributed Computing*, 69(2):117–124, 2009.

28 D. Peleg. Distributed computing. *SIAM Monographs on Discrete Mathematics and Applications*, 5, 2000.

29 R. Rabenseifner. Optimization of collective reduction operations. In *Proceedings of the International Conference on Computational Science*, pages 1–9, 2004.

30 R. Ravi. Rapid rumor ramification: Approximating the minimum broadcast time. In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 202–213, 1994.

31 T. Rothvoß. A simpler proof for O(congestion+dilation) packet routing. In *Proceedings of the International Conference on Integer Programming and Combinatorial Optimization*, pages 336–348, 2013.

32 A. Srinivasan and C. Teo. A constant-factor approximation algorithm for packet routing and balancing local vs. global criteria. *SIAM Journal on Computing*, 30(6):2051–2068, 2001.

33 S. Viswanathan, B. Veeravalli, and T. G. Robertazzi. Resource-aware distributed scheduling strategies for large-scale computational cluster/grid systems. *IEEE Transactions on Parallel and Distributed Systems*, 18(10), 2007.

A Formal Model, Problem, and Definitions

Let us formally define the Token Computation problem. The input to the problem is a Token Network specified by graph $G = (V, E)$ and parameters $t_c, t_m \in \mathbb{N}$. Each node starts with a single token.

An algorithm for this problem must provide a schedule, $S : V \times [l] \to V \cup \{\mathsf{idle}, \mathsf{busy}\}$ where we refer to $|S| \coloneqq l$ as the length of the schedule. Intuitively, a schedule S directs each node when to compute and when to communicate as follows:

- $S(v, r) = v' \neq v$ indicates that v begins passing a token to v' in round r of S;
- $S(v, r) = v$ indicates that v begins combining two token in round r of S;
- $S(v, r) = \mathsf{idle}$ indicates that v does nothing in round r;
- $S(v, r) = \mathsf{busy}$ indicates that v is currently communicating or computing.

Moreover, we define the number of computations that v has performed up to round r as $C_S(v, r) \coloneqq \sum_{r' \in [r-t_c]} \mathbb{1}(S(v, r') == v)$, the number of messages that v has received up to round r as $R_S(v, r) \coloneqq \sum_{r' \in [r-t_m]} \sum_{v' \neq v} \mathbb{1}(S(v', r') == v)$, and the number of messages that v has sent up to round r as $M_S(v, r) \coloneqq \sum_{r' \in [r-t_m]} \sum_{v' \neq v} \mathbb{1}(S(v, r') == v')$. Finally, define the number of tokens a node has in round r of S as follows.

$$\mathsf{tokens}_S(v, r) \coloneqq I(v) + R_S(v, r) - M_S(v, r) - C_S(v, r).$$

A schedule, S, is *valid* for Token Network (G, t_c, t_m) if:

1. Valid communication: If $S(v, r) = v' \neq v$ then $(v, v') \in E$, $S(v, r') = \mathsf{busy}$ for $r' \in [r + 1, r + t_m]$ and $\mathsf{tokens}_S(v, r) \geq 1$;
2. Valid computation: If $S(v, r) = v$ then $S(v, r') = \mathsf{busy}$ for $r' \in [r + 1, r + t_c]$ and $\mathsf{tokens}_S(v, r) \geq 2$;
3. Full aggregation: $\sum_{v \in V} \mathsf{tokens}_S(v, |S|) = 1$.

An algorithm solves Token Computation if it outputs a valid schedule.

B Deferred Related Work

There is a significant body of applied work in resource-aware scheduling, sensor networks, and high-performance computing that considers both the relative costs of communication and computation, often bundled together in an energy cost. However, these studies have been largely empirical rather than theoretical, and much of the work considers distributed algorithms (as opposed to our centralized setting).

AllReduce in HPC

There is much related applied work in the high-performance computing space on AllReduce [29, 16]. However, while there has been significant research on communication-efficient AllReduce algorithms, there has been relatively little work that explicitly considers the cost of computation, and even less work that considers the construction of optimal topologies for efficient distributed computation. Researchers have empirically evaluated the performance of different models of communication [25, 19] and have proven (trivial) lower bounds for communication without considering computation [26, 27]. Indeed, to the best of our knowledge, the extent to which they consider computation is through an additive penalty that consists of a multiplicative factor times the size of all inputs at all nodes, as in the work of Jain et al. [18]; crucially, this penalty is the same for any schedule and cannot be reduced via intelligent scheduling. Therefore, there do not seem to exist theoretical results for efficient algorithms that consider both the cost of communication and computation.

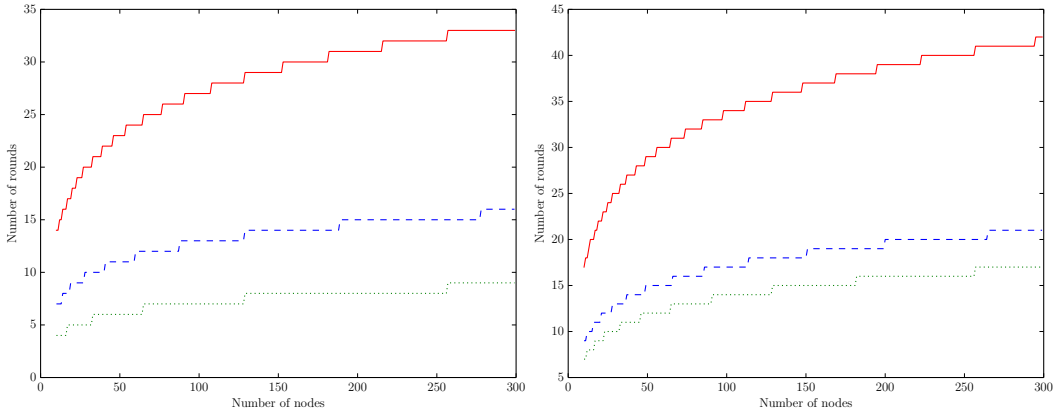
Resource-Aware Scheduling

In the distributed computation space, people have considered resource-aware scheduling on a completely connected topology with different nodes having different loads. Although this problem considers computation-aware communication, these studies are much more empirical than theoretical, and only consider distributed solutions as opposed to centralized algorithms [33, 24].

Sensor Networks

Members of the sensor networks community have studied the problem of minimizing an energy cost, which succinctly combines the costs of communication and computation. However, sensor networks involve rapidly-changing, non-static topologies [1, 2], which means that their objective is not to construct a fixed, optimal topology, but rather to develop adaptive algorithms for minimizing total energy cost with respect to an objective function [3].

C Deferred Figures



(a) $t_c = 1$, $t_m = 2$. **(b)** $t_c = 2$, $t_m = 1$.

Figure 6 An illustration of the optimal schedule length for different sized trees. The solid red line is the number of rounds taken by greedy aggregation with pipelining on a binary tree (i.e., $\lceil 2 \cdot t_c \cdot \log n + t_m \cdot \log n \rceil$); the dashed blue line is the number of rounds taken by greedy aggregation on T_n^*; and the dotted green line is the trivial lower bound of $\lceil t_c \cdot \log n \rceil$ rounds. Note that though we illustrate the trivial lower bound of $\lceil t_c \cdot \log n \rceil$ rounds, the true lower bound is given by the number of rounds taken by greedy aggregation on T_n^*.

D Proof of Lemma 7

Using our combining and shortcutting insights, we establish the following structure on a schedule which solves Token Computation on $K_{N^*(R)}$ in R rounds when $t_c < t_m$.

▶ **Lemma 15.** *When $t_c < t_m$, for all $R \in \mathbb{Z}_0^+$, there exists a schedule, $\tilde{S}^*$, of length R that solves* Token Computation *on $K_{N^*(R)}$ such that the terminus of $\tilde{S}^*$, v_t, never communicates and every computation performed by v_t involves a token that contains v_t's singleton token, a_{v_t}.*

Proof. Let S^* be some arbitrary schedule of length R which solves Token Computation on $K_{N^*(R)}$; we know that such a schedule exists by definition of $N^*(R)$. We first show how to modify S^* into another schedule, S^*_{1-4}, which not only also solves Token Computation on $K_{N^*(R)}$ in R rounds, but which also satisfies the following four properties.

(1) v only sends at time t if v at time t has exactly one token for $t \in [R]$;
(2) if v sends in round t then v does not receive any tokens in rounds $[t, t + t_m]$ for $t \in [R]$;
(3) if v sends in round t then v is idle during rounds $t' > t$ for $t \in [R]$;
(4) the terminus never communicates.

Achieving property (1)

Consider an optimal schedule S^*. We first show how to modify S^* to an R-round schedule S_1^* that solves Token Computation on $K_{N^*(R)}$ and satisfies property (1). We use our combining insight here. Suppose that (1) does not hold for S^*; i.e., a node v sends a token a_1 to node u at time t and v has at least one other token, say a_2, at time t. We modify S^* as follows. At time t, node v combines a_1 and a_2 into a token which it then performs operations on (i.e., computes and sends) as it does to a_2 in the original schedule. Moreover, node u

pretends that it receives token a_1 at time $t + t_m$: any round in which S^* has u compute on or communicate a_1, u now simply does nothing; nodes that were meant to receive a_1 do the same. It is easy to see that by repeatedly applying the above procedure to every node when it sends when it has more than one token, we can reduce the number of tokens every node has whenever it sends to at most one. The total runtime of this schedule is no greater than that of S^*, namely R, because $t_c < t_m$. Moreover, it clearly still solves Token Computation on $K_{N^*(R)}$. Call the schedule S_1^*.

Achieving properties (1) and (2)

Now, we show how to modify S_1^* into S_{1-2}^* such that properties (1) and (2) both hold. Again, S_{1-2}^* is of length R and solves Token Computation on $K_{N^*(R)}$. We use our shortcutting insight here. Suppose that (2) does not hold for S_1^*; i.e., there exists a v that receives a token a_1 from node u while sending another token a_2 to node u'. We say that node u is *bothering* node v in round t if node u communicates a token a_1 to v in round t, and node v communicates a token a_2 to node $u' \in V \setminus \{v, u\}$ in round $[t, t + t_m]$. Say any such pair is a *bothersome* pair. Furthermore, given a pair of nodes (u, v) and round t such that node u is bothering node v in round t, let the *resolution* of (u, v) in round t be the modification in which u sends its token directly to the node u' to which v sends its token. Note that each resolution does not increase the length of the optimal schedule because, by the definition of bothering, this will only serve as a shortcut; u' will receive a token from u at the latest in the same round it would have received a token from v in the original schedule, and nodes u' and v can pretend that they received tokens from v and u, respectively. However, it may now be the case that node u ends up bothering node u'. We now show how to repeatedly apply resolutions to modify S_1^* into a schedule S_{1-2}^* in which no node bothers another in any round t.

Consider the graph $B_t(S_1^*)$ where the vertices are the nodes in G and there exists a directed edge (u, v) if node u is bothering node v in round t in schedule S_1^*. First, consider cycles in $B_t(S_1^*)$. Note that, for any time t in which $B_t(S_1^*)$ has a cycle, we can create a schedule $\tilde{S}_1^*$ in which no nodes in any cycle in $B_t(S_1^*)$ send their tokens in round t; rather, they remain idle this round and pretend they received the token they would have received under S_1^*. Clearly, this does not increase the length of the optimal schedule and removes all cycles in round t. Furthermore, this does not violate property (1) because fewer nodes send tokens in round t, and no new nodes send tokens in round t.

Therefore, it suffices to consider an acyclic, directed graph $B_t(\tilde{S}_1)$. Now, for each round t, we repeatedly apply resolutions until no node bothers any other node during that round. Note that for every t, each node can only be bothering at most one other node because nodes can only send one message at a time. This fact, coupled with the fact that $B_t(\tilde{S}_1)$ is acyclic, means that $B_t(\tilde{S}_1)$ is a DAG where nodes have out-degree 1. It is not hard to see that repeatedly applying resolutions to a node v which bothers another node will decrease the number of edges in $B_t(\tilde{S}_1)$ by 1. Furthermore, because there are n total nodes in the network, the number of resolutions needed for any node v at time t is at most n.

Furthermore, repeatedly applying resolutions to $B_t(\tilde{S}_1)$ for times $t = 1, \ldots, R$ in order results in a schedule S_{1-2}^* with no bothersome pairs at any time t and that still satisfies property (1), and so schedule S_{1-2}^* satisfies properties (1) and (2). Since each resolution did not increase the length of the schedule we also have that S_{1-2}^* is of length R. Lastly, S_{1-2}^* clearly still solves Token Computation on $K_{N^*(R)}$.

Achieving properties (1) - (3)

Now, we show how to modify S^*_{1-2} into S^*_{1-3} which satisfies properties (1), (2), and (3). We use our shortcutting insight here as well as some new ideas. Given S^*_{1-2}, we show by induction over k from 0 to $R - t_m$, where R is the length of an optimal schedule, that we can modify S^*_{1-2} such that if a node finishes communicating in round $R-k$ (i.e., begins communicating in round $R-k-t_m$), it remains idle in rounds $t' \in (R-k, R]$ in the modified optimal schedule. The base case of $k = 0$ is trivial: If a node communicates in round $R - t_m$, it must remain idle in round R because the entire schedule is of length R.

Suppose there exists a node v that finishes communicating in round $t = R - k$ but is not idle in some round $t' > R - k$ in S^*_{1-2}; furthermore, let round t' be the first round after t in which node v is not idle. By property (1), node v must have sent its only token away in round t, and therefore node v must have received at least one other token after round t but before round t'. We now case on the type of action node v performs in round t'.

- If node v communicates in round t', it must send a token it received after time t but before round t'. Furthermore, as this is the first round after t in which v is not idle, v cannot have performed any computation on this token, and by the inductive hypothesis, v must remain idle from round $t' + t_m$ on. Therefore, v receives a token a_u from some node u and then forwards this token to node u' at time t'. One can modify this schedule such that u sends a_u directly to u' instead of sending to v.
- If node v computes in round t', consider the actions of node v after round $t' + t_c$. Either v eventually performs a communication after some number of computations, after which point it is idle by the inductive hypothesis, or v only ever performs computations from time t' on.

  In round t', v must combine two tokens it received after time $t + t_m$ by property (1). Note that two distinct nodes must have sent the two tokens to v because, by the inductive hypothesis, each node that sends after round t remains idle for the remainder of the schedule. Therefore, the nodes u_1 and u_2 that sent the two tokens to v must have been active at times $t'_1, t'_2 > t$, where $t_1 \leq t_2$, after which they remain idle for the rest of the schedule. Call the tuple (v, u_1, u_2) a *switchable* triple. We can modify the schedule to make v idle at round t' by picking the node that first sent to v and treating it as v while the original v stays idle for the remainder of the schedule. In particular, we can modify S^*_{1-2} such that, without loss of generality, u_2 sends its token to u_1 and u_1 performs the computation that v originally performed in S^*_{1-2}. Note that this now ensures that v will be idle in round t' and does not increase the length of the schedule, as u_1 takes on the role of v. Furthermore, node u_1's new actions do not violate the inductive hypothesis: Either u_1 only ever performs computations after time t', or it eventually communicates and thereafter remains idle.

  We can repeat this process for all nodes that are not idle after performing a communication in order to produce a schedule S^*_{1-3} in which property (3) is satisfied.

First, notice that these modifications do not change the length of S^*_{1-2}: in the first case u' can still pretend that it receives a_u at time $t' + t_m$ even though it now receives it in an earlier round and in the second case u_2 takes on the role of v at the expense of no additional round overhead. Also, it is easy to see that S^*_{1-3} still solves Token Computation on $K_{N^*(R)}$.

We now argue that the above modifications preserve (1) and (2). First, notice that the modifications we do for the first case do not change when any nodes send and so (1) is satisfied. In the second case, because we switch the roles of nodes, we may potentially add a send for a node. However, note that we only require a node u_1 to perform an additional send

when it is part of a switchable triple (v, u_1, u_2), and u_1 takes on the role of v in the original schedule from time t' on. However, because S^*_{1-2} satisfies (1), u was about to send its only token away and therefore only had one token upon receipt of the token from u_2. Therefore, because u_1 performs the actions that v performs in S^*_{1-2} from time t' on, and because at time t', both u_1 and v have exactly two tokens, (1) is still satisfied by S^*_{1-3}. Next, we argue that (3) is a strictly stronger condition than (2). In particular, we show that since S^*_{1-3} satisfies (3) it also satisfies (2). Suppose for the sake of contradiction that S^*_{1-3} satisfies (3) but not (2). Since (2) is not satisfied there must exist some node v that sends in some round t to, say node u, but receives a token in some round in $[t, t + t_m]$. By (3) it then follows that v is idle in all rounds after t. However, u also receives a token in round $t + t_m$. Therefore, in round $t + t_m$, two distinct nodes have tokens, one of which is idle in all rounds after $t + t_m$; this contradicts the fact that S^*_{1-3} solves Token Computation. Thus, S^*_{1-3} must also satisfy (2)

Achieving properties (1) - (4)

It is straightforward to see that S^*_{1-3} also satisfies property (4). Indeed, by property (3), if the terminus ever sends in round $t < R - t_c$, then the terminus must remain idle during rounds $t' > t$, meaning it must be idle in round $R - t_c$ which contradicts the fact that in this round the terminus performs a computation. Therefore, $S^*_{1-4} = S^*_{1-3}$ satisfies properties (1) - (4), and we know that there exists an optimal schedule in which v_t is always either computing or idle.

Achieving the final property

We now argue that we can modify S^*_{1-4} into another optimal schedule $\tilde{S}^*$ such that every computation done at the terminus v_t involves a token that contains the original singleton token that started at the terminus. Suppose that in S^*_{1-4}, v_t performs computation that does not involve a_{v_t}. Take the first instance in which v_t combines tokens a_1 and a_2, neither of which contains a_{v_t}, in round t. Because this is the first computation that does not involve a token containing a_{v_t}, both a_1 and a_2 must have been communicated to the terminus in round $t - t_m$ at the latest.

Consider the earliest time $t' > t$ in which v_t computes a token a_{comb} that contains all of a_1, a_2, and a_{v_t}. We now show how to modify S^*_{1-4} into $\tilde{S}'$ such that v_t computes a token a'_{comb} at time t' that contains all of a_1, a_2, and a_{v_t} and is at least the size of a_{comb} by having nodes swap roles in the schedule between times t and t'. Furthermore, because the rest of the schedule remains the same after time t', this implies that $\tilde{S}'$ uses at most as many rounds as S^*_{1-4}, and therefore that $\tilde{S}'$ uses at most R rounds.

The modification is as follows. At time t, instead of having v_t combine tokens a_1 and a_2, have v_t combine one of them (without loss of generality, a_1) with the token containing a_{v_t}. Now, continue executing S^*_{1-4} but substitute a_2 for the token containing a_{v_t} from round t on; this is a valid substitution because v_t possesses a_2 at time t. In round t', v_t computes a token $a'_{comb} = a_{comb}$; the difference from the previous schedule is that the new schedule has one fewer violation of property (4), i.e., one fewer round in which it computes on two tokens, neither of which contains a_{v_t}.

We repeat this process for every step in which the terminus does not compute on the token containing a_{v_t}, resulting in a schedule $\tilde{S}^*$ in which the terminus is always combining a communicated token with a token containing its own singleton token. Note that these modifications do not affect properties (1) - (4) because this does not affect the sending

actions of any node, and therefore $\tilde{S}^*$ still satisfies properties (1) - (4). It easily follows, then, that $\tilde{S}^*$ solves Token Computation on $K_{N^*(R)}$ in R rounds. Thus, $\tilde{S}^*$ is a schedule of length R that solves Token Computation on $K_{N^*(R)}$ in which every computation the terminus v_t does is on two tokens, one of which contains a_{v_t}, and, by (4), the terminus v_t never communicates. ◀

Having shown that the schedule corresponding to $N^*(R)$ can be modified to satisfy a nice structure when $t_c < t_m$, we can conclude our recursive bound on $N^*(R)$.

▶ **Lemma 7.** *When $t_c < t_m$, for $R \in \mathbb{Z}_0^+$ it holds that $N^*(R) = N^*(R-t_c)+N^*(R-t_c-t_m)$.*

Proof. Suppose $R \geq t_c + t_m$. We begin by applying Lemma 15 to show that $N^*(R) \leq N^*(R-t_c) + N^*(R-t_c-t_m)$. Let v_t be the terminus of the schedule $\tilde{S}^*$ using R rounds as given in Lemma 15. By Lemma 15, in all rounds after round t_m of $\tilde{S}^*$ it holds that v_t is either computing on a token that contains a_{v_t} or busy because it did such a computation. Notice that it follows that every token produced by a computation at v_t contains a_{v_t}.

Now consider the last token produced by our schedule. Call this token a. By definition of the terminus, a must be produced by a computation performed by v_t, combining two tokens, say a_1 and a_2, in round $R-t_c$ at the latest. Since every computation that v_t does combines two tokens, one of which contains a_{v_t}, without loss of generality let a_1 contain a_{v_t}.

We now bound the size of a_1 and a_2. Since a_1 exists in round $R-t_c$ we know that it is of size at most $N^*(R-t_c)$. Now consider a_2. Since every token produced by a computation at v_t contains a_{v_t} and a_2 does not contain a_{v_t} it follows that a_2 must either be a singleton token that originates at a node other than v, or a_2 was produced by a computation at another node. Either way, a_2 must have been sent to v, who then performed a computation on a_2 in round $R-t_c$ at the latest. It follows that a_2 exists in round $R-t_c-t_m$, and so a_2 is of size no more than $N^*(R-t_c-t_m)$.

Since the size of a just is the size of a_1 plus the size of a_2, we conclude that a is of size no more than $N^*(R-t_c) + N^*(R-t_c-t_m)$. Since, $\tilde{S}^*$ solves Token Computation on a complete graph of size $N^*(R)$, we have that a is of size $N^*(R)$ and so we conclude that $N^*(R) \leq N^*(R-t_c) + N^*(R-t_c-t_m)$ for $R \geq t_c + t_m$ when $t_c < t_m$.

Lastly, since $N^*(R) \geq N^*(R-t_c) + N^*(R-t_c-t_m)$ for $R \geq t_c + t_m$ by Lemma 5, we conclude that $N^*(R) = N^*(R-t_c) + N^*(R-t_c-t_m)$ for $R \geq t_c + t_m$ when $t_c < t_m$. ◀

E Proof of Theorem 8

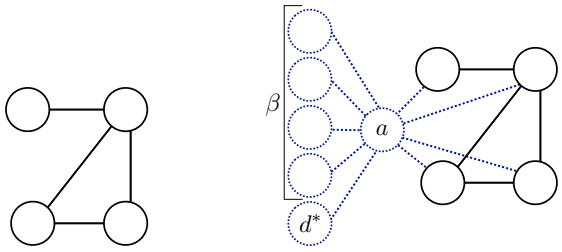
As a warmup for our hardness of approximation result, and to introduce some of the techniques, we begin with a proof that the decision version of Token Computation is NP-complete in Appendix E.1. We then prove the hardness of approximation result in Appendix E.2.

E.1 NP-Completeness (Warmup)

An instance of the decision version of Token Computation is given by an instance of Token Computation and a candidate ℓ. An algorithm must decide if there exists a schedule that solves Token Computation in at most ℓ rounds.

We reduce from k-dominating set.

▶ **Definition 16** (k-dominating set)**.** *An instance of k-dominating set consists of a graph $G = (V, E)$; the decision problem is to decide whether there exists $\kappa \subseteq V$ where $|\kappa| = k$ such that for all $v \in V \setminus \kappa$ there exists $\nu \in \kappa$ such that $(v, \nu) \in E$.*



(a) G. **(b)** $\Psi(G, 1)$.

Figure 7 An example of Ψ for a given graph G and $t_m = 1$. Nodes and edges added by Ψ are dashed and in blue. Notice that $|\beta| = \Delta + t_m = 3 + 1 = 4$.

Recall that k-dominating set is NP-complete.

▶ **Lemma 17** (Garey and Johnson [15]). *k-dominating set is NP-complete.*

Given an instance of k-dominating set, we would like to transform G into another graph G' in polynomial time such that G has a k-dominating set iff there exists a Token Computation schedule of some particular length for G' for some values of t_c and t_m.

We begin by describing the intuition behind the transformation we use, which we call Ψ. Any schedule on graph G in which every node only performs a single communication and which aggregates all tokens down to at most k tokens corresponds to a k-dominating set of G; in particular, those nodes that do computation form a k-dominating set of G. If we had a schedule of length $< 2t_m$ which aggregated all tokens down to k tokens, then we could recover a k-dominating set from our schedule. However, our problem aggregates down to only a single token, not k tokens. Our crucial insight, here, is that by structuring our graph such that a single node, a, must perform a great deal of computation, a must be the terminus of any short schedule. The fact that a must be the terminus and do a great deal of computation, in turn, forces any short schedule to aggregate all tokens in G down to at most k tokens at some point, giving us a k-dominating set.

Formally, Ψ is as follows. Ψ takes as input a graph G and a value for t_m and outputs $G' = (V', E')$. G' has G as a sub-graph and in addition has auxiliary node a where a is connected to all $v \in V$; a is also connected to dangling nodes $d \in \beta$, where $|\beta| = \Delta + t_m$, along with a special dangling node d^*.[9] Thus, $G' = (V \cup \{a, d^*\} \cup \beta, E \cup \{(a, v') : v' \in V' \setminus \{a\}\})$. See Figure 7.

We now prove that the optimal Token Computation schedule on $G' = \Psi(G, t_m)$ can be upper bounded as a function of the size of the minimum dominating set of G.

▶ **Lemma 18.** *The optimal* Token Computation *schedule on $G' = \Psi(G, t_m)$ is of length at most $2t_m + \Delta + k^*$ for $t_c = 1$, where k^* is the minimum dominating set of G.*

Proof. We know by definition of k^* that there is a dominating set of size k^* on G. Call this set κ and let $\sigma : V \to \kappa$ map any given $v \in V$ to a unique node in κ that dominates it. We argue that it must be the case that Token Computation requires at most $2t_m + t_c(\Delta + k^*)$ rounds on G' for $t_c = 1$. Roughly, we solve Token Computation by first aggregating at κ and then aggregating at a.

In more detail, in stage 1 of the schedule, every $d \in \beta$ sends to a, every node $v \in V$ sends to $\sigma(v)$ and a sends to d^*. This takes t_m rounds. In stage 2, each node does the following in parallel. Node d^* computes and sends its single token to a. Each $\nu \in \kappa$ computes until

[9] Δ is the max degree of G.

it has a single token and sends the result to a. Node a combines all tokens from $\beta \cup \{d^*\}$. Node d^* takes $1 + t_m$ rounds to do this. Each $\nu \in \kappa$ takes at most $\Delta + t_m$ rounds to do this. Node a takes $\Delta + t_m$ rounds to do this since a will receive d^*'s token after $t_m + 1$ rounds (and $\Delta \geq 1$ without loss of generality). Thus, stage 2, when done in parallel, takes $\Delta + t_m$ rounds. At this point a has $k^* + 1$ tokens and no other node in G' has a token. In stage 3, a computes until it has only a single token, which takes k^* rounds.

In total the number of rounds used by this schedule is $t_m + \Delta + t_m + k^* = 2t_m + \Delta + k^*$. Thus, the total number of rounds used by the optimal Token Computation schedule on G' is at most $2t_m + \Delta + k^*$. ◀

Next, we show that any valid Token Computation schedule on $G' = \Psi(G, t_m)$ that has at most two serialized sends corresponds to a dominating set of size bounded by the length of the schedule.

▶ **Lemma 19.** *Given $G' = \Psi(G, t_m)$ and a* Token Computation *schedule S for G' where $|S| < 3t_m$, $t_c = 1$, $\kappa = \{v : v \in G,\ v \text{ sends to } a \text{ in } S\}$ is a dominating set of G of size $|S| - 2t_m - \Delta$.*

Proof. Roughly, we argue that a must be the terminus of S and must perform at most $|S| - 2t_m - \Delta$ computations on tokens from G, each of which is the aggregation of a node's token and some of its neighbors' tokens. We begin by arguing that a must be the terminus.

First, we prove that no $d \in \beta \cup \{d^*\}$ is the terminus of S. Suppose for the sake of contradiction that some $\bar{d} \in \beta \cup \{d^*\}$ is the terminus. Since our schedule takes fewer than $3t_m$ rounds, we know that every node sends a token that is not just the singleton token with which it starts at most once. Thus, a sends tokens that are not just the singleton token that it starts with at most once. Since $|\beta \cup \{d^*\} \setminus \{\bar{d}\}| = \Delta + t_m$ and a is the only node connected to these nodes, we know that every singleton token that originates in $\beta \cup \{d^*\} \setminus \{\bar{d}\}$ must travel through a. Moreover, since a sends tokens that are not just the singleton token that it starts with at most once, a must send all such tokens as a single token. It follows that a must perform at least $\Delta + t_m$ computations, but then our entire schedule takes at least $t_m + \Delta + t_m + t_m = 3t_m + \Delta > 3t_m$ rounds – a contradiction to our assumption that our schedule takes less than $3t_m$ rounds.

We now argue that no $v \in G$ is the terminus. Suppose for the sake of contradiction that some $\bar{v} \in V$ is the terminus. Again, we know that a sends tokens that are not just the singleton token that it starts with at most once. Thus, every token in $\beta \cup \{d^*\}$ must travel through a, meaning that a must perform $\Delta + t_m + 1$ computations. It follows that the schedule takes $t_m + \Delta + t_m + t_m + 1 > 3t_m$ rounds, a contradiction to our assumption that the schedule takes $< 3t_m$ rounds.

Thus, since no $d \in \beta \cup \{d^*\}$ and no $v \in G$ is the terminus, we know that a must be the terminus.

We now argue that a sends a token in the first round and this is the only time that a sends (i.e., the only thing that a sends is the singleton token that it starts with, which it sends immediately). Assume for the sake of contradiction that a sends a token that it did not start with. It must have taken at least t_m rounds for this token to arrive at a and at least an additional t_m rounds for a to send a token containing it. Moreover, since a is the terminus, a token containing this token must eventually return to a and so an additional t_m rounds are required. Thus, at least $3t_m$ rounds are required if a sends a token other than that with which it starts, a contradiction to the fact that our schedule takes $< 3t_m$ rounds.

Thus, since a is the terminus, our schedule solves Token Computation in fewer than $3t_m$ rounds, and no computations occur in the first t_m rounds, a does at most $|S| - t_m$ computations. Since a never sends any token aside from its singleton token, and a is the only node to which $\beta \cup \{d^*\}$ are connected, we know that a must combine all tokens of nodes in $\beta \cup \{d^*\}$, where a must take $\Delta + t_m$ rounds to do so. Thus, since a takes $\Delta + t_m$ rounds to aggregate tokens from $\beta \cup \{d^*\}$ and it performs at most $|S| - t_m$ computations in total, a must receive at most $|S| - 2t_m - \Delta$ tokens from G. It follows that $|\kappa| \leq |S| - 2t_m - \Delta$.

Since each token sent by a node in κ to a must be sent at the latest in round $|S| - t_m$ and since $|S| < 3t_m$, we have that every token sent by a node in κ is formed in fewer than $2t_m$ rounds. It follows that each such token is formed by tokens that travel at most 1 hop in G. Since every node in G must eventually aggregate its tokens at a, it follows that every node in G is adjacent to a node in κ. Thus κ is a dominating set of G, and as shown before $|\kappa| \leq |S| - 2t_m - \Delta$. ◀

Having shown that the optimal Token Computation schedule of $G' = \Psi(G, t_m)$ is closely related to the size of the minimum dominating set, we prove that Token Computation is NP-complete.

▶ **Theorem 20.** *The decision version of* Token Computation *is NP-complete.*

Proof. The problem is clearly in NP. To show hardness, we reduce from k-dominating set. Specifically, we give a polynomial-time Karp reduction from k-dominating set to the decision version of Token Computation.

Our reduction is as follows. First, run $\Psi(G, t_m)$ for $t_m = \Delta + k + 1$ to get back G'. Next, return a decision version instance of Token Computation given by graph G' with $t_m = \Delta + k + 1$, $t_c = 1$ and $\ell = 2t_m + \Delta + k$. We now argue that G' has a schedule of length ℓ iff G has a k-dominating set.

- Suppose that G has a k-dominating set. We know that $k \geq k^*$, where k^* is the minimum dominating of G, and so by Lemma 18 we know that G' has a schedule of length at most $2t_m + \Delta + k^* \leq 2t_m + \Delta + k$.
- Suppose that G' has a Token Computation schedule S of length at most $2t_m + \Delta + k$. Notice that by our choice of t_m, we have that $|S| = 2t_m + \Delta + k < 3t_m$ and so by Lemma 19 we know that $\kappa = \{v : v \in G,\ v \text{ sends to } a \text{ in } S\}$ is a dominating set of G of size $|S| - 2t_m - \Delta$. Since $|S| \leq 2t_m + \Delta + k$ we conclude that $|\kappa| = |S| - 2t_m - \Delta \leq k$.

Lastly, notice that our reduction, Ψ, runs in polynomial time since it adds at most a polynomial number of vertices and edges to G. Thus, we conclude that k-dominating is polynomial-time reducible to the decision version of Token Computation, and therefore the decision version of Token Computation is NP-complete. ◀

E.2 Hardness of Approximation

We now show that unless P = NP there exists no polynomial-time algorithm that approximates Token Computation multiplicatively better than 1.5.

Recall that k-dominating set is $\Omega(\log n)$ hard to approximate.

▶ **Lemma 21** (Dinur and Steurer [11]). *Unless $P = NP$ every polynomial-time algorithm approximates minimum dominating set at best within a $(1 - o(1))(\log n)$ multiplicative factor.*

We prove hardness of approximation by using a $(1.5 - \epsilon)$ algorithm for Token Computation to approximate minimum dominating set with a polynomial-time algorithm better than $O(\log n)$. Similar to our proof of NP-completeness, given input graph G whose minimum

dominating set we would like to approximate, we would like to transform G into another graph G' such that a $(1.5-\epsilon)$-approximate Token Computation schedule for G' allows us to recover an approximately minimum dominating set.

One may hope to simply apply the transformation Ψ from the preceding section to do so. However, it is not hard to see that the approximation factor on the minimum dominating set recovered in this way has dependence on Δ, the maximum degree of G. If Δ is significantly larger than the minimum dominating set of G, we cannot hope that this will yield a good approximation to minimum dominating set. For this reason, before applying Ψ to G, we duplicate G a total of Δ/ϵ times to create graph G_α; this keeps Δ unchanged but increases the size of the minimum dominating set.[10] By applying Ψ to G_α instead of G to get back G'_α we are able to free our approximation factor from a dependence on Δ. Lastly, we show that we can efficiently recover an approximate minimum dominating set for G from an approximate Token Computation schedule for G'_α using our polynomial-time algorithm DSFromSchedule. Our full algorithm is given by MDSApx.

We first describe the algorithm – DSFromSchedule – we use to recover a minimum dominating set for G given a Token Computation schedule for $G'_\alpha = \Psi(G_\alpha, t_m)$. We denote copy i of G as G_i.

Algorithm 3 DSFromSchedule.

Input: $G'_\alpha = \Psi(G_\alpha, t_m)$; a valid Token Computation schedule for G'_α, S, of length $< 3t_m$; ϵ
Output: A dominating set for G of size $|S| - 2t_m - \Delta$
$\mathcal{K} \leftarrow \emptyset$
for $i \in \left[\frac{\Delta}{\epsilon}\right]$ **do**
  $\kappa_i \leftarrow \{v \in V_i : v \in G_\alpha \text{ sends to } a \text{ in } S\}$
  $\mathcal{K} \leftarrow \mathcal{K} \cup \{\kappa_i\}$
return $\arg\min_{\kappa_i \in \mathcal{K}} |\kappa_i|$

▶ **Lemma 22.** *Given $G'_\alpha = \Psi(G_\alpha, t_m)$ and a valid* Token Computation *schedule S for G'_α where $|S| < 3t_m$, $t_c = 1$ and $\epsilon \in (0, 1]$,* DSFromSchedule *outputs in polynomial time a dominating set of G of size $\frac{\epsilon}{\Delta}(|S| - 2t_m - \Delta)$.*[11]

Proof. Polynomial runtime is trivial, so we focus on the size guarantee. By Lemma 19 we know that $\kappa = \{v : v \in G_\alpha,\ v \text{ sends to } a \text{ in } S\}$ is a dominating set of G_α of size $|S| - 2t_m - \Delta$. Moreover, notice that $\kappa_i = \kappa \cap G_i$, and so it follows that κ_i is a dominating set of G_i, or equivalently G because G_i is just a copy of G. Thus we have that $\arg\min_{\kappa_i \in \mathcal{K}} |\kappa_i|$ will return a dominating set of G.

We now prove that $\arg\min_{\kappa_i \in \mathcal{K}} |\kappa_i|$ is small. Since each κ_i is disjoint we have $\sum_{i=1}^{\Delta/\epsilon} |\kappa_i| = |\kappa| \leq |S| - 2t_m - \Delta$. Thus, by an averaging argument we have that there must be some κ_i such that $|\kappa_i| \leq \frac{\epsilon}{\Delta}(|S| - 2t_m - \Delta)$. It follows that $\min_{\kappa_i \in \mathcal{K}} |\kappa_i| \leq \frac{\epsilon}{\Delta}(|S| - 2t_m - \Delta)$, meaning the κ_i that our algorithm returns is not only a dominating set of G but of size at most $\frac{\epsilon}{\Delta}(|S| - 2t_m - \Delta)$. ◀

[10] Since the max degree of G and G_α are the same, throughout this section Δ will be used to refer to both the max degree of G and the max degree of G_α.

[11] Since this lemma allows for $\epsilon \in (0, 1]$, it may appear that we will be able to achieve an arbitrarily good approximation for minimum dominating set. In fact, it might even seems as though we can produce a dominating set of size smaller than the minimum dominating set by simply letting ϵ be arbitrarily small. However, this is not the case. Intuitively, the smaller ϵ is, the larger G_α is and so the longer any feasible schedule S must be. Thus, decreases in ϵ are balanced out by increases in $|S|$ with respect to the size of our dominating set, $\frac{\epsilon}{\Delta}(|S| - 2t_m - \Delta)$.

Lastly, we combine Ψ with DSFromSchedule to get MDSApx, our algorithm for approximating minimum dominating set. Roughly, MDSApx constructs G'_α by applying Ψ to G_α, uses a $(1.5-\epsilon)$ approximation to Token Computation to get a schedule to G'_α and then uses DSFromSchedule to extract a minimum dominating set for G from this schedule. MDSApx will carefully choose a t_m that is large enough so that the schedule produced by the $(1.5-\epsilon)$ approximation for Token Computation is of length $< 3t_m$ but also small enough so that the produced schedule can be used to recover a small dominating set.

Algorithm 4 MDSApx.

Input: Graph G; $(1.5-\epsilon)$ Token Computation approximation algorithm $\mathcal{A}$
Output: An $O(1/\epsilon)$-approximation for the minimum dominating set of G
$\mathcal{D} \leftarrow \emptyset$
for $\hat{k} \in [n]$ **do**
  $G_\alpha \leftarrow \bigcup_{i=1}^{\Delta/\epsilon} G_i$
  $t_m \leftarrow \frac{1}{\epsilon}\left(\Delta + \frac{\hat{k}\Delta}{\epsilon}\right) + 1$; $t_c \leftarrow 1$
  $G'_\alpha \leftarrow \Psi(G_\alpha, t_m)$
  $S_{\hat{k}} \leftarrow \mathcal{A}\left(G'_\alpha, \frac{\hat{k}}{\epsilon}, t_m, t_c\right)$
  if $|S_{\hat{k}}| < 3t_m$ **then**
    $\kappa_{\hat{k}} \leftarrow$ DSFromSchedule(G_α, S, ϵ)
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{\kappa_{\hat{k}}\}$
return $\arg\min_{\kappa \in \mathcal{D}} |\kappa|$.

▶ **Lemma 23.** *Given graph G and a $(1.5-\epsilon)$-approximation algorithm for* Token Computation, *$\mathcal{A}$,* MDSApx *outputs in* poly$\left(n, \frac{1}{\epsilon}\right)$ *time a dominating set of G of size $O\left(\frac{k^*}{\epsilon}\right)$, where k^* is the size of the minimum dominating set of G.*

Proof. By Lemma 22 we know that any set $\kappa_{\hat{k}} \in \mathcal{D}$ is a dominating set of G of size at most $\frac{\Delta}{\epsilon}\left(|S_{\hat{k}}| - 2t_m - \Delta\right)$. Thus, it suffices to show that $\mathcal{D}$ contains at least one dominating set of G, $\kappa_{\hat{k}}$ such that $S_{\kappa_{\hat{k}}}$ is small. We do so now.

Let k^* be the size of the minimum dominating set of G. We know that $k^* \leq n$ and so in some iteration of MDSApx we will have $\hat{k} = k^*$. Moreover, the minimum dominating set of G_α in this iteration just is $\frac{\Delta k^*}{\epsilon}$ since G_α is just $\frac{\Delta}{\epsilon}$ copies of G. Consider this iteration. Let S^* be the optimal schedule for G'_α when $\hat{k} = k^*$. By Lemma 18 we know that $|S^*| \leq 2t_m + \Delta + \frac{k^*\Delta}{\epsilon}$. We now leverage the fact that that we chose t_m to be *large enough* so that $|S^*| < 3t_m$. In particular, combining the fact that $|S^*| \leq 2t_m + \Delta + \frac{k^*\Delta}{\epsilon}$ with the fact that $\mathcal{A}$ is a $(1.5-\epsilon)$ approximation we have that

$$
\begin{aligned}
|S_{k^*}| &\leq (1.5-\epsilon)|S^*| \\
&\leq (1.5-\epsilon)\left(2t_m + \Delta + \frac{k^*\Delta}{\epsilon}\right) \\
&= 3t_m - 2\,\epsilon\, t_m + (1.5-\epsilon)\left(\Delta + \frac{k^*\Delta}{\epsilon}\right) \\
&= 3t_m - 2\,\epsilon\, t_m + (1.5-\epsilon)\,\epsilon\,(t_m - 1) && \text{(By } t_m \text{ dfn.)} \\
&= 3t_m - \epsilon(0.5+\epsilon)t_m - \epsilon(1.5-\epsilon) \\
&< 3t_m. && (1)
\end{aligned}
$$

Thus, since $|S_{k^*}| < 3t_m$ we know that $\kappa_{k^*} \in \mathcal{D}$. Lastly, we argue that $|\kappa_{k^*}| = O\left(\frac{k^*}{\epsilon}\right)$, thereby showing that $\arg\min_{\kappa \in \mathcal{D}} |\kappa|$, the returned dominating set of our algorithm, is $O\left(\frac{k^*}{\epsilon}\right)$.

We now leverage the fact that we chose t_m to be *small enough* to give us a small dominating set. Applying Lemma 22 we have that

$$\begin{aligned}
|\kappa_{k^*}| &\leq \frac{\epsilon}{\Delta}\left(|S_{k^*}| - 2t_m - \Delta\right) && \text{(By Lemma 22)} \\
&< \frac{\epsilon}{\Delta}(t_m - \Delta) && \text{(By Equation (1))} \\
&= \frac{\epsilon}{\Delta}\left(\frac{1}{\epsilon}\left(\Delta + \frac{k^*\Delta}{\epsilon}\right) + 1 - \Delta\right) && \text{(By } t_m \text{ dfn.)} \\
&= \left(1 + \frac{k^*}{\epsilon}\right) + \frac{\epsilon}{\Delta} - \epsilon \\
&= O\left(\frac{k^*}{\epsilon}\right)
\end{aligned}$$

Thus, we conclude that MDSApx produces an $O\left(\frac{k^*}{\epsilon}\right)$ minimum dominating set of G.

Lastly, we argue a polynomial in n and $1/\epsilon$ runtime of MDSApx. First we argue that each iteration requires polynomial time. Constructing G_α takes polynomial time since the algorithm need only create $\frac{\Delta}{\epsilon} = \text{poly}\left(n, \frac{1}{\epsilon}\right)$ copies of G. Running Ψ also requires polynomial time since it simply adds polynomially many nodes to G_α. $\mathcal{A}$ is polynomial by assumption and DSFromSchedule is polynomial by Lemma 22. Thus, each iteration takes polynomial time and since MDSApx has n iterations, MDSApx takes polynomial time in n and $1/\epsilon$. ◀

Given that MDSApx demonstrates an efficient approximation for minimum dominating set given a polynomial-time $(1.5 - \epsilon)$ approximation for Token Computation, we conclude our hardness of approximation.

▶ **Theorem 8.** Token Computation *cannot be approximated by a polynomial-time algorithm within* $(1.5 - \epsilon)$ *for* $\epsilon \geq \frac{1}{o(\log n)}$ *unless* $P = NP$.

Proof. Assume for the sake of contradiction that P ≠ NP and there existed a polynomial-time algorithm $\mathcal{A}$ that approximated Token Computation within $(1.5 - \epsilon)$ for $\epsilon = \frac{1}{o(\log n)}$. It follows by Lemma 23 that MDSApx when run with $\mathcal{A}$ is a $o(\log n)$-approximation for minimum dominating set. However, this contradicts Lemma 21, and so we conclude that Token Computation cannot be approximated within $(1.5 - \epsilon)$ for $\epsilon \geq \frac{1}{o(\log n)}$. ◀

F Omitted Lemmas of the Proof of Theorem 9

F.1 Proof of Lemma 11

The goal of this section is to prove Lemma 11, which states the properties of GetDirectedPaths. To this end we will begin by rigorously defining the LP we use for GetDirectedPaths and establishing its relevant properties. We then formally define GetDirectedPaths, establish the properties of its subroutines and then prove Lemma 11.

F.1.1 Our Flow LP

The flow LP we use for GetDirectedPaths can be thought of as flow on a graph G "time-expanded" by the maximum length that a token in the optimal schedule travels. Given any schedule we define the *distance* that singleton token a travels as the number of times

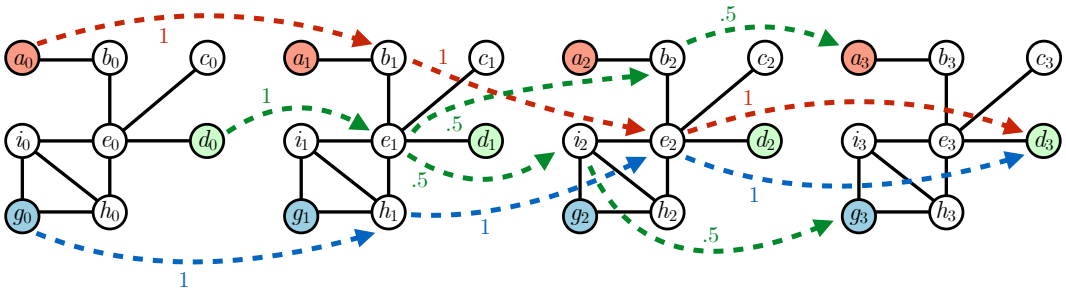


Figure 8 An illustration of non-zero flows for a feasible solution for PathsFlowLP(3) for graph G. Nodes a, d, and g are in W, and f_w is colored by w. For this feasible solution, $z = 2$.

any token containing a is sent in said schedule. Let L^* be the furthest distance a singleton token travels in the optimal schedule. Given a guess for L^*, namely $\hat{L}$, we define a graph $G_{\hat{L}}$ with vertices $\{v_r : v \in V, r \in [\hat{L}]\}$ and edges $\{e = (u_r, v_{r+1}) : (u, v) \in E, r \in [\hat{L} - 1]\}$. We have a flow type for each $w \in W$, where $W = \{v : v \text{ has at least 1 token}\}$, which uses $\{w' : w' \in W \wedge w' \neq w\}$ as sinks. Correspondingly, we have a flow variable, $f_w(x_r, y_{r+1})$ for every $r \in [\hat{L} - 1]$, $w \in W$ and $(x, y) \in E$. The objective function of the LP is to minimize the maximum vertex congestion, given by variable z. Let $z(\hat{L})$ be the objective value of our LP given our guess $\hat{L}$. Formally, our LP is given in PathsFlowLP($\hat{L}$), where $\Gamma(v)$ gives the neighbors of v in G. See Figure 8 for an illustration of a feasible solution to this LP.

$$\min z \text{ s.t.} \qquad \text{(PathsFlowLP}(\hat{L})\text{)}$$

"Conserve flow across rounds"

$$\sum_{x' \in \Gamma(x)} f_w(x'_{r-1}, x_r) = \sum_{x'' \in \Gamma(x)} f_w(x_r, x''_{r+1}) \qquad \forall w \in W, x \notin W, r \in [\hat{L} - 1] \quad (2)$$

"Every $w \in W$ is a source for f_w and not a sink for f_w"

$$\sum_{r \in [\hat{L}-1]} \left[\sum_{x' \in \Gamma(w)} f_w(w_r, x'_{r+1}) - \sum_{x' \in \Gamma(w)} f_w(x'_r, w_{r+1}) \right] \geq 1 \qquad \forall w \in W \quad (3)$$

"w-flow ends at $w' \in W$ s.t. $w' \neq w$"

$$\sum_{w' \in W : w' \neq w} \sum_{u \in \Gamma(w')} f_w(u_{\hat{L}}, w'_{\hat{L}}) = 1 \qquad \forall w \quad (4)$$

"z is the vertex congestion"

$$z \geq \sum_{w} \sum_{v \in \Gamma(v)} \sum_{r \in [D-1]} f_w(v'_r, v_{r+1}) \qquad \forall v \quad (5)$$

"Non-negative flow"

$$f_w(x_r, y_{r+1}) \geq 0 \qquad \forall, x, y, r, w \in W \quad (6)$$

F.1.2 Proof of the Key Property of our LP

The key property of our LP is that it has an optimal vertex congestion comparable to OPT. In particular, we can produce a feasible solution for our LP of cost 2OPT by routing tokens along the paths taken in the optimal schedule.

▶ **Lemma 24.** $\min(t_c, t_m) \cdot z(2L^*) \leq 2\text{OPT}$.

The remainder of this section is a proof of Lemma 24. Consider a W as in Section 4.2.1 where $W \leftarrow \{v : v \text{ has at least 1 token}\}$ and the optimal schedule that solves Token Computation in time OPT.

We will prove Lemma 24 by showing that, by sending flow along paths taken by certain tokens in the optimal schedule, we can provide a feasible solution to PathsFlowLP($\hat{L}$) with value commensurate with OPT. For this reason we now formally define these paths, OptPaths(W). Roughly, these are the paths taken by tokens containing singleton tokens that originate in W. Formally, these paths are as follows. Recall that a_w is the singleton token with which node w starts in the optimal schedule. Notice that in any given round of the optimal schedule exactly one token contains a_w. As such, order every round in which a token containing a_w is received by a node in ascending order as $r_0(w), r_1(w) \ldots$ where we think of w as receiving a_w in the first round. Correspondingly, let $v_i(w)$ be the vertex that receives a token containing a_w in round $r_i(w)$; that is $(v_1(w), v_2(w), \ldots)$ is the path "traced out" by a_w in the optimal schedule. For token a, let $C(a) \coloneqq \{a_{w'} : w' \in W \wedge a'_w \in a\}$ stand for all singleton tokens contained by token a that originated at a $w' \in W$. Say that token a is *active* if $|C(a)|$ is odd. Let $v_{L_w}(w)$ be the first vertex in $(v_1(w), v_2(w), \ldots)$ where an active token containing a_w is combined with another active token. Correspondingly, let $c(w)$ be the first round in which an active token containing a_w is combined with another active token. Say that a singleton token a_w is *pending* in round r if $r < c(w)$. We note the following behavior of pending singleton tokens.

▶ **Lemma 25.** *In every round of the optimal schedule, if a token is active then it contains exactly one pending singleton token and if a token is inactive then it contains no pending singleton tokens.*

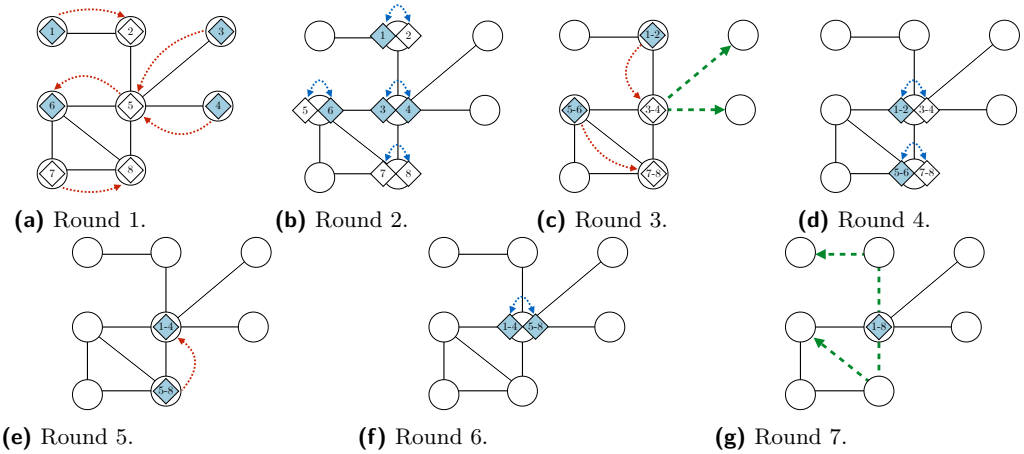
Proof. We prove this by induction over the rounds of the optimal schedule. As a base case, we note that in the first round of the optimal schedule a token is active iff it is a singleton node and every singleton node is pending. Now consider an arbitrary round i and assume that our claim holds in previous rounds. Consider an arbitrary token a. If a is not computed on by a node in this round then by our inductive hypothesis we have that it contains exactly one pending singleton token if it is active and no pending singleton tokens if it is not active. If a is active and combined with an inactive token, by our inductive hypothesis, the resulting token contains exactly one pending singleton token. Lastly, if a is active and combined with another active token by our inductive hypothesis these contain pending singletons a_w and a_u respectively such that $c(w) = c(u) = i$; it follows that the resulting token is inactive and contains no pending singleton tokens. This completes our induction. ◀

This behavior allows us to pair off vertices in W based on how their singleton tokens are combined.[12]

▶ **Lemma 26.** *For each $w \in W$ there exists a unique $u \in W$ such that $u \neq w$ and $v_{L_w}(w) = v_{L_u}(u)$ and $c(w) = c(u)$.*

Proof. Consider the round in which a token containing a_w, say a, is combined with an active token, say b, at vertex $v_{L_w}(w)$. Recall that this round is notated $c(w)$. By Lemma 25 we know that a and b contain exactly one pending singleton token, say a_w and a_u respectively.

[12] Without loss of generality we assume that $|W|$ is even here; if not, we can simply drop one element from W each time we construct OptPaths(W).



(a) Round 1. **(b)** Round 2. **(c)** Round 3. **(d)** Round 4.

(e) Round 5. **(f)** Round 6. **(g)** Round 7.

Figure 9 An illustration of the optimal schedule and how OptPaths(W) is constructed from it for a particular G. Active tokens are denoted by blue diamonds; inactive tokens are denoted by white diamonds; a dotted red arrow from node u to node v means that u sends to v; a double-ended blue arrow between two tokens a and b means that a and b are combined at the node; thick, dashed green lines give a path and its reversal in OptPaths(W) (for a total of 4 paths across all rounds) where $(v_1(w), v_2(w), \ldots v_{L_w}(w) = v_{L_u}(u), v_{L_u-1}(u), \ldots, v_1(u)) = P \in \text{OptPaths(W)}$ drawn only in round $c(w)$. Furthermore, token a labeled with $\{v : a \text{ contains } a_v\}$ and $W = \{1, 3, 4, 6\}$.

Since both a and b are active in this round and b contains a_u we have $c(u) = c(w)$. Moreover, since both a and b are combined at the same vertex we have $v_{L_u}(u) = v_{L_w}(w)$. Lastly, notice that this u is unique since by Lemma 25 there is exactly one singleton token, a_u, contained by b such that $c(u) \leq c(w)$. ◀

Having paired off vertices in W, we can now define OptPaths(W). Fix a w and let u be the vertex it is paired off with as in Lemma 26. We define $\text{OptPath}(w) := (v_1(w), v_2(w), \ldots v_{L_w}(w) = v_{L_u}(u), v_{L_u-1}(u), \ldots, v_1(u))$. Lastly, define $\text{OptPaths}(W) = \bigcup_{w \in W} \text{OptPath}(w)$. See Figure 9 for an illustration of how OptPaths(W) is constructed from the optimal schedule.

The critical property of OptPaths(W) is that it has vertex congestion commensurate with OPT as follows.

▶ **Lemma 27.** $con(\text{OptPaths}(W)) \leq \frac{2 \cdot \text{OPT}}{\min(t_c, t_m)}$.

Proof. Call a pair of directed paths in OptPaths(W) *complementary* if one path is OptPath(w) and the other OptPath(u) where u is to w as in Lemma 26. We argue that each pair of complementary paths passing through a given vertex v uniquely account for either t_c or t_m rounds of v's OPT rounds in the optimal schedule. Consider a pair of complementary paths, $P = (\text{OptPath}(w), \text{OptPath}(u))$, passing through a given vertex v. This pair of paths pass through v because in some round, say r_P, v sends a token containing a_u or a_w or v combines together tokens a and a' containing a_u and a_w respectively. Say that whichever of these operations accounts for P is *responsible* for P. Now suppose for the sake of contradiction that this operation of v in round r_P is responsible for another distinct pair P' of complementary paths, OptPath(w') and OptPath(u'). Notice that a_w, $a_{w'}$, a_u and $a_{u'}$ are all pending in round r_P. We case on whether v's action is a communication or a computation and show that v's operation cannot be responsible for P' in either case.

- Suppose that v is responsible for P and P' because it performs a computation in r_P. It follows that v combines an active token a and another active token a' where without loss of generality $a_w, a_{w'} \in a$ and $a_{w'}, a_u \in a'$. However, it then follows that a is active and contains two pending singleton tokens, which contradicts Lemma 25.
- Suppose that v is responsible for P and P' because it performs a communication in r_P by sending token a. It follows that without loss of generality $a_w, a_{w'} \in a$. However, either a is active or it is not. But by Lemma 25 if a is active it contains 1 pending singleton token and if a is not active then it contains 0 pending singleton tokens. Thus, the fact that v sends a token containing two pending singleton tokens contradicts Lemma 25.

Thus, it must be the case that v's action in r_P is uniquely responsible for P.

It follows that each computation and communication performed by v uniquely corresponds to a pair of complementary paths (consisting of a pair of paths in $\textsc{OptPaths}(W)$) that passes through v. Since v performs at most $\mathrm{OPT}/\min(t_c, t_m)$ operations in the optimal schedule, it follows that there are at most $\mathrm{OPT}/\min(t_c, t_m)$ pairs of complementary paths in $\textsc{OptPaths}(W)$ incident to v. Since each pair consists of two paths, there are at most $2 \cdot \mathrm{OPT}/\min(t_c, t_m)$ paths in $\textsc{OptPaths}(W)$ incident to v and so v has vertex congestion at most $2 \cdot \mathrm{OPT}/\min(t_c, t_m)$ in $\textsc{OptPaths}(W)$. Since v was arbitrary, this bound on congestion holds for every vertex. ◀

We now use $\textsc{OptPaths}(W)$ to construct a feasible solution for $\textsc{PathsFlowLP}(2L^*)$. We let $\tilde{f}$ be this feasible solution. Intuitively, $\tilde{f}$ simply sends flow along the paths of $\textsc{OptPaths}(W)$. More formally define $\tilde{f}$ as follows. For $w \in W$ and its corresponding path $\textsc{OptPath}(w) = (v_1(w), v_2(w), \ldots)$ we set $\tilde{f}_w(v_i, v_{i+1}) = 1$. We set all other variables of $\tilde{f}$ to 0 and let $\tilde{z}$ be the vertex congestion of $\textsc{OptPaths}(W)$.

▶ **Lemma 28.** *$(\tilde{f}, \tilde{z})$ is a feasible solution for $\textsc{PathsFlowLP}(2L^*)$ where $\tilde{z} \leq 2\mathrm{OPT}/\min(t_c, t_m)$.*

Proof. We begin by noting that every path in $\textsc{OptPaths}(W)$ is of length at most $2L^*$: for each $w \in W$, $\textsc{OptPath}(w)$ is the concatenation of two paths, each of which is of length no more than L^*. Moreover, notice that for each $w \in W$, the sink of $\textsc{OptPath}(w)$ is a $w' \in W$ such that $w' \neq w$.

We now argue that $(\tilde{f}, \tilde{z})$ is a feasible solution for $\textsc{PathsFlowLP}(2L^*)$: each vertex v with incoming w-flow that is not in $W \setminus w$ sends out this unit of flow and so Equation (2) is satisfied; since each $\textsc{OptPath}(w)$ is of length at most $2L^*$ and ends at a $w' \in W$ we have that every $w \in W$ is a source for f_w and not a sink for f_w, satisfying Equation (3); for the same reason, Equation (4) is satisfied; letting $\tilde{z}$ be the vertex congestion of $\textsc{OptPaths}(W)$ clearly satisfies Equation (5); and flow is trivially non-zero.

Lastly, since $\tilde{f}$ simply sends one unit of flow along each path in $\textsc{OptPaths}(W)$, our bound of $\tilde{z} \leq 2\mathrm{OPT}/\min(t_c, t_m)$ follows immediately from Lemma 27. ◀

We conclude that $\tilde{f}$ demonstrates that our LP has value commensurate with OPT.

▶ **Lemma 24.** $\min(t_c, t_m) \cdot z(2L^*) \leq 2\mathrm{OPT}$.

Proof. Since *Lemma* 28 shows that $(\tilde{f}, \tilde{z})$ is a feasible solution for $\textsc{PathsFlowLP}(2L^*)$ with cost at most $2\mathrm{OPT}/\min(t_c, t_m)$, our claim immediately follows. ◀

F.1.3 GetDirectedPaths Formally Defined

GetDirectedPaths solves our LP for different guesses of the longest path used by the optimal, samples paths based on the LP solution for our best guess, and then directs these paths. Formally, GetDirectedPaths is given in Algorithm 5, where $\xi := \lceil 2(n-1) \cdot (t_c + D \cdot t_m)/t_m \rceil$ is the range over which we search for L^*.

Algorithm 5 GetDirectedPaths(G, W).

Input: $W \subseteq V$ where $w \in W$ has a token
Output: Directed paths between nodes in W
$L \leftarrow \arg\min_{\hat{L} \in [\xi]} \left[t_m \cdot \hat{L} + \min(t_c, t_m) \cdot t(\hat{L}) \right]$
$f_w^* \leftarrow$ PathsFlowLP(L)
$\mathcal{P}_W \leftarrow$ SampleLPPaths(f_w^*, L, W)
$\vec{\mathcal{P}_U} \leftarrow$ AssignPaths($\mathcal{P}_W, W$)
return $\vec{\mathcal{P}_U}$

F.1.4 Sampling Paths from LP

Having shown that our LP has value commensurate with OPT and defined our algorithm based on this LP, we now provide the algorithm which we use to sample paths from our LP solution, SampleLPPaths. This algorithm produces a single sample by taking a random walk from each $w \in W$ where edges are taken with probability corresponding to their LP value. It repeats this $O(\log n)$ times to produce $O(\log n)$ samples. It then takes the sample with the most low congestion paths, discarding any high congestion paths in said sample. In particular, SampleLPPaths takes the sample $\mathcal{P}_W^i$ that maximizes $|Q(\mathcal{P}_W^i)|$ where $Q(\mathcal{P}_W^i) = \{P_w : P_w \in \mathcal{P}_W^i, \text{con}(P_w) \leq 10 \cdot z(\hat{L}) \log \hat{L}\}$ for an input $\hat{L}$.

Algorithm 6 SampleLPPaths(f_w^*).

Input: f_w^*, solution to PathsFlowLP($\hat{L}$); $\hat{L}$, guess of L^*; $W \subseteq V$
Output: Undirected paths between nodes in W
$\mathcal{C} \leftarrow \emptyset$
for sample $i \in O(\log n)$ **do**
  $\mathcal{P}_W^i \leftarrow \emptyset$
  for $w \in W$ **do**
    $v \sim f_w^*(w_1, v_2)$
    $P_w \leftarrow (w, v)$
    while $v \notin W$ **do**
      $v' \sim f_w^*(v_{|P_w|}, v'_{|P_w|+1})$
      $v \leftarrow v'$
      $P_w += v$
    $\mathcal{P}_W^i \leftarrow \mathcal{P}_W^i \cup \{P_w\}$
  $\mathcal{C} \leftarrow \mathcal{C} \cup \mathcal{P}_W^i$
$\mathcal{P}_S \leftarrow Q(\arg\max_{\mathcal{P}_W^i \in \mathcal{C}} |Q(\mathcal{P}_W^i)|)$
return $\mathcal{P}_W$

The properties of SampleLPPaths are as follows.

▶ **Lemma 29.** *For any fixed $W \subseteq V$, $\hat{L}$ and an optimal solution f^*_w to PathsFlowLP($\hat{L}$),* SampleLPPaths *is a polynomial-time randomized algorithm that outputs a set of undirected paths $\mathcal{P}_W$ such that $P_w \in \mathcal{P}_W$ is an undirected path with endpoints $w, w' \in S$ where $w \neq w'$. Also $|\mathcal{P}_W| \geq \frac{1}{3}|W|$ w.h.p., $con(\mathcal{P}_W) \leq z(\hat{L}) \cdot O(\log \hat{L})$, and $dil(\mathcal{P}_W) \leq \hat{L}$.*

Proof. Our proof consists of a series of union and Chernoff bounds over our samples. Consider an arbitrary $W \subseteq V$. Define T_w for $w \in W$ as the (directed) subgraph of $G_{\hat{L}}$ containing arc $(v, u) \in E_{\hat{L}}$ if for some r we have $f^*_w(x, y) > 0$. Notice that T_w is a weakly connected DAG where w has no edges into it: T_w does not contain any cycles since flow only moves from x_r to y_{r+1} for $x, y \in V$; by our flow constraints T_w must be weakly connected and w_r must have no edges into it for any r. Moreover, notice that P_w is generated by a random walk on T_w starting at w_1, where if the last vertex added to P_w was v, then we add u to P_w in step r of the random walk with probability $f^*_w(v_r, u_{r+1})$.

We first argue that every $P_w \in \mathcal{P}_W$ has endpoints $w, w' \in W$ for $w \neq w'$ and $\text{dil}(\mathcal{P}_W) \leq \hat{L}$. By construction, one endpoint of P_w is w. Moreover, the other endpoint of P_w will necessarily be a $w' \in W$ such that $w' \neq w$: by Equation (2) flow is conserved and by Equation (4) all flow from w must end at a point $w' \in W$ such that $w' \neq w$; thus our random walk will always eventually find such an w'. Moreover, notice that our random walk is of length at most $\hat{L}$ since T_w is of depth at most $\hat{L}$. Thus, every P_w is of length at most $\hat{L}$, meaning $\text{dil}(\mathcal{P}_W) \leq \hat{L}$.

Next, notice that, by the definition of Q, $\text{con}(\mathcal{P}_W) \leq z(\hat{L}) \cdot O(\log \hat{L})$ by construction since every element in $Q(\arg\max_{\mathcal{P}^i_W \in \mathcal{C}} |Q(\mathcal{P}^i_W)|)$ has $O(z(\hat{L}) \cdot O(\log \hat{L}))$ congestion.

Thus, it remains only to prove that $|\mathcal{P}_W| \geq \frac{1}{3}|W|$. We begin by arguing that for a fixed path P_w in a fixed set of sampled paths, $\mathcal{P}^i_W$ we have $\text{con}(P_w) \geq z(\hat{L}) \cdot O(\log \hat{L})$ with probability at most $\frac{1}{3}$. Consider a fixed path $P_w \in \mathcal{P}^i_W$ and fix an arbitrary $v \in P_w$. Now let X_{wv} stand for the random variable indicating the number of times that path P_w visits vertex w. without loss of generality we know that P_w contains no cycles (since if it did we could just remove said cycles) and so X_{sv} is either 1 or 0. By a union bound over rounds, then, we have $\mathbb{E}[X_{wv}] \leq \sum_r \sum_{u \in \Gamma(v)} f^*_W(u_r, v_{r+1}) \cdot \Pr(u \text{ taken in } (r-1)\text{th step}) \leq \sum_{u \in \Gamma(v)} \sum_r f^*_W(u_r, v_{r+1})$.

Now note that the congestion of a single vertex under our solution is just $\text{con}(v) = \sum_{w \in W} X_{wv}$. It follows that

$$\mathbb{E}[\text{con}(v)] = \sum_{w \in W} \mathbb{E}[X_{wv}] \leq \max_v \sum_w \sum_{u \in \Gamma(v)} \sum_r f^*_W(u_r, v_{r+1}) \leq z(\hat{L}).$$

Also notice that for a fixed v every X_{wv} is independent. Thus, we have by a Chernoff bound that that

$$\begin{aligned} \Pr(\text{con}(v) \geq z(\hat{L}) \cdot O(\log \hat{L})) &\leq \Pr\left(\sum_{w \in W} X_{wv} \geq \mathbb{E}\left[\sum_{w \in W} X_{wv}\right] \cdot O(\log \hat{L})\right) \\ &\leq \frac{1}{(\hat{L})^c} \end{aligned} \tag{7}$$

for c given by constants of our choosing. P_w is of length at most $\hat{L}$ by construction. Thus, by a union over $v \in P_w$ and Equation (7) we have that

$$\Pr\left(\text{con}(P_w) \geq z(\hat{L}) \cdot O(\log \hat{L})\right) \leq \frac{1}{\hat{L}^{c-1}} \leq \frac{1}{3}.$$

Thus, for a fixed path $P_w \in \mathcal{P}_W^i$ we know that this path has congestion at least $z(\hat{L}) \cdot O(\log \hat{L}))$ with probability at most $\frac{1}{3}$.

We now argue at least one of our $O(\log n)$ samples is such that at least $\frac{1}{3}$ of the paths in the sample have congestion at most $z(\hat{L}) \cdot O(\log \hat{L}))$. Let Y_{iw} be the random variable that is 1 if $P_w \in \mathcal{P}_W^i$ is such that $\text{con}(P_w) \geq z(\hat{L}) \cdot O(\log \hat{L}))$ and 0 otherwise. Notice that $\mathbb{E}[Y_{iw}] \leq \frac{1}{3}$ by the fact that a path has congestion at least $z(\hat{L}) \cdot O(\log \hat{L})$ with probability at most $\frac{1}{3}$. Now let $Z_i = \sum_{w \in W} Y_{iw}$ stand for the number of paths in sample i with high congestion. By linearity of expectation we have $\mathbb{E}[Z_i] \leq |W|\frac{1}{3}$. By Markov's inequality we have for a fixed i that $\Pr(Z_i \geq \frac{2}{3}|W|) \leq \Pr(Z_i \geq 2\,\mathbb{E}[Z_i]|W|) \leq \frac{1}{2}$. Now consider the probability that every sample i is such that more than $\frac{2}{3}$ of the paths have congestion more than $z(\hat{L}) \cdot O(\log \hat{L})$, i.e. consider the probability that for all i we have $Z_i \geq |W|\frac{2}{3}$. We have

$$\Pr\left(Z_i \geq |W|\frac{2}{3}, \forall i\right) \leq \left(\frac{1}{2}\right)^{O(\log n)} = \frac{1}{\text{poly}(n)}.$$

Thus, with high probability there will be some sample, i, such that $Z_i \leq |W|\frac{2}{3}$. It follows that with high probability $\max_{\mathcal{P}_W^i \in \mathcal{C}} |Q(\mathcal{P}_W^i)| \geq \frac{1}{3}|W|$ and since $\mathcal{P}_W = Q(\arg\max_{\mathcal{P}_W^i \in \mathcal{C}} |Q(\mathcal{P}_W^i)|)$, we conclude that with high probability $\mathcal{P}_W \geq \frac{1}{3}|W|$.

◀

F.1.5 Directing Paths

Given the undirected paths that we sample from our LP, $\mathcal{P}_W$, we produce a set of directed paths $\vec{\mathcal{P}}_U$ using AssignPaths, which works as follows. Define G' as the directed supergraph consisting of nodes W and directed edges $E' = \{(w, w') : w' \text{ is an endpoint of } P_w \in \mathcal{P}_W)\}$. Let $\Gamma_{G'}(v) = \{v' : (v', v) \in E' \vee (v, v') \in E'\}$ give the neighbors of v in G'. For each node $w \in G'$ with in-degree of at least two we do the following: if v has odd degree delete an arbitrary neighbor of w from G'; arbitrarily pair off the neighbors of w; for each such pair (w_1, w_2) add the directed path $P_{w_1} \circ rev(P_{w_2})$ to $\vec{\mathcal{P}}_U$ where $rev(P_{w_2})$ gives the result of removing the last element of P_{w_2} (namely, w) and reversing the direction of the path; remove $\{w, w_1, w_2\}$ from G'. Since we remove all vertices with in-degree of two or more and every vertex has out-degree 1, the remaining graph trivially consists only of nodes with in-degree at most 1 and out-degree at most 1. The remaining graph, therefore, is all cycles and paths. For each cycle or path $w_1, w_2, w_3, \ldots$ add the path corresponding to the edge from w_i to w_{i+1} for odd i to $\vec{\mathcal{P}}_U$. We let U be all sources of paths in $\vec{\mathcal{P}}_U$ and we let P_u be the path in $\vec{\mathcal{P}}_U$ with source u.

The properties of AssignPaths are as follows.

▶ **Lemma 30.** *Given $W \subseteq V$ and $\mathcal{P}_W = \{P_w : w \in W\}$ where the endpoints of P_w are $w, w' \in W$ for $w \neq w'$,* AssignPaths *in polynomial-time returns directed paths $\vec{\mathcal{P}}_U$ where at least $1/4$ of the nodes in W are the source of a directed path in $\vec{\mathcal{P}}_U$, each path in $\vec{\mathcal{P}}_U$ is of length at most $2 \cdot dil(\mathcal{P}_W)$ with congestion at most $con(\mathcal{P}_W)$ and each path in $\vec{\mathcal{P}}_U$ ends in a unique sink in W.*

Proof. When we add paths to $\vec{\mathcal{P}}_U$ that go through vertices of in-degree at least two, for every 4 vertices we remove we add at least one directed path to $\vec{\mathcal{P}}_U$ that is at most double the length of the longest a path in $\mathcal{P}_U$: in the worst case v has odd in-degree of 3 and we add only a single path. When we do the same for our cycles and paths for every 3 vertices we remove we add at least one directed path to $\vec{\mathcal{P}}_U$. Notice that by construction we clearly never reuse sinks in our directed paths. The bound on congestion and a polynomial runtime are trivial. ◀

F.1.6 Proof of Lemma 11

Finally, we conclude with the proof of Lemma 11.

▶ **Lemma 11.** *Given $W \subseteq V$,* GetDirectedPaths *is a randomized polynomial-time algorithm that returns a set of directed paths, $\vec{\mathcal{P}}_U = \{P_u : u \in U\}$ for $U \subseteq W$, such that with high probability at least $1/12$ of nodes in W are sources of paths in $\vec{\mathcal{P}}_U$ each with a unique sink in W. Moreover,*

$$con(\vec{\mathcal{P}}_U) \leq O\left(\frac{\mathrm{OPT}}{\min(t_c, t_m)} \log \frac{\mathrm{OPT}}{t_m}\right) \text{ and } dil(\vec{\mathcal{P}}_U) \leq \frac{8\mathrm{OPT}}{t_m}.$$

Proof. The fact that GetDirectedPaths returns a set of directed paths, $\vec{\mathcal{P}}_U$, such that at least $1/12$ of nodes in W are sources in a path with a sink in W follows directly from Lemma 29 and Lemma 30.

We now give the stated bounds on congestion and dilation. First notice that $2L^* \in [\xi]$. Moreover, $2\mathrm{OPT} \leq 2(n-1)(t_c + D \cdot t_m)$: the schedule that picks a pair of nodes, routes one to the other then aggregates and repeats $n-1$ times is always feasible and takes $(n-1)(t_c + D \cdot t_m)$ rounds. Thus, $2L^* \leq 2\frac{\mathrm{OPT}}{t_m} \leq \xi$.

Thus, by definition of L we know that

$$\begin{aligned} t_m \cdot L + \min(t_c, t_m) \cdot t(L) &\leq 2t_m \cdot L^* + \min(t_c, t_m) \cdot z(2L^*) \\ &\leq 2L^* + 2\mathrm{OPT} && \text{(By Lemma 24)} \\ &\leq 4\mathrm{OPT} && \text{(By dfn. of } L^*) \end{aligned}$$

It follows, then, that $t_m \cdot L \leq 4\mathrm{OPT}$ and so $L \leq \frac{4\mathrm{OPT}}{t_m}$. Similarly, we know that $\min(t_c, t_m) \cdot z(L) \leq 4\mathrm{OPT}$ and so $z(L) \leq \frac{4\mathrm{OPT}}{\min(t_c, t_m)}$.

Lastly, by Lemma 29 we know that $\mathrm{dil}(\mathcal{P}_W) \leq L \leq \frac{4\mathrm{OPT}}{t_m}$ and $\mathrm{con}(\mathcal{P}_W) \leq t(L) \cdot O(\log L) \leq O\left(\frac{\mathrm{OPT}}{\min(t_c, t_m)} \cdot \log \frac{\mathrm{OPT}}{t_m}\right)$. By Lemma 30 we get that the same congestion bound holds for $\vec{\mathcal{P}}_U$ and $\mathrm{dil}(\vec{\mathcal{P}}_U) \leq \frac{8\mathrm{OPT}}{\min(t_c, t_m)}$.

A polynomial runtime comes from the fact that we solve at most $(n-1)(t_c + D \cdot t_m) = \mathrm{poly}(n)$ LPs and then sample at most $(n-1)(t_c + D \cdot t_m)$ edges $O(\log n)$ times to round the chosen LP. ◀

F.2 Deferred Proofs of Section 4.2.2

▶ **Lemma 12.** *Given a set of directed paths $\vec{\mathcal{P}}_U$ with some subset of endpoints of paths in $\vec{\mathcal{P}}_U$ designated sources and the rest of the endpoints designated sinks,* OptRoute *is a randomized polynomial-time algorithm that w.h.p. produces a* Token Network *schedule that sends from all sources to sinks in $O(con(\vec{\mathcal{P}}_U) + dil(\vec{\mathcal{P}}_U))$.*

Proof. Given a set of paths $\vec{\mathcal{P}_U}$, Rothvoß [31] provides a polynomial-time algorithm that produces a schedule that routes along all paths in $O(\text{con}_E(\vec{\mathcal{P}_U}) + \text{dil}(\vec{\mathcal{P}_U})$ where $\text{con}_E(\mathcal{P}) = \max_e \sum_{P \in \mathcal{P}} \mathbb{1}(e \in P)$ is the edge congestion. However, the algorithm of Rothvoß [31] assumes that in each round a vertex can send a token along each of its incident edges whereas we assume that in each round a vertex can only forward a single token.

However, it is easy to use the algorithm of Rothvoß [31] to produce an algorithm that produces a Token Network routing schedule using $O(\text{con}(\vec{\mathcal{P}_U}) + \text{dil}(\vec{\mathcal{P}_U}))$ rounds which assumes that vertices only send one token per round as we assume in the Token Network model as follows. Let G be our input network with paths $\vec{\mathcal{P}_U}$ along which we would like to route where we assume that vertices can only send one token per round. We will produce another graph G' on which to run the algorithm of Rothvoß [31]. For each node $v \in G$ add nodes v_i and v_o to G'. Project each path $P \in \vec{\mathcal{P}_U}$ into G' to get $P' \in \vec{\mathcal{P}_U}'$ as follows: if edge (u, v) is in path $P \in \vec{\mathcal{P}_S}$ then add edge (u_o, v_i) and edge (v_i, v_o) to path P' in G'. Notice that $\text{con}(\vec{\mathcal{P}_U}) = \text{con}_E(\vec{\mathcal{P}_U}')$ and $\text{dil}(\vec{\mathcal{P}_U}) = 2\text{dil}(\vec{\mathcal{P}_U}')$. Now run the algorithm of Rothvoß [31] on G' with paths $\mathcal{P}'_U$ to get back some routing schedule S'.

Without loss of generality we can assume that S' only has nodes in G' send along a single edge in each round: every v_i is incident to a single outbound edge across all paths (namely (v_i, v_o)) and so cannot send more than one token per round; every v_o has a single incoming edge and so receives at most one token per round which, without loss of generality, we can assume v_o sends as soon as it receives (it might be the case that v_o collects some number of tokens over several rounds and then sends them all out at once but we can always just have v_o forward these tokens as soon as they are received and have the recipients "pretend" that they do not receive them until v_o would have sent out many tokens at once).

Now generate a routing schedule for G as follows: if v_o sends token a in round r of S' then v will send token a in round r of S. Since S only ever has vertices send one token per round, it is easy to see by induction over rounds that S will successfully route along all paths. Moreover, S takes as many rounds as S' which by [31] we know takes $O(\text{con}(\vec{\mathcal{P}'_U}) + \text{dil}(\vec{\mathcal{P}'_U})) = O(\text{con}(\vec{\mathcal{P}_U}) + 2\text{dil}(\vec{\mathcal{P}_U})) = O(\text{con}(\vec{\mathcal{P}_U}) + \text{dil}(\vec{\mathcal{P}_U}))$. Thus, we let OPTRoute be the algorithm that returns S. ◀

▶ **Lemma 13.** RoutePaths$_m$ *is a polynomial-time algorithm that, given* $\vec{\mathcal{P}_U}$*, solves the* Route and Compute *Problem w.h.p. using* $O(t_m(con(\vec{\mathcal{P}_U}) + dil(\vec{\mathcal{P}_U})) + t_c)$ *rounds.*

Proof. By Lemma 12, OPTRoute takes $t_m(\text{con}(\vec{\mathcal{P}_U}) + \text{dil}(\vec{\mathcal{P}_U}))$ rounds to route all sources to sinks. All sources are combined with sinks in the following computation and so RoutePaths$_m$ successfully solves the Route and Compute Problem since every source has its token combined with another token. The polynomial runtime of the algorithm is trivial. ◀

▶ **Lemma 14.** RoutePaths$_c$ *is a polynomial-time algorithm that, given* $\vec{\mathcal{P}_U}$*, solves the* Route and Compute *Problem w.h.p. using* $O(t_c \cdot con(\vec{\mathcal{P}_U}) + t_m \cdot dil(\vec{\mathcal{P}_U}))$ *rounds.*

Proof. We argue that every source's token ends at an asleep node with at least two tokens and no more than $\text{con}(\vec{\mathcal{P}_U})$ tokens. It follows that our computation at the end at least halves the number of tokens.

First notice that if a vertex falls asleep then it will receive at most $\text{con}(\vec{\mathcal{P}_S})$ tokens by the end of our algorithm since it is incident to at most this many paths. Moreover, notice that every token will either end at a sink or a sleeping vertex and every sleeping vertex is asleep because it has two or more tokens. It follows that every token is combined with at least one other token and so our schedule at least halves the total number of tokens.

The length of our schedule simply comes from noting that we have $O(\mathrm{dil}(\vec{\mathcal{P}_U}) \cdot t_m)$ forwarding rounds followed by $\mathrm{con}(\vec{\mathcal{P}_U}) \cdot t_c$ rounds of computation. Thus, we get a schedule of total length $O(t_c \cdot \mathrm{con}(\vec{\mathcal{P}_S}) + t_m \cdot \mathrm{dil}(\vec{\mathcal{P}_S}))$. A polynomial runtime is trivial. ◀

F.3 Proof of Theorem 9

▶ **Theorem 9.** SolveTC *is a polynomial-time algorithm that gives an* $O(\log n \cdot \log \frac{\mathrm{OPT}}{t_m})$*-approximation for* Token Computation *with high probability.*

Proof. By Lemma 11 we know that the paths returned by GetDirectedPaths, $\vec{\mathcal{P}_U}$ are such that $\mathrm{con}(\vec{\mathcal{P}_U}) \leq O\left(\frac{\mathrm{OPT}}{\min(t_c,t_m)} \log \frac{\mathrm{OPT}}{t_m}\right)$ and $\mathrm{dil}(\vec{\mathcal{P}_U}) \leq \frac{8\mathrm{OPT}}{t_m}$ and the paths returned have unique sinks and sources in W and there are at least $|W|/12$ paths w.h.p.

If $t_c > t_m$ then $\textsc{RoutePaths}_m$ is run which by Lemma 13 solves the Route and Compute Problem in $O(t_m \cdot \mathrm{con}(\vec{\mathcal{P}_U}) + t_m \cdot \mathrm{dil}(\vec{\mathcal{P}_U}) + t_c)$ rounds which is

$$\leq O\left(t_m \cdot \frac{\mathrm{OPT}}{\min(t_c,t_m)} \cdot \log \frac{\mathrm{OPT}}{t_m} + t_m \cdot \frac{8\mathrm{OPT}}{t_m} + t_c\right)$$
$$= O\left(\mathrm{OPT} \cdot \log \frac{\mathrm{OPT}}{t_m} + t_c\right)$$

If $t_c \leq t_m$ then $\textsc{RoutePaths}_c$ is run to solve the Route and Compute Problem which by Lemma 14 takes $O(t_c \cdot \mathrm{con}(\vec{\mathcal{P}_U}) + t_m \cdot \mathrm{dil}(\vec{\mathcal{P}_U}))$ rounds which is

$$\leq O\left(t_c \cdot \frac{4\mathrm{OPT}}{\min(t_c,t_m)} \cdot \log \frac{\mathrm{OPT}}{t_m} + t_m \cdot \frac{8\mathrm{OPT}}{t_m}\right)$$
$$= O\left(\mathrm{OPT} \cdot \log \frac{\mathrm{OPT}}{t_m}\right)$$

Thus, in either case, the produced schedule takes at most $O\left(\mathrm{OPT} \cdot \log \frac{\mathrm{OPT}}{t_m} + t_c\right)$ rounds to solve the Route and Compute Problem on at least $|W|/12$ paths in each iteration. Since solving the Route and Compute Problem reduces the total number of tokens by a constant fraction on the paths over which it is solved, and we have at least $|W|/12$ paths in each iteration w.h.p., by a union bound, every iteration reduces the total number of tokens by a constant fraction w.h.p. Thus, the concatenation of the $O(\log n)$ schedules produced, each of length $O(\mathrm{OPT} \cdot \log \frac{\mathrm{OPT}}{t_m} + t_c)$, is sufficient to reduce the total number of tokens to 1.

Thus, SolveTC produces a schedule that solves the problem of Token Computation in $O(\mathrm{OPT} \cdot \log n \log \frac{\mathrm{OPT}}{t_m} + t_c \cdot \log n)$ rounds. However, notice that $t_c \cdot \log n \leq \mathrm{OPT}$ (since the optimal schedule must perform at least $\log n$ serialized computations) and so the produced schedule is of length $O(\mathrm{OPT} \cdot \log n \log \frac{\mathrm{OPT}}{t_m} + t_c \log n) \leq O(\mathrm{OPT} \cdot \log n \log \frac{\mathrm{OPT}}{t_m})$. Lastly, a polynomial runtime is trivial given the polynomial runtime of our subroutines. ◀

Convertible Codes: New Class of Codes for Efficient Conversion of Coded Data in Distributed Storage

Francisco Maturana
Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA
http://www.cs.cmu.edu/~fmaturan/
fmaturan@cs.cmu.edu

K. V. Rashmi
Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA
http://www.cs.cmu.edu/~rvinayak/
rvinayak@cs.cmu.edu

Abstract

Erasure codes are typically used in large-scale distributed storage systems to provide durability of data in the face of failures. In this setting, a set of k blocks to be stored is encoded using an $[n, k]$ code to generate n blocks that are then stored on different storage nodes. A recent work by Kadekodi et al. [32] shows that the failure rate of storage devices vary significantly over time, and that changing the rate of the code (via a change in the parameters n and k) in response to such variations provides significant reduction in storage space requirement. However, the resource overhead of realizing such a change in the code rate on already encoded data in traditional codes is prohibitively high.

Motivated by this application, in this work we first present a new framework to formalize the notion of *code conversion* – the process of converting data encoded with an $[n^I, k^I]$ code into data encoded with an $[n^F, k^F]$ code while maintaining desired decodability properties, such as the maximum-distance-separable (MDS) property. We then introduce *convertible codes*, a new class of code pairs that allow for code conversions in a resource-efficient manner. For an important parameter regime (which we call the merge regime) along with the widely used linearity and MDS decodability constraint, we prove tight bounds on the number of nodes accessed during code conversion. In particular, our achievability result is an explicit construction of MDS convertible codes that are optimal for all parameter values in the merge regime albeit with a high field size. We then present explicit low-field-size constructions of optimal MDS convertible codes for a broad range of parameters in the merge regime. Our results thus show that it is indeed possible to achieve code conversions with significantly lesser resources as compared to the default approach of re-encoding.

2012 ACM Subject Classification Mathematics of computing → Coding theory; Theory of computation → Error-correcting codes

Keywords and phrases Coding theory, Reed-Solomon codes, Erasure codes, Code conversion, Distributed storage

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.66

Funding This work was funded in part by a Google Faculty Research Award.

Acknowledgements We thank Venkatesan Guruswami and Michael Rudow for their helpful suggestions on improving the writing for the paper.

1 Introduction

Erasure codes have become an essential tool for protecting against failures in distributed storage systems [18, 9, 30, 4]. Under erasure coding, a set of k data symbols to be stored is encoded using an $[n, k]$ code to generate n coded symbols, called a *codeword* (or *stripe*).

Each of the n symbols in a codeword is stored on a different storage node, and the system as a whole will contain several independent codewords distributed across different sets of storage nodes in the cluster.

A key factor that determines the choice of parameters n and k is the failure rate of the storage devices. It has been shown that failure rates of storage devices in large-scale storage systems can vary significantly over time and that changing the code rate, by changing n and k, in response to these variations yields substantial savings in storage space and hence the operating costs [32]. For example, in [32], the authors show that an 11% to 44% reduction in storage space can be achieved by tailoring n and k to changes in observed device failure rates. Such a reduction in storage space requirement translates to significant savings in the cost of resources and energy consumed in large-scale storage systems. It is natural to think of potentially achieving such a change in code rate by changing only n while keeping k fixed. However, due to several practical system constraints, changing code rate in storage systems often necessitates change in both the parameters n and k [32]. We refer the reader to [32] for a more detailed discussion on the practical benefits and constraints of adapting the erasure-code parameters with the variations in failure rates in a storage system.

Changing n and k for stripes in a storage system would involve converting already encoded data from one code to another. Such conversions, however, can generate a large amount of load (as explained below) which adversely affects the operation of the cluster. Furthermore, in some cases these conversions might need to be performed in an expedited manner, for example, to avoid the risk of data loss when facing an unexpected rise in failure rate. Hence it is critical to minimize the resource consumption of code conversion operations. Clearly, it is always possible to re-encode the data in a codeword (or a stripe) according to a new code by accessing (or decoding if needed) all the (original) message symbols. However, such an approach, which we call the *default approach*, requires accessing a large number of nodes (for example, for MDS codes, the initial value of k number of nodes need to be accessed) reading out all the data, transferring over the network, and re-encoding, which consumes large amounts of resources.

To the best of our knowledge, the existing literature [44, 59, 29] on formally studying the problem of changing parameters n and k for already encoded data model the code conversion problem within the framework of *repair-efficient* codes. Repair-efficient codes (e.g., [16, 47, 71, 49, 21]), which are a class of codes that can reconstruct a small subset of codeword symbols more efficiently than reconstructing the entire (original) message, has been an active area of research in the recent past. While the existing approach of exploiting repair efficiency for conversion efficiency provides reduction in the network bandwidth consumed as compared to the default approach for several parameter regimes, it has several drawbacks. For example, under this approach conversion requires accessing every symbol in the codeword and redistributing ("subsymbols") around. A more detailed discussion on repair-efficient codes and existing works which exploit repair efficiency for conversion efficiency is provided in Section 6.

In this paper, we propose a fundamentally new framework to model the code conversion problem, which is independent of repair efficiency. Our approach is based on the observation that the problem of changing code parameters in a storage system can be viewed as converting *multiple codewords* of an $[n^I, k^I]$ code (denoted by $\mathcal{C}^I$) into (potentially multiple) codewords of an $[n^F, k^F]$ code (denoted by $\mathcal{C}^F$)[1], with desired constraints on decodability such as both

[1] The superscripts I and F stand for initial and final respectively, representing the initial and final state of the conversion.

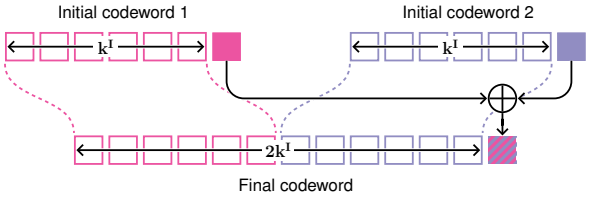


Figure 1 Example of code conversion: two codewords of a $[k^I + 1, k^I]$ single-parity-check code become one codeword of a $[2k^I + 1, 2k^I]$ single-parity-check code. The parity symbols are shown shaded. The data symbols from each codeword are preserved, and the parity symbol from the final codeword is the sum of the parities from each initial codeword.

codes satisfying the maximum-distance-separability (MDS) property. The code constructions that we present in this paper, under this new framework, require accessing significantly fewer symbols during conversion than existing works. Furthermore, even though the constructions presented in this paper optimize for reducing the number of symbols accessed for conversion, the bandwidth consumed for conversion is also lower in several parameter regimes of practical interest, compared to existing works (based on repair efficiency) that optimize for network bandwidth.

We now present a toy example to elucidate the concept of code conversion in our framework.

▶ **Example 1.** Consider conversion from an $[n^I = k^I + 1, k^I]$ code $\mathcal{C}^I$ to an $[n^F = k^F + 1, k^F]$ code $\mathcal{C}^F$. In our framework, this conversion is achieved by "merging" two codewords of the initial code for each codeword of the final code. Let us focus on the number of symbols accessed during conversion. The default approach requires accessing k^I symbols from each codeword belonging to $\mathcal{C}^I$ (initial codewords), and accessing at least one symbol for each codeword belonging to $\mathcal{C}^F$ (final codewords) to write out the result, totalling $2k^I + 1$ symbols accessed per final codeword. Alternatively, as depicted in Figure 1, one can choose $\mathcal{C}^I$ and $\mathcal{C}^F$ to be single-parity-check codes with the parity symbol holding the sum of the data symbols in each codeword (shown with a shaded box in the figure). To convert from $\mathcal{C}^I$ to $\mathcal{C}^F$, one sums the parity symbol from each initial codeword and stores the result as the parity symbol of the final codeword $\mathcal{C}^F$. This alternative approach requires accessing only three symbols for each final codeword, which is significantly more efficient.

Next, we give an overview of the main results and key ideas presented in this paper.

1.1 Overview of results

In this paper, we propose a *new framework to model the problem of code conversion*, that is, the process of converting data encoded with an $[n^I, k^I]$ code into data encoded with an $[n^F, k^F]$ code while maintaining desired decodability properties, such as maximum-distance-separable (MDS) property (Section 2). We then introduce a new class of code pairs, which we call *convertible codes*, that allow for resource-efficient conversions. We begin the study of this new class of code pairs, by focusing on an important regime where $k^F = \lambda k^I$ for any integer $\lambda \geq 2$ with arbitrary values of n^I and n^F, which we call the *merge regime*. Furthermore, we focus on the *access cost* of code conversion, which corresponds to *the total number of symbols accessed during conversion*. Keeping the number of symbols accessed small makes conversion less disruptive, allows the unaffected nodes to remain available for normal operations, and also reduces the amount of computation and communication needed.

Using our framework, we prove a tight lower bound on the access cost of conversions for linear MDS codes in the merge regime (Section 3). Let $r^I = n^I - k^I$ and $r^F = n^F - k^F$. This lower bound identifies two regions: (1) $r^F \leq r^I$, where significant savings in access cost can be achieved when $r^F < k^I$, and (2) the complement $r^F > r^I$, where linear MDS codes cannot achieve lower access cost than the default approach. Specifically, we prove:

▶ **Theorem 15.** *For all linear MDS $(n^I, k^I; n^F, k^F = \lambda k^I)$ convertible codes, the access cost of conversion is at least $r^F + \lambda \min\{k^I, r^F\}$. Furthermore, if $r^I < r^F$, the access cost of conversion is at least $r^F + \lambda k^I$.*

We prove this lower bound first on a class of linear MDS convertible codes that we call *stable* convertible codes. We then extend this lower bound to all linear MDS convertible codes by showing that non-stable convertible codes cannot achieve this lower bound, and thus all optimal linear MDS convertible codes in the merge regime are stable.

Then, we describe a construction of linear MDS Convertible Codes in the merge regime that meet the bound of Theorem 15, thereby providing convertible codes that are access-optimal (Section 4). Specifically:

▶ **Theorem 22.** *The explicit construction provided in Section 4.1 yields access-optimal linear MDS convertible codes for all parameter values in the merge regime.*

The construction is deterministic and yields codes over a finite extension field. To prove that those codes are access-optimal, we show that the resulting generator matrices have a particular structure, and that none of their minors are zero when the degree of the field extension is large enough. This implies that the field size required is very large (exponential in n^F for fixed constant final code rate and typical parameters). Field size is an important aspect of code design that has been widely studied in the context of other code families [12, 62, 2, 24]. To address this issue, we introduce a sequence of constructions of convertible codes that have significantly lower field size requirements (Section 5). This sequence of constructions, which we denote Hankel$_s$ ($s \in \{\lambda, \ldots, r^I\}$), presents a tradeoff between field size and the parameter range that they support. Specifically, we show:

▶ **Theorem 25.** *Given parameters k^I, r^I, λ, and a field $\mathbb{F}_q$, Hankel$_s$ ($s \in \{\lambda, \ldots, r^I\}$) constructs an access-optimal $(n^I, k^I; n^F, k^F = \lambda k^I)$ convertible code if:*

$$r^F \leq (s - \lambda + 1)\left\lfloor \frac{r^I}{s} \right\rfloor + \max\{(r^I \bmod s) - \lambda + 1, 0\} \quad \text{and} \quad q \geq sk^I + \left\lfloor \frac{r^I}{s} \right\rfloor - 1.$$

Of particular interest are the two extreme points of this tradeoff which we call Hankel-I and Hankel-II respectively: Hankel-I supports $r^F \leq \lfloor r^I/\lambda \rfloor$ requiring field size $q \geq \max\{n^I - 1, n^F - 1\}$; and Hankel-II supports a wider range $r^F \leq r^I - \lambda + 1$ requiring field size $q \geq k^I r^I$.

The key idea behind our construction is to construct the parity generator matrices of the initial and final codes as cleverly-chosen submatrices of a specially constructed upper-left triangular array. This array has two important properties: (1) every square submatrix of this array is non-singular, and (2) it has *Hankel form*, that is, each ascending diagonal from left to right is constant. By exploiting the repetitive structure of the array, our constructions yield convertible codes that achieve optimal access cost during conversion. Given the way in which these codes are constructed, they also have the additional property of being (punctured) generalized doubly-extended Reed-Solomon codes.

Our results thus show that there is a broad regime where code conversions can be achieved with significantly less access cost than the default approach, and another regime where achieving less access cost than the default approach is not possible. We provide a

general explicit construction of access-optimal convertible codes in the former regime, and low-field-size constructions for a wide parameter range within this regime. The framework of convertible codes presented in this work opens several new opportunities which pose interesting theoretical questions that have a potential for impact on real-world systems.

1.2 Background

In this subsection we introduce some notation and basic definitions related to linear codes. Let $\mathbb{F}_q$ be a finite field of size q. An $[n, k]$ linear code $\mathcal{C}$ over $\mathbb{F}_q$ is a k-dimensional subspace $\mathcal{C} \subseteq \mathbb{F}_q^n$. Here, n is called the length of the code, and k is called the dimension of the code. A *generator matrix* of an $[n, k]$ linear code $\mathcal{C}$ over $\mathbb{F}_q$ is a $k \times n$ matrix $\mathbf{G}$ over $\mathbb{F}_q$ such that the rows of $\mathbf{G}$ form a basis of the subspace $\mathcal{C}$. A $k \times n$ generator matrix $\mathbf{G}$ is said to be *systematic* if it has the form $\mathbf{G} = [\mathbf{I} \mid \mathbf{P}]$, where $\mathbf{I}$ is the $k \times k$ identity matrix and $\mathbf{P}$ is a $k \times (n - k)$ matrix. Even though the generator matrix of a code $\mathcal{C}$ is not unique, we will sometimes associate code $\mathcal{C}$ to a specific generator matrix $\mathbf{G}$, which will be clear from the context. The encoding of a message $\mathbf{m} \in \mathbb{F}_q^k$ under an $[n, k]$ code $\mathcal{C}$ with generator matrix $\mathbf{G}$ is denoted $\mathcal{C}(\mathbf{m}) = \mathbf{m}^T\mathbf{G}$.

Let $[i]$ denote the set $\{1, 2, \ldots, i\}$. A linear code $\mathcal{C}$ is *maximum distance separable* (MDS) if the minimum distance of the code is the maximum possible:

$$\text{min-dist}(\mathcal{C}) = \min_{c \neq c' \in \mathcal{C}} |\{i \in [n] : c_i \neq c'_i\}| = n - k + 1$$

where $c_i \in \mathbb{F}_q$ denotes the i-th coordinate of c. Equivalently, a linear code $\mathcal{C}$ is MDS if and only if every $k \times k$ submatrix of its generator matrix $\mathbf{G}$ is non-singular [34].

2 A framework for studying code conversions

In this section, we formally define our new framework for studying *code conversions* and introduce *convertible codes*. While we use the notation of linear codes introduced in Section 1.2, the framework introduced in this section can be applied to arbitrary (not necessarily linear) codes. Suppose one wants to convert data that is already encoded using an $[n^I, k^I]$ initial code $\mathcal{C}^I$ into data encoded using an $[n^F, k^F]$ final code $\mathcal{C}^F$ where both codes are over the same field $\mathbb{F}_q$. In the initial and final configurations, the system must store the same information, but encoded differently. In order to capture the changes in the dimension of the code during conversion, we consider $M = \text{lcm}(k^I, k^F)$ number of "message" symbols (i.e., the data to be stored) over a finite field $\mathbb{F}_q$, denoted by $\mathbf{m} \in \mathbb{F}_q^M$. This corresponds to multiple codewords in the initial and final configurations. We note that this *need for considering multiple codewords in order to capture the smallest instance of the problem* deviates from existing literature on the code repair problem (e.g., [16, 26, 47, 49]) and code locality (e.g., [21, 41, 27]), where a single codeword is sufficient to capture the problem.

Since there are multiple codewords, we first specify an *initial partition* $\mathcal{P}_I$ and a *final partition* $\mathcal{P}_F$ of the set $[M]$, which map the message symbols of $\mathbf{m}$ to their corresponding initial and final codewords. The initial partition $\mathcal{P}_I \subseteq 2^{[M]}$ is composed of M/k^I disjoint subsets of size k^I, and the final partition $\mathcal{P}_F \subseteq 2^{[M]}$ is composed of M/k^F disjoint subsets of size k^F. In the initial (respectively, final) configuration, the data indexed by each subset $S \in \mathcal{P}_I$ (respectively, $\mathcal{P}_F$) is encoded using the code $\mathcal{C}^I$ (respectively, $\mathcal{C}^F$). The codewords $\{\mathcal{C}^I(\mathbf{m}_S),\ S \in \mathcal{P}_I\}$ are referred to as *initial codewords*, and the codewords $\{\mathcal{C}^F(\mathbf{m}_S),\ S \in \mathcal{P}_F\}$ are referred to as *final codewords*, where $\mathbf{m}_S$ corresponds to the projection of $\mathbf{m}$ onto the coordinates in S. The descriptions of the initial and final partitions and codes, along with the conversion procedure, define a convertible code. We now proceed to define conversions and convertible codes formally.

▶ **Definition 2** (Code conversion). *A* conversion *from an initial code $\mathcal{C}^I$ to a final code $\mathcal{C}^F$ with initial partition $\mathcal{P}_I$ and final partition $\mathcal{P}_F$ is a procedure, denoted by $T_{\mathcal{C}^I \to \mathcal{C}^F}$, that for any $\mathbf{m}$, takes the set of initial codewords $\{\mathcal{C}^I(\mathbf{m}_S) \mid S \in \mathcal{P}_I\}$ as input, and outputs the corresponding set of final codewords $\{\mathcal{C}^F(\mathbf{m}_S) \mid S \in \mathcal{P}_F\}$.*

▶ **Definition 3** (Convertible code). *A $(n^I, k^I; n^F, k^F)$ convertible code over $\mathbb{F}_q$ is defined by: (1) a pair of codes $(\mathcal{C}^I, \mathcal{C}^F)$ where $\mathcal{C}^I$ is an $[n^I, k^I]$ code over $\mathbb{F}_q$ and $\mathcal{C}^F$ is an $[n^F, k^F]$ code over $\mathbb{F}_q$; (2) a pair of partitions $\mathcal{P}_I, \mathcal{P}_F$ of $[M = \text{lcm}(k^I, k^F)]$ such that each subset in $\mathcal{P}_I$ is of size k^I and each subset in $\mathcal{P}_F$ is of size k^F; and (3) a conversion procedure $T_{\mathcal{C}^I \to \mathcal{C}^F}$ that on input $\{\mathcal{C}^I(\mathbf{m}_S) \mid S \in \mathcal{P}_I\}$ outputs $\{\mathcal{C}^F(\mathbf{m}_S) \mid S \in \mathcal{P}_F\}$ for all $\mathbf{m} \in \mathbb{F}_q^M$.*

Typically, additional constraints would be imposed on $\mathcal{C}^I$ and $\mathcal{C}^F$, such as requiring both codes to be MDS.

▶ Remark 4. Note that the definition of convertible codes (Definition 3) assumes that $(n^I, k^I; n^F, k^F)$ are known at the time of code construction. This will be helpful in understanding the fundamental limits of the conversion process. In practice, this assumption might not hold. For example, n^F, k^F might depend on the node failure rates that are yet to be observed. *Interestingly, it is possible for a $(n^I, k^I; n^F, k^F)$ convertible code to facilitate conversion to multiple values of n^F, k^F simultaneously, as is the case for the code constructions presented in this paper.*

The cost of conversion is determined by the cost of the conversion procedure $T_{\mathcal{C}^I \to \mathcal{C}^F}$, as a function of the parameters $(n^I, k^I; n^F, k^F)$. Towards minimizing the overhead of the conversion, our general objective is to design codes $(\mathcal{C}^I, \mathcal{C}^F)$, partitions $(\mathcal{P}_I, \mathcal{P}_F)$ and conversion procedure $T_{\mathcal{C}^I \to \mathcal{C}^F}$ that satisfy Definition 3 and minimize the conversion cost for given parameters $(n^I, k^I; n^F, k^F)$, subject to desired decodability constraints on $\mathcal{C}^I$ and $\mathcal{C}^F$.

Depending on the relative importance of various resources in the cluster, one might be interested in optimizing the conversion with respect to various types of costs such as symbol access, computation (CPU), communication (network bandwidth), read/writes (disk IO), etc., or a combination of these costs. The general formulation of code conversions above provides a powerful framework to theoretically reason about convertible codes. In what follows, we will focus on a specific regime and a specific cost model.

3 Lower bounds on access cost of code conversion

The focus of this section is on deriving lower bounds on the access cost of code conversion in the merge regime (as defined below). We focus on a fundamental regime given by $k^F = \lambda k^I$, where integer $\lambda \geq 2$ is the number of initial codewords merged, with arbitrary values of n^I and n^F. We call this regime as *merge regime*. We additionally require that both the initial and final code are linear and MDS. Since linear MDS codes are widely used in storage systems and are well understood in the Coding Theory literature, they constitute a good starting point. In terms of cost of conversion, we focus on the *access cost* of code conversion, that is, the number of symbols that are affected by the conversion. Each symbol read from the initial codewords requires one symbol access and each symbol written to the final codeword requires one symbol access. Therefore, minimizing access cost amounts to *minimizing the sum of the number of symbols written to the final codeword and the number of symbols read from the initial codewords.*[2] Keeping this number small makes code conversion less disruptive

[2] Readers who are familiar with the literature on regenerating codes might observe that convertible codes optimizing for the access cost are "scalar" codes as opposed to being "vector" codes.

and allows the unaffected symbols to remain available for normal operation. Furthermore, reducing the number of accesses also reduces the amount of computation and communication required in contrast to the default approach.

▶ Remark 5. Note that, when defining the access cost above, we implicitly assume that conversion is performed by downloading all the required data to a central location, where the new symbols are computed and then distributed to their final locations. This assumption does not affect the access cost, but it could affect other forms of cost, such as network bandwidth, which could be reduced by transferring data in a different way.

We now introduce some notation. We use the term *unchanged symbols* to refer to symbols from the initial codewords that remain *as is* in the final codeword, and we use the term *new symbols* to refer to symbols from the final codeword that are not present in the initial codewords (i.e. they are not unchanged). For example, in Figure 1, all the data symbols are unchanged symbols (unshaded boxes), and the parity symbol of the final codeword is a new symbol (striped box). We define the *read access set* of an $(n^I, k^I; n^F, k^F = \lambda k^I)$ convertible code as a set of tuples $\mathcal{D} \in [\lambda] \times [n^I]$, where $(i, j) \in \mathcal{D}$ corresponds to the j-th symbol of initial codeword i. The set $\mathcal{D}$ must be such that all new symbols are linear combinations of symbols indexed by the tuples in $\mathcal{D}$. Furthermore, we use $\mathcal{D}_i = \{j \mid (i, j) \in \mathcal{D}\}$, $\forall i \in [\lambda]$ to denote the symbols read from a particular initial codeword.

In Section 4, we show that the lower bounds on the access cost derived in this section are in fact achievable. Therefore, we refer to MDS convertible codes in the merge regime achieving these lower bounds as *access-optimal*.

▶ **Definition 6** (Access-optimal). *A linear MDS $(n^I, k^I; n^F, k^F = \lambda k^I)$ convertible code is said to be* access-optimal *if and only if it attains the minimum access cost over all linear MDS $(n^I, k^I; n^F, k^F = \lambda k^I)$ convertible codes.*

Now we present the access cost lower bounds of convertible codes in the merge regime.

3.1 Lower bounds on the access cost of code conversion

In this subsection, we present lower bounds on the access cost of linear MDS convertible codes in the merge regime. This is done in four steps:

1. We show that in the merge regime, all possible pairs of partitions $\mathbf{P}^I$ and $\mathbf{P}^F$ partitions are equivalent up to relabeling, and hence do not need to be specified.
2. An upper bound on the maximum number of unchanged symbols is proved. We call convertible codes that meet this bound as "*stable*".
3. Lower bounds on the access cost of linear MDS convertible codes are proved, under the added restriction that the convertible codes are stable.
4. The stability restriction is removed, by showing that non-stable linear MDS convertible codes necessarily incur higher access cost, and hence it suffices to consider only stable MDS convertible codes.

In the general regime, partitions need to be specified since they indicate how message symbols from the initial codewords are mapped into the final codewords. In the merge regime, however, the choice of the partitions does not matter.

▶ **Proposition 7.** *For every $(n^I, k^I; n^F, k^F = \lambda k^I)$ convertible code, all possible pairs of initial and final partitions $(\mathcal{P}_I, \mathcal{P}_F)$ are equivalent up to relabeling.*

Proof. It holds that $M = \lambda k^I$, and there is only one possible final partition $\mathcal{P}_F = \{[\lambda k^I]\}$. Thus, all data is mapped to the same final codeword, regardless of $\mathcal{P}_I$. ◀

Since one of the terms in access cost is the number of new symbols, a natural way to reduce access cost is to maximize the number of unchanged symbols. However, there is a limit on the number of symbols that can remain unchanged.

▶ **Proposition 8.** *In an MDS* $(n^I, k^I; n^F, k^F = \lambda k^I)$ *convertible code, there can be at most* k^I *unchanged symbols from each initial codeword.*

Proof. By the MDS property of $\mathcal{C}^I$ every subset of $k^I + 1$ symbols is linearly dependent. Hence, there can be at most k^I unchanged symbols from each initial codeword for $\mathcal{C}^F$ to be MDS. ◀

This implies that there are at most λk^I unchanged symbols and at least r^F new symbols in total.

Intuitively, having more new symbols means that more symbols have to be read in order to construct them, resulting in higher access cost. With this intuition in mind, we first focus on convertible codes that minimize the number of new symbols, which we call *stable*.

▶ **Definition 9** (Stability)**.** *An MDS* $(n^I, k^I; n^F, k^F = \lambda k^I)$ *convertible code is* stable *if and only if it has exactly* λk^I *unchanged symbols, or in other words, exactly* r^F *new blocks.*

We first prove lower bounds on the access cost of stable linear MDS convertible codes, and then show that the minimum access cost of conversion in MDS codes without this stability property can only be higher. Minimizing the access cost of a stable convertible code reduces to minimizing the size of its read access set $\mathcal{D}$. The first lower bound on the size of $\mathcal{D}_i$ is given by the interaction between new symbols and the MDS property.

▶ **Lemma 10.** *For all linear stable MDS* $(n^I, k^I; n^F, k^F = \lambda k^I)$ *convertible codes, the read access set* $\mathcal{D}_i$ *from each initial codeword* $i \in [\lambda]$ *satisfies* $|\mathcal{D}_i| \geq \min\{k^I, r^F\}$.

Proof sketch. For the MDS property to hold in the final code, the encoding vectors of the new symbols must fulfill certain independence requirements. This requires reading at least as many symbol from each initial codeword as there are new symbols, up to k^I symbols. Please refer to Appendix A.2 for full proof. ◀

We next show that when the number of new symbols r^F is greater than r^I in a MDS stable convertible code in the merge regime, then the default approach is optimal in terms of access cost.

▶ **Lemma 11.** *For all linear stable MDS* $(n^I, k^I; n^F, k^F = \lambda k^I)$ *convertible codes, if* $r^I < r^F$ *then the read access set* $\mathcal{D}_i$ *from each initial codeword* $i \in [\lambda]$ *satisfies* $|\mathcal{D}_i| \geq k^I$.

Proof sketch. By Lemma 10, one is forced to read at least $r^I + 1$ symbols. Hence there exist symbols that are both unchanged and are read during conversion. Since unchanged blocks are also part of the final codeword, the information read from these symbols is not useful in creating a new symbol that retains the MDS property of the final code, unless k^I symbols (that is, full data) are read. Please refer to Appendix A.3 for full proof. ◀

Combining the above results leads to the following theorem on the lower bound of read access set size of linear stable MDS convertible codes.

▶ **Theorem 12.** *Let* $d^*(n^I, k^I; n^F, k^F)$ *denote the minimum integer* d *such that there exists a linear stable MDS* $(n^I, k^I; n^F, k^F = \lambda k^I)$ *convertible code with read access set* $\mathcal{D}$ *of size* $|\mathcal{D}| = d$*. For all valid parameters,* $d^*(n^I, k^I; n^F, k^F) \geq \lambda \min\{k^I, r^F\}$*. Furthermore, if* $r^I < r^F$*, then* $d^*(n^I, k^I; n^F, k^F) \geq \lambda k^I$.

Proof. Follows directly from Lemma 10 and Lemma 11. ◀

We next show that this lower bound generally applies even for non-stable convertible codes by proving that increasing the number of new symbols from the minimum possible does not decrease the lower bound on the size of the read access set $\mathcal{D}$.

▶ **Lemma 13.** *The lower bounds on the size of the read access set from Theorem 12 hold for* ***all*** *linear MDS* $(n^I, k^I; n^F, k^F = \lambda k^I)$ *convertible codes.*

Proof. In Appendix A.4. ◀

The above result, along with the fact that the lower bound in Theorem 12 is achievable (as will be shown in Section 4), implies that all access-optimal linear MDS convertible codes in the merge regime are stable.

▶ **Lemma 14.** *All access-optimal linear MDS* $(n^I, k^I; n^F, k^F = \lambda k^I)$ *convertible codes are stable.*

Proof. In Appendix A.5. ◀

Thus, for MDS convertible codes in the merge regime, it suffices to focus only on stable codes. Combining all the results above, leads to the following key result.

▶ **Theorem 15.** *For all linear MDS* $(n^I, k^I; n^F, k^F = \lambda k^I)$ *convertible codes, the access cost of conversion is at least* $r^F + \lambda \min\{k^I, r^F\}$. *Furthermore, if* $r^I < r^F$, *the access cost of conversion is at least* $r^F + \lambda k^I$.

Proof. Follows from Theorem 12, Lemma 13, Lemma 14, and the definition of access cost. ◀

Next, in Section 4 we show that the lower bound of Theorem 15 is achievable for all parameters. Thus, Theorem 15 implies that it is possible to perform conversion of MDS convertible codes in the merge regime with significantly less access cost than the default approach if and only if $r^F \leq r^I$ and $r^F < k^I$.

▶ Remark 16. As discussed in Section 1, existing works [44, 59, 29] on code conversion model the problem within the framework of repair-efficient codes and optimize for network bandwidth consumed during conversion. These works require accessing every symbol, i.e. n^F symbols in total, during conversion. Our approach, thus, provides a significant reduction in terms of access cost in comparison to existing solutions for code conversion. To compare in terms of network bandwidth, consider each symbol to be a vector of size α as in these existing works. Then, our constructions (Section 4) can be easily implemented in a way that only requires a network bandwidth of $(\lambda - 1)r^F\alpha$ in the merge regime, by independently constructing each new symbol at the same location as one of the retired symbols it depends on. On the other hand, existing solutions [44, 59, 29] require network bandwidth of at least $[(\lambda - 1)k^I + r^F - r^I]\alpha$. Thus, although our constructions are optimized for access cost and not network bandwidth, they outperform existing solutions for several parameter regimes of practical interest, such as the merge regime with $r^I = r^F < k^I$ which corresponds to increasing the code rate for a code with rate greater than 0.5 (most storage systems use codes with rate greater than 0.5 and a conversion that increases the rate has been shown to be beneficial when a reduction in failure rate of storage devices is observed [32]).

4 Achievability: Explicit access-optimal convertible codes in the merge regime

In this section, we present an explicit construction of access-optimal MDS convertible codes for all parameters in the merge regime. In Section 4.1, we describe the construction of the generator matrices for the initial and final code. Then, in Section 4.2, we describe sufficient conditions for optimality and show that this construction yields access-optimal convertible codes.

4.1 Explicit construction

Recall that, in the merge regime, $k^F = \lambda k^I$, for an integer $\lambda \geq 2$ and arbitrary n^I and n^F. Also, recall that $r^I = n^I - k^I$ and $r^F = n^F - k^F$. Notice that when $r^I < r^F$, or $k^I \leq r^F$, constructing an access-optimal convertible code is trivial, since one can simply use the default approach. Thus, assume $r^F \leq \min\{r^I, k^I\}$.

Let $\mathbb{F}_q$ be a finite field of size $q = p^D$, where p is any prime and the degree D is a function of the convertible code parameters, which grows as $\Theta((n^F)^3)$ when $n^F > n^I$ and the rate of the final code is constant. Let θ be a primitive element of $\mathbb{F}_q$. Let $\mathbf{G}^I = [\mathbf{I}|\mathbf{P}^I]$ and $\mathbf{G}^F = [\mathbf{I}|\mathbf{P}^F]$ be systematic generator matrices of $\mathcal{C}^I$ and $\mathcal{C}^F$ over $\mathbb{F}_q$, where $\mathbf{P}^I$ is a $k^I \times r^I$ matrix and $\mathbf{P}^F$ is a $k^F \times r^F$ matrix. Define entry (i, j) of $\mathbf{P}^I \in \mathbb{F}_q^{k^I \times r^I}$ as $\theta^{(i-1)(j-1)}$, where (i, j) ranges over $[k^I] \times [r^I]$. Entry (i, j) of $\mathbf{P}^F \in \mathbb{F}_q^{k^F \times r^F}$ is defined identically as $\theta^{(i-1)(j-1)}$, where (i, j) ranges over $[k^F] \times [r^F]$. Notice that this construction is stable, because it is access-optimal (recall Lemma 14). The unchanged symbols of the initial code are exactly the systematic symbols.

4.2 Proof of optimality

Throughout this section, we use the following notation for submatrices: let M be a $n \times m$ matrix, the submatrix of M defined by row indices $\{i_1, \ldots, i_a\}$ and column indices $\{j_1, \ldots, j_b\}$ is denoted by $M[i_1, \ldots, i_a; j_1, \ldots, j_b]$. For conciseness, we use $*$ to denote all row or column indices, e.g., $M[*; j_1, \ldots, j_b]$ denotes the submatrix composed by columns $\{j_1, \ldots, j_b\}$, and $M[i_1, \ldots, i_a; *]$ denotes the submatrix composed by rows $\{i_1, \ldots, i_a\}$.

We first recall an important fact about systematic generator matrices of MDS codes.

▶ **Proposition 17** ([34]). *Let $\mathcal{C}$ be an $[n, k]$ code with generator matrix $G = [I|P]$. Then $\mathcal{C}$ is MDS if and only if P is superregular, that is, every square submatrix of P is nonsingular*[3].

Thus, to be MDS, both $\mathbf{P}^I$ and $\mathbf{P}^F$ need to be superregular.

To be access-optimal during conversion in the non-trivial case, the new symbols (corresponding to the columns of $\mathbf{P}^F$) have to be such that they can be generated by accessing r^F symbols from the initial codewords (corresponding to columns of $\mathbf{G}^I$).

During conversion, the encoding vectors of symbols from the initial codewords are represented as λk^I-dimensional vectors, where each initial codeword occupies a disjoint subset of k^I coordinates. To capture this property, we introduce the following definition.

[3] This definition of superregularity is stronger than the definition introduced in [19] in the context of convolutional codes.

▶ **Definition 18** (t-column block-constructible)**.** *We will say that an $n \times m_1$ matrix M_1 is* t-column constructible *from an $n \times m_2$ matrix M_2 if and only if there exists a subset $S \subseteq \text{cols}(M_2)$ of size t, such that the m_1 columns of M_1 are in the span of S. We say that a $\lambda n \times m_1$ matrix M_1 is* t-column block-constructible *from an $n \times m_2$ matrix M_2 if and only if for every $i \in [\lambda]$, the submatrix $M_1[(i-1)n+1, \ldots, in; *]$ is t-column constructible from M_2.*

▶ **Theorem 19.** *A systematic $(n^I, k^I; n^F, k^F = \lambda k^I)$ convertible code with $k^I \times r^I$ initial parity generator matrix $\mathbf{P}^I$ and $k^F \times r^F$ final parity generator matrix $\mathbf{P}^F$ is MDS and access-optimal, if the following two conditions hold: (1) if $r^I \geq r^F$ then $\mathbf{P}^F$ is r^F-column block-constructible from $\mathbf{P}^I$, and (2) $\mathbf{P}^I, \mathbf{P}^F$ are superregular.*

Proof. Follows from Proposition 17 and the fact that $\mathbf{P}^F$ must be generated by accessing just r^F symbols from each initial codeword (Lemma 10). ◀

Thus, we can reduce the problem of proving the optimality of a systematic MDS convertible code in the merge regime to that of showing that matrices $\mathbf{P}^I$ and $\mathbf{P}^F$ satisfy the two properties mentioned in Theorem 19.

We first show that the construction specified in Section 4.1 satisfies condition (1) of Theorem 19.

▶ **Lemma 20.** *Let $\mathbf{P}^I, \mathbf{P}^F$ be as defined in Section 4.1. Then $\mathbf{P}^F$ is r^F-column block-constructible from $\mathbf{P}^I$.*

Proof sketch. Given the structure of $\mathbf{P}^I$ and $\mathbf{P}^F$, it is easy to see that $\mathbf{P}^F$ can be obtained by "vertically stacking" copies of $\mathbf{P}^I$, with their columns appropriately scaled by powers of θ. Please refer to Appendix B.1 for full proof. ◀

It only remains to show that the construction in Section 4.1 satisfies condition (2) of Theorem 19, that is, that $\mathbf{P}^I$ and $\mathbf{P}^F$ are superregular.

▶ **Lemma 21.** *Let $\mathbf{P}^I, \mathbf{P}^F$ be as defined in Section 4.1. Then $\mathbf{P}^I$ and $\mathbf{P}^F$ are superregular, for sufficiently large field size.*

Proof sketch. Consider the minors of $\mathbf{P}^I$ and $\mathbf{P}^F$ as polynomials on θ. Due to the structure of the the matrices $\mathbf{P}^I$ and $\mathbf{P}^F$ as specified in Section 4.1, all of these are non-zero polynomials which cannot have θ as a root as long as the degree of the extension field is large enough. Therefore none of the minors can be zero. Please refer to Appendix B.2 for the full proof. ◀

Combining the above results leads to the following key result on the achievability of the lower bounds on access cost derived in Section 3.

▶ **Theorem 22.** *The explicit construction provided in Section 4.1 yields access-optimal linear MDS convertible codes for all parameter values in the merge regime.*

Proof. Follows from Theorem 19, Lemma 20, and Lemma 21. ◀

The construction presented in this section is practical only for very small values of these parameters since the required field size grows exponentially with the lengths of the initial and final codes. In Section 5 we present practical low-field-size constructions.

5 Low field-size convertible codes based on superregular Hankel arrays

In this section we present alternative constructions for $(n^I, k^I; n^F, k^F = \lambda k^I)$ convertible codes that require a significantly lower (polynomial) field size than the construction presented in Section 4. The key idea behind our constructions is to take the matrices $\mathbf{P}^I$ and $\mathbf{P}^F$ as cleverly-chosen submatrices from a specially constructed triangular array of the following form:

$$T_m: \quad \begin{matrix} b_1 & b_2 & b_3 & \cdots & b_{m-1} & b_m \\ b_2 & b_3 & \cdots & \cdots & b_m & \\ b_3 & \vdots & \iddots & \iddots & & \\ \vdots & \vdots & \iddots & & & \\ b_{m-1} & b_m & & & & \\ b_m & & & & & \end{matrix} \tag{1}$$

with the property that every submatrix of T_m is superregular (the submatrix must lie completely within the triangular array). Here, (1) $b_1, \ldots, b_m$ are (not necessarily distinct) elements from $\mathbb{F}_q$, and (2) m is at most the field size q. The array T_m has Hankel form, that is, $T_m[i, j] = T_m[i-1, j+1]$, for all $i \in [2, m]$, $j \in [m-1]$. We denote T_m a *superregular Hankel array*. Such an array can be constructed by employing the algorithm proposed in [52] (where the algorithm was employed to generate generalized Cauchy matrices to construct generalized Reed-Solomon codes). This algorithm is described in Appendix D for reference, although it is not necessary for understanding the constructions in this section.

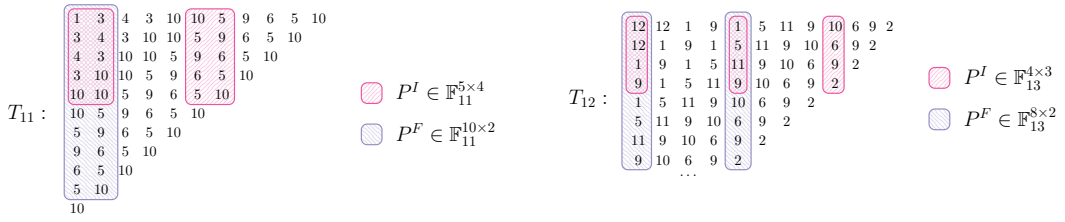
We construct the initial and final codes by taking submatrices $\mathbf{P}^I$ and $\mathbf{P}^F$ in a specific manner from superregular Hankel arrays (the submatrices have to be contained in the triangle where the array is defined). This guarantees that $\mathbf{P}^I$ and $\mathbf{P}^F$ are superregular. In addition, we exploit the Hankel form of the array by carefully choosing the submatrices that form $\mathbf{P}^I$ and $\mathbf{P}^F$ to ensure that $\mathbf{P}^F$ is r^F-column block-constructible from $\mathbf{P}^I$. Given the way we construct these matrices and the properties of T_m, all the initial and final codes presented in this subsection are (punctured) *generalized doubly-extended Reed-Solomon* codes [52].

The above idea yields a sequence of constructions with a tradeoff between the field size and the range of r^F supported. We first present two examples that correspond to the extreme ends of this tradeoff, which we call *Hankel-I* and *Hankel-II*. Construction Hankel-I, shown in Example 23, can be applied whenever $r^F \leq \lfloor r^I/\lambda \rfloor$, and requires a field size of $q \geq \max\{n^I, n^F\} - 1$. Construction Hankel-II, shown in Example 24, can be applied whenever $r^F \leq r^I - \lambda + 1$, and requires a field size of $q \geq k^I r^I$.

We then describe in detail the sequence of constructions that define a tradeoff between field size and coverage of r^F values in Section 5.1. In Section 5.2, we finalize with a discussion on the ability of these constructions to be optimal even when parameters of the final code are a priori unknown. Throughout this section we will assume that $\lambda \leq r^I \leq k^I$. The ideas presented here are still applicable when $r^I > k^I$, but the constructions and analysis change in minor ways.

▶ **Example 23** (Hankel-I)**.** Consider the parameters $(n^I = 9, k^I = 5; n^F = 12, k^F = 10)$ and the field $\mathbb{F}_{11}$. Notice that these parameters satisfy:

$$r^F = 2 \leq \left\lfloor \frac{r^I}{\lambda} \right\rfloor = 2 \quad \text{and} \quad q = 11 \geq \max\{n^I, n^F\} - 1 = 11.$$



(a) Hankel-I **(b)** Hankel-II

Figure 2 Examples of constructions based on Hankel arrays: (a) Hankel-I construction parity generator matrices for systematic $(n^I = 9, k^I = 5; n^F = 12, k^F = 10)$ convertible code. Notice how matrix $\mathbf{P}^F$ corresponds to the vertical concatenation of the first two columns and the last two columns of matrix $\mathbf{P}^I$. (b) Hankel-II construction parity generator matrices for systematic $(n^I = 7, k^I = 4; n^F = 10, k^F = 8)$ convertible code. Notice how matrix $\mathbf{P}^F$ corresponds to the vertical concatenation of the first and second column of $\mathbf{P}^I$, and the second and third column of $\mathbf{P}^I$.

First, construct a superregular Hankel array of size $n^F - 1 = 11$, T_{11}, employing the algorithm in [52]. Then, divide the $r^I = 4$ initial parities into $\lambda = 2$ groups: encoding vectors of parities in the same group will correspond to contiguous columns of T_{11}. The submatrix $\mathbf{P}^I \in \mathbb{F}_{11}^{5\times 4}$ is formed from the top $k^I = 5$ rows and columns $1, 2, k^I + 1 = 6$ and $k^I + 2 = 7$ of T_{11}, as shown in Figure 2a. The submatrix $\mathbf{P}^F \in \mathbb{F}_{11}^{10\times 2}$ is formed from the top $k^I = 10$ rows and columns $1, 2$ of T_{11}, as shown in Figure 2a. Checking that these matrices are superregular follows from the superregularity of T_{11}. It is to check that both these matrices are superregular, which follows from the the superregularity of T_{11}. Furthermore, notice that the chosen parity matrices have the following structure:

$$\mathbf{P}^I = \begin{bmatrix} | & | & | & | \\ \mathbf{p}_1 & \mathbf{p}_2 & \mathbf{p}_3 & \mathbf{p}_4 \\ | & | & | & | \end{bmatrix}, \qquad \mathbf{P}^F = \begin{bmatrix} \mathbf{p}_1 & \mathbf{p}_2 \\ \mathbf{p}_3 & \mathbf{p}_4 \end{bmatrix}.$$

From this structure, it is clear that $\mathbf{P}^F$ is 2-column block-constructible from $\mathbf{P}^I$. Therefore, $\mathbf{P}^I$ and $\mathbf{P}^F$ satisfy the sufficient conditions of Theorem 19, and define an access-optimal convertible code.

▶ **Example 24** (Hankel-II). Consider parameters $(n^I = 7, k^I = 4; n^F = 10, k^F = 8)$ and field $\mathbb{F}_{13}$. Notice that these parameters satisfy:

$$r^F = 2 \leq r^I - \lambda + 1 = 2 \quad \text{and} \quad q = 13 \geq k^I r^I = 12$$

First, construct a superregular Hankel array of size $k^I r^I = 12$, T_{12}, by choosing $q = 13$ as the field size, and employing the algorithm in [52]. The submatrix $\mathbf{P}^I \in \mathbb{F}_{13}^{4\times 3}$ is formed by the top $k^I = 4$ rows and columns $1, k^I + 1 = 5$ and $2k^I + 1 = 9$ of T_{12}, as shown in Figure 2b. The submatrix $\mathbf{P}^F \in \mathbb{F}_{13}^{8\times 2}$ is formed by the top $k^F = 8$ rows and columns 1 and $k^I + 1 = 5$ of T_{12}, as shown in Figure 2b. It is easy to check that $\mathbf{P}^I$ and $\mathbf{P}^F$ are superregular, which follows from the superregularity of T_{12}. Furthermore, notice that the chosen parity matrices have the following structure:

$$\mathbf{P}^I = \begin{bmatrix} | & | & | \\ \mathbf{p}_1 & \mathbf{p}_2 & \mathbf{p}_3 \\ | & | & | \end{bmatrix}, \qquad \mathbf{P}^F = \begin{bmatrix} \mathbf{p}_1 & \mathbf{p}_2 \\ \mathbf{p}_2 & \mathbf{p}_3 \end{bmatrix}$$

It is easy to see that $\mathbf{P}^F$ is 2-column block-constructible from $\mathbf{P}^I$. Therefore, $\mathbf{P}^I$ and $\mathbf{P}^F$ satisfy the sufficient conditions of Theorem 19, and define an access-optimal convertible code.

5.1 General Hankel-array-based construction of convertible codes

In this subsection, we present a sequence of Hankel-array-based constructions of access-optimal MDS convertible codes. This sequence of constructions presents a tradeoff between field size and the range of r^F supported. To index the sequence we use a $s \in \{\lambda, \lambda+1, \ldots, r^I\}$ which corresponds to the number of groups into which the initial parity encoding vectors are divided. Given parameters k^I, r^I, λ and a field $\mathbb{F}_q$, construction Hankel_s ($s \in \{\lambda, \lambda+1, \ldots, r^I\}$) supports:

$$r^F \leq (s-\lambda+1)\left\lfloor \frac{r^I}{s} \right\rfloor + \max\{(r^I \bmod s) - \lambda + 1, 0\}, \text{ requiring } q \geq \max\{sk^I + \left\lfloor \frac{r^I}{s} \right\rfloor - 1, n^I - 1\}.$$

Therefore, Hankel-I, from Example 23 corresponds to Hankel_λ and Hankel-II from Example 24 corresponds to Hankel_{r^I}.

Construction of Hankel_s. Assume, for the sake of simplicity, that $s \mid r^I$ and let $t = r^I/s$. Now we describe how to construct $\mathbf{P}^I$ and $\mathbf{P}^F$ over a field $\mathbb{F}_q$ whenever:

$$r^F \leq (s-\lambda+1)t \quad \text{and} \quad q \geq sk^I + t - 1.$$

Without loss of generality, we consider $r^F = (s-\lambda+1)t$ (lesser values of r^F can be obtained by puncturing the final code, i.e., eliminating some of the final parities). Let T_m be as in Equation (1), with $m = sk^I + t - 1$. Divide the r^I initial parity encoding vectors into s disjoint sets $S_1, S_2, \ldots, S_s$ of size t each. We associate each set S_i ($i \in [s]$) with a set of column indices $\text{col}(S_i) = \{(i-1)k^I+1, (i-1)k^I+2, \ldots, (i-1)k^I+t\}$ of T_m. Matrix $\mathbf{P}^I$ is the submatrix formed by the top k^I rows and the columns indexed by the set $\text{col}(S_1) \cup \cdots \cup \text{col}(S_s)$ of T_m. Matrix $\mathbf{P}^F$ is the submatrix formed by the top λk^I rows and the columns indexed by the set $\text{col}(S_1) \cup \cdots \cup \text{col}(S_{s-\lambda+1})$ of T_m. This results in the following matrices $\mathbf{P}^I$ and $\mathbf{P}^F$:

$$\mathbf{P}^I = \left[\begin{array}{ccc|c|ccc|c|ccc} b_1 & \cdots & b_t & \cdots & b_{(i-1)k^I+1} & \cdots & b_{(i-1)k^I+t} & \cdots & b_{(s-1)k^I+1} & \cdots & b_{(s-1)k^I+t} \\ b_2 & \cdots & b_{t+1} & \cdots & b_{(i-1)k^I+2} & \cdots & b_{(i-1)k^I+t+1} & \cdots & b_{(s-1)k^I+2} & \cdots & b_{(s-1)k^I+t+1} \\ \vdots & \vdots & \vdots & \cdots & \vdots & \vdots & \vdots & \cdots & \vdots & \vdots & \vdots \\ b_{k^I} & \cdots & b_{k^I+t-1} & \cdots & b_{ik^I} & \cdots & b_{ik^I+t-1} & \cdots & b_{sk^I} & \cdots & b_{sk^I+t-1} \end{array}\right],$$

$$\mathbf{P}^F = \left[\begin{array}{ccc|c|ccc|c|ccc} b_1 & \cdots & b_t & \cdots & b_{(i-\lambda)k^I+1} & \cdots & b_{(i-\lambda)k^I+t} & \cdots & b_{(s-\lambda)k^I+1} & \cdots & b_{(s-\lambda)k^I+t} \\ b_2 & \cdots & b_{t+1} & \cdots & b_{(i-\lambda)k^I+2} & \cdots & b_{(i-\lambda)k^I+t+1} & \cdots & b_{(s-\lambda)k^I+2} & \cdots & b_{(s-\lambda)k^I+t+1} \\ \vdots & \vdots & \vdots & \cdots & \vdots & \vdots & \vdots & \cdots & \vdots & \vdots & \vdots \\ b_{\lambda k^I} & \cdots & b_{\lambda k^I+1} & \cdots & b_{ik^I} & \cdots & b_{ik^I+t-1} & \cdots & b_{sk^I} & \cdots & b_{sk^I+t-1} \end{array}\right].$$

▶ **Theorem 25.** *Given parameters k^I, r^I, λ, and a field $\mathbb{F}_q$ Hankel_s ($s \in \{\lambda, \ldots, r^I\}$) constructs an access-optimal $(n^I, k^I; n^F, k^F = \lambda k^I)$ convertible code if:*

$$r^F \leq (s-\lambda+1)\left\lfloor \frac{r^I}{s} \right\rfloor + \max\{(r^I \bmod s) - \lambda + 1, 0\} \quad \textit{and} \quad q \geq sk^I + \left\lfloor \frac{r^I}{s} \right\rfloor - 1.$$

Proof. In Appendix C.1. ◀

(Access-optimal) conversion process. During conversion, the k^I data symbols from each of the λ initial codewords remain unchanged, and become the $k^F = \lambda k^I$ data symbols from the final codeword. The r^F new (parity) blocks from the final codeword are constructed by accessing symbols from the initial codewords as detailed below. To construct the l-th new

symbol (corresponding to the l-th column of $\mathbf{P}^F$, $l \in [r^F]$), read parity symbol $l + (i-1)t$ from each initial codeword $i \in [\lambda]$, and then sum the λ symbols read. The encoding vector of the new symbol will be equal to the sum of the encoding vectors of the symbols read. This is done for every new encoding vector $l \in [r^F]$.

5.2 Handling a priori unknown parameters

So far, we had assumed that the parameters of the final code, n^F, k^F, are known a priori and are fixed. As discussed in Section 2, this is useful in developing an understanding of the fundamental limits of code conversion. When realizing code conversion in practice, however, the parameters n^F, k^F might not be known at code construction time (as it depends on the empirically observed failure rates). Thus, it is of interest to be able to *convert a code optimally to multiple different parameters*. The Hankel-array based constructions presented above indeed provide such a flexibility. Our constructions continue to enable access-optimal conversion for any $k^{F'} = \lambda' k^I$ and $n^{F'} = r^{F'} + k^{F'}$ with $0 \leq r^{F'} \leq r^F$ and $2 \leq \lambda' \leq \lambda$.

6 Related work

MDS erasure codes, such as Reed-Solomon codes, are widely used in storage systems because they achieve the optimal tradeoff between failure tolerance and storage overhead [43, 42]. However, the use of erasure codes in storage systems raises a host of other challenges. Several works in the literature have studied these aspects.

The encoding and decoding of data, and the finite field arithmetic that they require, can be compute intensive. Motivated by this, *array codes* [7, 70, 31, 28] are designed to use XOR operations exclusively, which are typically faster to execute, and aim to decrease the complexity of encoding and decoding.

The repair of failed nodes can require a large amount of network bandwidth. Several approaches have been proposed to alleviate this problem. Dimakis et al. [16] proposed a new class of codes called *regenerating codes* that minimize the amount of network bandwidth consumed during repair operations. Under the regenerating codes model [16], each symbol (i.e., node) is represented as an α-dimensional vector over a finite field. During repair of a failed node, download of elements of the finite field (i.e., "sub-symbols") is allowed as opposed to the whole vector (i.e., one "entire" symbol). This line of research has led to several constructions [10, 47, 55, 60, 68, 69, 64, 11, 40, 53, 71, 73, 50, 54, 22, 13, 35, 36], generalizations [56, 58, 1], and more efficient repair algorithms for Reed-Solomon codes [57, 26, 72, 15, 66, 37, 14, 67]. It has been shown that meeting the lower bound on the repair bandwidth requirement when MDS property and high rate are desired necessitates large sub-packetization [65, 23, 5, 3], which negatively affects certain key performance metrics in storage systems [45]. To overcome this issue, several works [49, 25] have proposed code constructions that relax the requirement of meeting lower bounds on IO and bandwidth requirements for repair operations. For example, the Piggybacking framework [49] provides a general framework to construct repair-efficient codes by transforming any existing codes, while allowing a small sub-packetization (even as small as 2).

Several works [46, 39] study the problem of two stage encoding: first generating a certain number of parities during the encoding process and then adding additional parities. As discussed in [46], adding additional parities can be conceptually viewed as a repair process by considering the new parity nodes to be generated as failed nodes. Furthermore, as shown in [55], for MDS codes, the bandwidth requirement for *repair of even a single node* is lower bounded by the same amount as in regenerating codes that require repair of *all* nodes. Thus one can always employ a regenerating code to add additional parities with minimum

bandwidth overhead. However, when MDS property and high rate are desired, as discussed above, using regenerating codes requires a large sub-packetization. The paper [39] employs the Piggybacking framework [48, 49] to construct codes that reduce the sub-packetization factor for two-stage encoding. The scenario of adding a fixed number of additional parities, when viewed under the setting of conversions, corresponds to having $k^I = k^F$ and $n^I < n^F$.

Existing works that consider changing the parameters n and k of already encoded data [44, 59, 29] consider a model similar to the regenerating codes model, wherein symbols are represented as vectors and communication is modeled via an information flow graph. In order to accommodate the changes in parameter k, the dimension α of each vector is changed, and constructions exploit repair efficiency to achieve conversion efficiency. However, this approach has the disadvantage that during conversion every symbol must be accessed. Our approach circumvents this problem by considering multiple codewords at a time, which allows convertible codes to achieve significantly smaller access cost during conversion. For a more detailed comparison between convertible codes and existing works on code conversion, see Remark 16 in Section 3.

Another class of codes, called *locally repairable codes* (LRCs) [21, 51, 8, 20, 41, 61, 33, 12, 63, 62, 6, 17, 2, 38, 27], focuses on the locality of codeword symbols during repair, that is, the number of nodes that need to be accessed when repairing a single failure. LRCs improve repair and degraded read performance, since missing information can be recovered by accessing a small subset of symbols. The objective of LRCs and convertible codes optimized for access cost is similar, as both aim to minimize the number of symbols that need to be accessed for different operations in storage systems.

7 Conclusions and future directions

In this paper, we propose a new framework to model the code conversion problem, that of converting data encoded with an $[n^I, k^I]$ code into data encoded with an $[n^F, k^F]$ code in a resource-efficient manner. The code conversion problem is motivated by the practical necessity of reducing the overhead of redundancy adaptation in erasure-coded storage systems [32]. We present the framework of convertible codes for studying code conversions, and fully characterize the fundamental limits on the access cost of conversions for an important regime of convertible codes. Furthermore, we present practical low-field-size constructions for access-optimal convertible codes for a wide range of parameters.

This work leads to a number of challenging and potentially impactful open problems. An important future direction is to go beyond the merge regime considered in this paper. While the construction techniques presented in this paper can be easily extended to upper bound the access cost of some parameter values outside the merge regime, identifying the fundamental limits on the access cost in general and constructing access-optimal convertible codes for all parameters remains open. Another important future direction is to analyze the fundamental limits of convertible codes on the overhead of other resources, such as disk IO (i.e., device bandwidth), communication (network bandwidth), computation (CPU), and construct convertible codes optimizing these resources. Note that while the access-optimal convertible codes considered in this paper also reduce the total disk IO, communication and computation during conversion as compared to the default approach, the overhead on these other resources may not be optimal. A final important direction is to explore additional applications of convertible codes beyond our initial motivation, to other problems within theoretical computer science where codes are used.

References

1 Vitaly Abdrashitov, N Prakash, and Muriel Médard. The storage vs repair bandwidth trade-off for multiple failures in clustered storage networks. In *2017 IEEE Information Theory Workshop (ITW)*, pages 46–50. IEEE, 2017.

2 Abhishek Agarwal, Alexander Barg, Sihuang Hu, Arya Mazumdar, and Itzhak Tamo. Combinatorial alphabet-dependent bounds for locally recoverable codes. *IEEE Transactions on Information Theory*, 64(5):3481–3492, 2018.

3 Omar Alrabiah and Venkatesan Guruswami. An Exponential Lower Bound on the Sub-packetization of MSR Codes. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2019, pages 979–985, New York, NY, USA, 2019. ACM.

4 Apache Software Foundation. Apache hadoop: HDFS erasure coding. Accessed: 2019-07-23. URL: `https://hadoop.apache.org/docs/r3.0.0/hadoop-project-dist/hadoop-hdfs/HDFSErasureCoding.html`.

5 SB Balaji and P Vijay Kumar. A tight lower bound on the sub-packetization level of optimal-access MSR and MDS codes. In *2018 IEEE International Symposium on Information Theory (ISIT)*, pages 2381–2385. IEEE, 2018.

6 Alexander Barg, Kathryn Haymaker, Everett W Howe, Gretchen L Matthews, and Anthony Várilly-Alvarado. Locally recoverable codes from algebraic curves and surfaces. In *Algebraic Geometry for Coding Theory and Cryptography*, pages 95–127. Springer, 2017.

7 M. Blaum, J. Brady, J. Bruck, and Jai Menon. EVENODD: an efficient scheme for tolerating double disk failures in RAID architectures. *IEEE Transactions on Computers*, 44(2):192–202, February 1995.

8 Mario Blaum, James Lee Hafner, and Steven Hetzler. Partial-MDS codes and their application to RAID type of architectures. *IEEE Transactions on Information Theory*, 59(7):4510–4519, 2013.

9 Dhruba Borthakur, Rodrigo Schmidt, Ramkumar Vadali, Scott Chen, and Patrick Kling. HDFS RAID - Facebook. URL: `http://www.slideshare.net/ydn/hdfs-raid-facebook`.

10 Viveck R Cadambe, Cheng Huang, Jin Li, and Sanjeev Mehrotra. Polynomial length MDS codes with optimal repair in distributed storage. In *2011 Conference Record of the Forty Fifth Asilomar Conference on Signals, Systems and Computers (ASILOMAR)*, pages 1850–1854. IEEE, 2011.

11 Viveck R Cadambe, Syed A Jafar, Hamed Maleki, Kannan Ramchandran, and Changho Suh. Asymptotic interference alignment for optimal repair of MDS codes in distributed storage. *IEEE Transactions on Information Theory*, 59(5):2974–2987, 2013.

12 Viveck R Cadambe and Arya Mazumdar. Bounds on the size of locally recoverable codes. *IEEE Transactions on Information Theory*, 61(11):5787–5794, 2015.

13 Ameera Chowdhury and Alexander Vardy. New Constructions of MDS Codes with Asymptotically Optimal Repair. In *2018 IEEE International Symposium on Information Theory*, pages 1944–1948, 2018.

14 Hoang Dau, Iwan M Duursma, Han Mao Kiah, and Olgica Milenkovic. Repairing Reed-Solomon codes with multiple erasures. *IEEE Transactions on Information Theory*, 64(10):6567–6582, 2018.

15 Hoang Dau and Olgica Milenkovic. Optimal repair schemes for some families of full-length Reed-Solomon codes. In *2017 IEEE International Symposium on Information Theory (ISIT)*, pages 346–350. IEEE, 2017.

16 A. G. Dimakis, P. B. Godfrey, Y. Wu, M. J. Wainwright, and K. Ramchandran. Network Coding for Distributed Storage Systems. *IEEE Transactions on Information Theory*, 56(9):4539–4551, September 2010.

17 S Luna Frank-Fischer, Venkatesan Guruswami, and Mary Wootters. Locality via Partially Lifted Codes. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

18 S. Ghemawat, H. Gobioff, and S.T. Leung. The Google file system. In *ACM SIGOPS Operating Systems Review*, volume 37-5, pages 29–43. ACM, 2003.

19 H. Gluesing-Luerssen, J. Rosenthal, and R. Smarandache. Strongly-MDS convolutional codes. *IEEE Transactions on Information Theory*, 52(2):584–598, February 2006.

20 Parikshit Gopalan, Cheng Huang, Bob Jenkins, and Sergey Yekhanin. Explicit maximally recoverable codes with locality. *IEEE Transactions on Information Theory*, 60(9):5245–5256, 2014.

21 Parikshit Gopalan, Cheng Huang, Huseyin Simitci, and Sergey Yekhanin. On the locality of codeword symbols. *IEEE Transactions on Information Theory*, 58(11):6925–6934, 2012.

22 Sreechakra Goparaju, Arman Fazeli, and Alexander Vardy. Minimum Storage Regenerating Codes for All Parameters. *IEEE Transactions on Information Theory*, 63(10):6318–6328, 2017.

23 Sreechakra Goparaju, Itzhak Tamo, and Robert Calderbank. An improved sub-packetization bound for minimum storage regenerating codes. *IEEE Transactions on Information Theory*, 60(5):2770–2779, 2014.

24 Sivakanth Gopi, Venkatesan Guruswami, and Sergey Yekhanin. Maximally Recoverable LRCs: A field size lower bound and constructions for few heavy parities. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2154–2170. SIAM, 2019.

25 Venkatesan Guruswami and Ankit Singh Rawat. MDS code constructions with small sub-packetization and near-optimal repair bandwidth. In *ACM-SIAM Symposium on Discrete Algorithms*, 2017.

26 Venkatesan Guruswami and Mary Wootters. Repairing Reed-Solomon codes. In *ACM Symposium on Theory of Computing*, pages 216–226, 2016.

27 Venkatesan Guruswami, Chaoping Xing, and Chen Yuan. How Long Can Optimal Locally Repairable Codes Be? In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

28 James Lee Hafner. WEAVER codes: Highly fault tolerant erasure codes for storage systems. In *Proceedings of the 4th Conference on USENIX Conference on File and Storage Technologies - Volume 4*, FAST'05, pages 16–16, Berkeley, CA, USA, 2005. USENIX Association.

29 Yuchong Hu, Xiaoyang Zhang, Patrick PC Lee, and Pan Zhou. Generalized Optimal Storage Scaling via Network Coding. In *2018 IEEE International Symposium on Information Theory (ISIT)*, pages 956–960. IEEE, 2018.

30 C. Huang, H. Simitci, Y. Xu, A. Ogus, B. Calder, P. Gopalan, J. Li, and S. Yekhanin. Erasure Coding in Windows Azure Storage. In *Proceedings of USENIX Annual Technical Conference (ATC)*, 2012.

31 Cheng Huang and Lihao Xu. STAR: An efficient coding scheme for correcting triple storage node failures. *IEEE Transactions on Computers*, 57(7):889–901, 2008.

32 Saurabh Kadekodi, K. V. Rashmi, and Gregory R. Ganger. Cluster storage systems gotta have HeART: improving storage efficiency by exploiting disk-reliability heterogeneity. *USENIX FAST*, 2019.

33 Govinda M Kamath, N Prakash, V Lalitha, and P Vijay Kumar. Codes with local regeneration and erasure correction. *IEEE Transactions on Information Theory*, 60(8):4637–4660, 2014.

34 F.J. MacWilliams and N.J.A. Sloane. *The Theory of Error-Correcting Codes*. North-holland Publishing Company, 2nd edition, 1978.

35 Kaveh Mahdaviani, Ashish Khisti, and Soheil Mohajer. Bandwidth adaptive & error resilient MBR exact repair regenerating codes. *IEEE Transactions on Information Theory*, 65(5):2736–2759, 2018.

36 Kaveh Mahdaviani, Soheil Mohajer, and Ashish Khisti. Product matrix MSR codes with bandwidth adaptive exact repair. *IEEE Transactions on Information Theory*, 64(4):3121–3135, 2018.

37 Jay Mardia, Burak Bartan, and Mary Wootters. Repairing multiple failures for scalar MDS codes. *IEEE Transactions on Information Theory*, 65(5):2661–2672, 2018.

38 A. Mazumdar. Capacity of Locally Recoverable Codes. In *2018 IEEE Information Theory Workshop*, pages 1–5, November 2018.

39 S. Mousavi, T. Zhou, and C. Tian. Delayed Parity Generation in MDS Storage Codes. In *2018 IEEE International Symposium on Information Theory (ISIT)*, pages 1889–1893, June 2018.

40 D Papailiopoulos, A Dimakis, and V Cadambe. Repair Optimal Erasure Codes through Hadamard Designs. *IEEE Transactions on Information Theory*, 59(5):3021–3037, May 2013.

41 Dimitris S Papailiopoulos and Alexandros G Dimakis. Locally repairable codes. *IEEE Transactions on Information Theory*, 60(10):5843–5855, 2014.

42 David A Patterson, Garth Gibson, and Randy H Katz. *A case for redundant arrays of inexpensive disks (RAID)*, volume 17-3. ACM, 1988.

43 J.S. Plank. T1: Erasure codes for storage applications. *Proceedings of the 4th USENIX Conference on File and Storage Technologies*, pages 1–74, January 2005.

44 Brijesh Kumar Rai, Vommi Dhoorjati, Lokesh Saini, and Amit K Jha. On adaptive distributed storage systems. In *2015 IEEE international symposium on information theory (ISIT)*, pages 1482–1486. IEEE, 2015.

45 K. V. Rashmi, Nihar B Shah, Dikang Gu, Hairong Kuang, Dhruba Borthakur, and Kannan Ramchandran. A Hitchhiker's guide to fast and efficient data reconstruction in erasure-coded data centers. In *ACM SIGCOMM*, 2014.

46 K. V. Rashmi, Nihar B Shah, and P Vijay Kumar. Enabling node repair in any erasure code for distributed storage. In *2011 IEEE International Symposium on Information Theory Proceedings*, pages 1235–1239. IEEE, 2011.

47 K. V. Rashmi, Nihar B Shah, and P Vijay Kumar. Optimal exact-regenerating codes for distributed storage at the MSR and MBR points via a product-matrix construction. *IEEE Transactions on Information Theory*, 57(8):5227–5239, 2011.

48 K. V. Rashmi, Nihar B Shah, and Kannan Ramchandran. A piggybacking design framework for read-and download-efficient distributed storage codes. In *2013 IEEE International Symposium on Information Theory*, 2013.

49 K. V. Rashmi, Nihar B Shah, and Kannan Ramchandran. A piggybacking design framework for read-and download-efficient distributed storage codes. *IEEE Transactions on Information Theory*, 63(9):5802–5820, 2017.

50 Ankit Singh Rawat, O Ozan Koyluoglu, and Sriram Vishwanath. Progress on high-rate MSR codes: Enabling arbitrary number of helper nodes. In *2016 Information Theory and Applications Workshop (ITA)*, pages 1–6. IEEE, 2016.

51 Ankit Singh Rawat, Onur Ozan Koyluoglu, Natalia Silberstein, and Sriram Vishwanath. Optimal locally repairable and secure codes for distributed storage systems. *IEEE Transactions on Information Theory*, 60(1):212–236, 2013.

52 Ron M Roth and Gadiel Seroussi. On Generator Matrices of MDS Codes. *IEEE Transactions on Infortmation Theory*, 31(6):826–830, November 1985.

53 Birenjith Sasidharan, Gaurav Kumar Agarwal, and P Vijay Kumar. A high-rate MSR code with polynomial sub-packetization level. In *2015 IEEE International Symposium on Information Theory (ISIT)*, pages 2051–2055. IEEE, 2015.

54 Birenjith Sasidharan, Myna Vajha, and P Vijay Kumar. An explicit, coupled-layer construction of a high-rate MSR code with low sub-packetization level, small field size and d<(n- 1). In *2017 IEEE International Symposium on Information Theory (ISIT)*, pages 2048–2052. IEEE, 2017.

55 Nihar B Shah, K. V. Rashmi, P Vijay Kumar, and Kannan Ramchandran. Interference alignment in regenerating codes for distributed storage: Necessity and code constructions. *IEEE Transactions on Information Theory*, 58(4):2134–2158, 2011.

56 Nihar B Shah, KV Rashmi, and P Vijay Kumar. A flexible class of regenerating codes for distributed storage. In *2010 IEEE International Symposium on Information Theory*, pages 1943–1947. IEEE, 2010.

57 Karthikeyan Shanmugam, Dimitris S Papailiopoulos, Alexandros G Dimakis, and Giuseppe Caire. A repair framework for scalar MDS codes. *IEEE Journal on Selected Areas in Communications*, 32(5):998–1007, 2014.

58 Kenneth W Shum. Cooperative regenerating codes for distributed storage systems. In *2011 IEEE International Conference on Communications (ICC)*, pages 1–5. IEEE, 2011.

59 Mridupawan Sonowal and Brijesh Kumar Rai. On adaptive distributed storage systems based on functional MSR code. In *2017 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, pages 338–343. IEEE, 2017.

60 C. Suh and Kannan Ramchandran. Exact-Repair MDS Code Construction Using Interference Alignment. *IEEE Transactions on Information Theory*, pages 1425–1442, March 2011.

61 Itzhak Tamo and Alexander Barg. A family of optimal locally recoverable codes. *IEEE Transactions on Information Theory*, 60(8):4661–4676, 2014.

62 Itzhak Tamo, Alexander Barg, and Alexey Frolov. Bounds on the parameters of locally recoverable codes. *IEEE Transactions on Information Theory*, 62(6):3070–3083, 2016.

63 Itzhak Tamo, Dimitris S Papailiopoulos, and Alexandros G Dimakis. Optimal locally repairable codes and connections to matroid theory. *IEEE Transactions on Information Theory*, 62(12):6661–6671, 2016.

64 Itzhak Tamo, Zhiying Wang, and Jehoshua Bruck. Zigzag codes: MDS array codes with optimal rebuilding. *IEEE Transactions on Information Theory*, 59(3):1597–1616, 2013.

65 Itzhak Tamo, Zhiying Wang, and Jehoshua Bruck. Access versus bandwidth in codes for storage. *IEEE Transactions on Information Theory*, 60(4):2028–2037, 2014.

66 Itzhak Tamo, Min Ye, and Alexander Barg. Optimal repair of Reed-Solomon codes: Achieving the cut-set bound. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 216–227. IEEE, 2017.

67 Itzhak Tamo, Min Ye, and Alexander Barg. The Repair Problem for Reed–Solomon Codes: Optimal Repair of Single and Multiple Erasures With Almost Optimal Node Size. *IEEE Transactions on Information Theory*, 65(5):2673–2695, 2018.

68 Zhiying Wang, Itzhak Tamo, and Jehoshua Bruck. On codes for optimal rebuilding access. In *2011 49th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 1374–1381. IEEE, 2011.

69 Zhiying Wang, Itzhak Tamo, and Jehoshua Bruck. Long MDS codes for optimal repair bandwidth. In *2012 IEEE International Symposium on Information Theory Proceedings*, pages 1182–1186. IEEE, 2012.

70 Lihao Xu and Jehoshua Bruck. X-code: MDS array codes with optimal encoding. *IEEE Transactions on Information Theory*, 45(1):272–276, 1999.

71 M. Ye and A. Barg. Explicit Constructions of Optimal-Access MDS Codes With Nearly Optimal Sub-Packetization. *IEEE Transactions on Information Theory*, 63(10):6307–6317, October 2017.

72 Min Ye and Alexander Barg. Explicit constructions of MDS array codes and RS codes with optimal repair bandwidth. In *2016 IEEE International Symposium on Information Theory (ISIT)*, pages 1202–1206. IEEE, 2016.

73 Min Ye and Alexander Barg. Explicit constructions of high-rate MDS array codes with optimal repair bandwidth. *IEEE Transactions on Information Theory*, 63(4):2001–2014, 2017.

A Proofs of Section 3

A.1 Notation used in proofs

Let $\mathcal{C}^I$ be an $[n^I, k^I]$ MDS code over field $\mathbb{F}_q$, specified by generator matrix $\mathbf{G}^I$, with columns (that is, encoding vectors) $\{\mathbf{g}^I_1, \ldots, \mathbf{g}^I_{n^I}\} \subseteq \mathbb{F}_q^{k^I}$. Let $\lambda \geq 2$ be an integer, and let $\mathcal{C}^F$ be an $[n^F, k^F = \lambda k^I]$ MDS code over field $\mathbb{F}_q$, specified by generator matrix $\mathbf{G}^F$, with columns (that is, encoding vectors) $\{\mathbf{g}^F_1, \ldots, \mathbf{g}^F_{n^F}\} \subseteq \mathbb{F}_q^{k^F}$. Let $r^I = n^I - k^I$ and $r^F = n^F - k^F$. When $\mathcal{C}^I$ and $\mathcal{C}^F$ are systematic, r^I and r^F correspond to the initial number of parities and final number of parities, respectively. All vectors are assumed to be column vectors. We will use the notation $\mathbf{v}[l]$ to denote the l-th coordinate of a vector $\mathbf{v}$.

We will represent *all* the code symbols in the initial codewords as being generated by a single $\lambda k^I \times \lambda n^I$ matrix $\tilde{\mathbf{G}}^I$, with encoding vectors $\{\tilde{\mathbf{g}}^I_{i,j} \mid i \in [\lambda], j \in [n^I]\} \subseteq \mathbb{F}_q^{k^F}$. This representation can be viewed as embedding the column vectors of the generator matrix $\mathbf{G}^I$ in an λk^I-dimensional space, where the index set $\mathcal{K}_i = \{(i-1)k^I+1, \ldots, ik^I\}, i \in [\lambda]$ corresponds to the encoding vectors for initial codeword i. Let $\tilde{\mathbf{g}}^I_{i,j}$ denote the j-th encoding vector in the initial codeword i in this (embedded) representation. Thus, $\tilde{\mathbf{g}}^I_{i,j}[l] = \mathbf{g}^I_j[l - (i-1)k^I]$ for $l \in \mathcal{K}_i$, and $\tilde{\mathbf{g}}^I_{i,j}[l] = 0$ otherwise. As an example, Figure 3 shows the values of the defined terms for the single parity-check code from Figure 1 with $n^I = 3, k^I = 2, n^F = 5, k^F = 4$.

At times, focus will be only on the coordinates of an encoding vector of a certain initial codeword i. For this purpose, define $\text{proj}_{\mathcal{K}_i}(\mathbf{v}) \in \mathbb{F}_q^{k^I}$ to be the projection of $\mathbf{v} \in \mathbb{F}_q^{k^F}$ to the coordinates in an index set $\mathcal{K}_i$, and for a set $\mathcal{V}$ of vectors, $\text{proj}_{\mathcal{K}_i}(\mathcal{V}) = \{\text{proj}_{\mathcal{K}_i}(\mathbf{v}) \mid \mathbf{v} \in \mathcal{V}\}$. For example, $\text{proj}_{\mathcal{K}_i}(\tilde{\mathbf{g}}^I_{i,j}) = \mathbf{g}^I_j$ for all $i \in [\lambda]$ and $j \in [n^I]$.

The following sets of vectors are defined: the encoding vectors from initial codeword i, $\mathcal{S}^I_i = \{\tilde{\mathbf{g}}^I_{i,j} \mid j \in [n^I]\}$, all the encoding vectors from all the initial codewords, $\mathcal{S}^I = \cup_{i \in [\lambda]} \mathcal{S}^I_i$, and all the encoding vectors from the final codeword $\mathcal{S}^F = \{\mathbf{g}^F_j \mid j \in [n^F]\}$.

We use the term *unchanged symbols* to refer to symbols from the initial codewords that remain as is (that is, unchanged) in the final codeword. The symbols in the final codeword that were not present in the initial codewords are called *new*, and the symbols from the initial codewords that do not carry over to the final codeword are called *retired*. For example, in Figure 1, all the data symbols are unchanged symbols (unshaded boxes), the single parity symbol of the final codeword is a new symbol, and the two parity blocks from the initial codewords are retired symbols. Each unchanged symbol corresponds to a pair of identical initial and final encoding vectors, that is, a tuple of indices (i, j, l) such that $\tilde{\mathbf{g}}^I_{i,j} = \mathbf{g}^F_l$. For instance, the example in Figure 1 has four unchanged symbols, corresponding to the identical encoding vectors $\tilde{\mathbf{g}}^I_{i,j} = \mathbf{g}^F_{2(i-1)+j}$ for $i, j \in [2]$. The final encoding vectors $\mathcal{S}^F$ can thus be partitioned into the following sets: *unchanged encoding vectors* from initial codeword i, $\mathcal{U}_i = \mathcal{S}^F \cap \mathcal{S}^I_i$ for all $i \in [\lambda]$, and *new encoding vectors* $\mathcal{N} = \mathcal{S}^F \setminus \mathcal{S}^I$.

From the point of view of conversion cost, unchanged symbols are ideal, because they require no extra work. On the other hand, constructing new symbols require accessing symbols from the initial codewords. When a symbol from the initial codewords is accessed, all of its contents are downloaded to a central location, where they are available for the construction of all new symbols. For example, in Figure 1, one symbol from each initial codeword is accessed during conversion.

During conversion, new symbols are constructed by reading symbols from the initial codewords. That is, every new encoding vector is simply a linear combination of a specific subset of $\mathcal{S}^I$. Define the *read access set* for an MDS $(n^I, k^I; n^F, k^F = \lambda k^I)$ convertible code as the set of tuples $\mathcal{D} \in [\lambda] \times [n^I]$ such that the set of new encoding vectors $\mathcal{N}$ is contained in the span of the set $\{\tilde{\mathbf{g}}^I_{i,j} \mid (i,j) \in \mathcal{D}\}$. Furthermore, define the index sets $\mathcal{D}_i = \{j \mid (i,j) \in \mathcal{D}\}$, $\forall i \in [\lambda]$ which denote the encoding vectors accessed from each initial codeword.

$$\mathbf{G}^I = \begin{matrix} \mathbf{g}_1^I & \mathbf{g}_2^I & \mathbf{g}_3^I \end{matrix} \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} \quad \tilde{\mathbf{G}}^I = \begin{matrix} \tilde{\mathbf{g}}_{1,1}^I & \tilde{\mathbf{g}}_{1,2}^I & \tilde{\mathbf{g}}_{1,3}^I & \tilde{\mathbf{g}}_{2,1}^I & \tilde{\mathbf{g}}_{2,2}^I & \tilde{\mathbf{g}}_{2,3}^I \end{matrix} \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \quad \mathbf{G}^F = \begin{matrix} \mathbf{g}_1^F & \mathbf{g}_2^F & \mathbf{g}_3^F & \mathbf{g}_4^F & \mathbf{g}_5^F \end{matrix} \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

Figure 3 Generator matrices for a specific $(n^I = 3, k^I = 2; n^F = 5, k^F = 4)$ convertible code: $\mathbf{G}^I$ is the generator matrix of the initial code; $\tilde{\mathbf{G}}^I$ is the generator matrix of all initial codewords; $\mathbf{G}^F$ is the generator matrix of the final code.

A.2 Proof of Lemma 10

The notation used in this proof is introduced in Appendix A.1. By the MDS property, every subset $\mathcal{V} \in \mathcal{S}^F$ of size at most $k^F = \lambda k^I$ is linearly independent. For any initial codeword $i \in [\lambda]$, consider the set of all unchanged encoding vectors from other codewords, $\cup_{\ell \neq i} \mathcal{U}_\ell$, and pick any subset of new encoding vectors $\mathcal{W} \subseteq \mathcal{N}$ of size $|\mathcal{W}| = \min\{k^I, r^F\}$. Consider the subset $\mathcal{V} = (\cup_{\ell \neq i} \mathcal{U}_\ell \cup \mathcal{W})$: it is true that $\mathcal{V} \subseteq \mathcal{S}^F$ and $|\mathcal{V}| = (\lambda - 1)k^I + \min\{k^I, r^F\} \leq k^F$. Therefore, all the encoding vectors in $\mathcal{V}$ are linearly independent.

Notice that the encoding vectors in $\mathcal{V} \setminus \mathcal{W}$ contain no information about initial codeword i and complete information about every other initial codeword $\ell \neq i$. Therefore, the information about initial codeword i in each encoding vector in $\mathcal{W}$ has to be linearly independent since, otherwise, $\mathcal{V}$ could not be linearly independent. Formally, it must be the case that $\mathcal{W}_i = \text{proj}_{\mathcal{K}_i}(\mathcal{W})$ has rank equal to $\min\{k^I, r^F\}$ (recall from Appendix A.1 that $\mathcal{K}_i$ is the set of coordinates belonging to initial codeword i). However, by definition, the subset $\mathcal{W}_i$ must be contained in the span of $\{\mathbf{g}_j^I \mid j \in \mathcal{D}_i\}$. Therefore, the rank of $\{\mathbf{g}_j^I \mid j \in \mathcal{D}_i\}$ is at least that of $\mathcal{W}_i$, which implies that $|\mathcal{D}_i| \geq \min\{k^I, r^F\}$. ◀

A.3 Proof of Lemma 11

The notation used in this proof is introduced in Appendix A.1. When $r^F \geq k^I$, this lemma is equivalent to Lemma 10, so assume $r^I < r^F < k^I$. From the proof of Lemma 10, for every initial codeword $i \in [\lambda]$ it holds that $|\mathcal{D}_i| \geq r^F$. Since $r^F > r^I$, this implies that $\mathcal{D}_i$ must contain at least one index of an unchanged encoding vector.

Choose a subset of at most $k^F = \lambda k^I$ encoding vectors from $\mathcal{S}^F$, which must be linearly independent by the MDS property. In this subset, include all the unchanged encoding vectors from the other initial codewords, $\cup_{l \neq i} \mathcal{U}_l$. Then, choose all the unchanged encoding vectors from initial codeword i that are accessed during conversion, $\mathcal{W}_1 = (\{\tilde{\mathbf{g}}_{i,j}^I \mid j \in \mathcal{D}_i\} \cap \mathcal{U}_i)$. For the remaining vectors (if any), choose an arbitrary subset of new encoding vectors, $\mathcal{W}_2 \subseteq \mathcal{N}$, such that:

$$|\mathcal{W}_2| = \min\{k^I - |\mathcal{W}_1|, r^F\}. \tag{2}$$

It is easy to check that the subset $\mathcal{V} = \cup_{l \neq i} \mathcal{U}_l \cup \mathcal{W}_1 \cup \mathcal{W}_2$ is of size at most $k^F = \lambda k^I$, and therefore it is linearly independent. This choice of $\mathcal{V}$ follows from the idea that the information contributed by $\mathcal{W}_1$ to the new encoding vectors is already present in the unchanged encoding vectors, which will be at odds with the linear independence of $\mathcal{V}$.

Since the elements of $\mathcal{W}_1$ and $\mathcal{W}_2$ are the only encoding vectors in $\mathcal{V}$ that contain information from initial codeword i, it must be the case that $\widetilde{\mathcal{W}} = \text{proj}_{\mathcal{K}_i}(\mathcal{W}_1) \cup \text{proj}_{\mathcal{K}_i}(\mathcal{W}_2)$ has rank $|\mathcal{W}_1| + |\mathcal{W}_2|$. Moreover, $\widetilde{\mathcal{W}}$ is contained in the span of $\{\mathbf{g}_j^I \mid j \in \mathcal{D}_i\}$ by definition, so it holds that:

$$|\mathcal{D}_i| \geq |\mathcal{W}_1| + |\mathcal{W}_2|. \tag{3}$$

From Equation (2), there are two cases:

Case 1: $k^I - |\mathcal{W}_1| \leq r^F$. Then $|\mathcal{W}_2| = k^I - |\mathcal{W}_1|$ and by Equation (3) it holds that $|\mathcal{D}_i| \geq |\mathcal{W}_1| + |\mathcal{W}_2| = k^I$.

Case 2: $k^I - |\mathcal{W}_1| > r^F$. Then $|\mathcal{W}_2| = r^F$ and by Equation (3) it holds that:

$$|\mathcal{D}_i| \geq |\mathcal{W}_1| + r^F. \tag{4}$$

Notice that there are only r^I retired (i.e. not unchanged) encoding vectors in codeword i. Since every accessed encoding vector is either in $\mathcal{W}_1$ or is a retired encoding vector, it holds that:

$$|\mathcal{D}_i| \leq |\mathcal{W}_1| + r^I. \tag{5}$$

By combining Equation (4) and Equation (5), we arrive at the contradiction $r^F \leq r^I$, which occurs because there are not enough retired symbols in the initial codeword i to ensure that the final code has the MDS property. Therefore, case 1 always holds, and $|\mathcal{D}_i| \geq k$. ◀

A.4 Proof of Lemma 13

The notation used in this proof is introduced in Appendix A.1. We show that, even for non-stable convertible codes, that is, when there are more than r^F new symbols, the bounds on the read access set $\mathcal{D}$ from Theorem 12 still hold.

Case 1: $r^I \geq r^F$. Let $i \in [\lambda]$ be an arbitrary initial codeword. We lower bound the size of $\mathcal{D}_i$ by invoking the MDS property on a subset $\mathcal{V} \subseteq \mathcal{S}^F$ of size $|\mathcal{V}| = \lambda k^I$ that minimizes the size of the intersection $|\mathcal{V} \cap \mathcal{U}_i|$. There are exactly r^F encoding vectors in $\mathcal{S}^F \setminus \mathcal{V}$, so the minimum size of the intersection $|\mathcal{V} \cap \mathcal{U}_i|$ is $\max\{|\mathcal{U}_i| - r^F, 0\}$. Clearly, the subset $\text{proj}_{\mathcal{K}_i}(\mathcal{V})$ has rank k^I due to the MDS property. Therefore, it holds that $|\mathcal{D}_i| + \max\{|\mathcal{U}_i| - r^F, 0\} \geq k^I$. By reordering, the following is obtained:

$$|\mathcal{D}_i| \geq k^I - \max\{|\mathcal{U}_i| - r^F, 0\} \geq \min\{r^F, k^I\},$$

which means that the bound on $\mathcal{D}_i$ established in Lemma 10 continues to hold for non-stable codes.

Case 2: $r^I < r^F$. Let $i \in [\lambda]$ be an arbitrary initial codeword, let $\mathcal{W}_1 = (\{\tilde{\mathbf{g}}^I_{i,j} \mid j \in \mathcal{D}_i\} \cap \mathcal{U}_i)$ be the unchanged encoding vectors that are accessed during conversion, and let $\mathcal{W}_2 = \mathcal{U}_i \setminus \mathcal{W}_1$ be the unchanged encoding vectors that are *not* accessed during conversion. Consider the subset $\mathcal{V} \subseteq \mathcal{S}^F$ of $k^F = \lambda k^I$ encoding vectors from the final codeword such that $\mathcal{W}_1 \subseteq \mathcal{V}$ and the size of the intersection $\mathcal{W}_3 = (S \cap \mathcal{W}_2)$ is minimized. Since $\mathcal{V}$ may exclude at most r^F encoding vectors from the final codeword, it holds that:

$$|\mathcal{W}_3| = \max\{0, |\mathcal{W}_2| - r^F\}. \tag{6}$$

By the MDS property, $\mathcal{V}$ is a linearly independent set of encoding vectors of size k^F, and thus, must contain all the information to recover the contents of every initial codeword, and in particular, initial codeword i. Since all the information in $\mathcal{V}$ about codeword i is in either $\mathcal{W}_3$ or the accessed encoding vectors, it must hold that:

$$|\mathcal{D}_i| + |\mathcal{W}_3| \geq k^I. \tag{7}$$

From Equation (6), there are two cases:

Subcase 2.1: $|\mathcal{W}_2| - r^F \leq 0$. Then $|\mathcal{W}_3| = 0$, and by Equation (7) it holds that $|\mathcal{D}_i| \geq k^I$, which matches the bound of Lemma 11.

Subcase 2.2: $|\mathcal{W}_2| - r^F > 0$. Then $|\mathcal{W}_3| = |\mathcal{W}_2| - r^F$, and by Equation (7) it holds that:

$$|\mathcal{D}_i| + |\mathcal{W}_2| - r^F \geq k^I. \tag{8}$$

The initial codeword i has $k^I + r^I$ symbols. By the principle of inclusion-exclusion we have that:

$$|\mathcal{D}_i| + |\mathcal{U}_i| - |\mathcal{W}_1| \leq k^I + r^I. \tag{9}$$

By using Equation (8), Equation (9) and the fact that $|\mathcal{W}_2| = |\mathcal{U}_i| - |\mathcal{W}_1|$, we conclude that $r^I \geq r^F$, which is a contradiction and means that subcase 2.1 always holds in this case. ◀

A.5 Proof of Lemma 14

Lemma 13 shows that the lower bound on the read access set $\mathcal{D}$ for stable linear MDS convertible codes continues to hold in the non-stable case. Furthermore, this bound is achievable by stable linear MDS convertible codes in the merge regime (as will be shown in Section 4). The number of new blocks written during conversion under stable MDS convertible codes is r^F. On the other hand, the number of new symbols under a non-stable convertible code is strictly greater than r^F. Thus, the overall access cost of a non-stable MDS $(n^I, k^I; n^F, k^F = \lambda k^I)$ convertible code is strictly greater than the access cost of an access-optimal $(n^I, k^I; n^F, k^F = \lambda k^I)$ convertible code. ◀

B Proofs of Section 4

B.1 Proof of Lemma 20

Consider the first r^F columns of $\mathbf{P}^I$, which we denote as $\mathbf{P}^I_{r^F} = \mathbf{P}^I[*; 1, \ldots, r^F]$. Notice that $\mathbf{P}^F$ can be written as the following block matrix:

$$\mathbf{P}^F = \begin{bmatrix} \mathbf{P}^I_{r^F} \\ \mathbf{P}^I_{r^F} \operatorname{diag}(1, \theta^{k^I}, \theta^{2k^I}, \ldots, \theta^{k^I(r^F-1)}) \\ \mathbf{P}^I_{r^F} \operatorname{diag}(1, \theta^{2k^I}, \theta^{2\cdot 2k^I}, \ldots, \theta^{2k^I(r^F-1)}) \\ \vdots \\ \mathbf{P}^I_{r^F} \operatorname{diag}(1, \theta^{(\lambda-1)k^I}, \theta^{2(\lambda-1)k^I}, \ldots, \theta^{(\lambda-1)k^I(r^F-1)}) \end{bmatrix},$$

where $\operatorname{diag}(a_1, a_2, \ldots, a_n)$ is the $n \times n$ diagonal matrix with $a_1, \ldots, a_n$ as the diagonal elements. From this representation, it is clear that $\mathbf{P}^F$ can be constructed from the the first r^F columns of $\mathbf{P}^I$. ◀

B.2 Proof of Lemma 21

Let $\mathbf{R}$ be a $t \times t$ submatrix of $\mathbf{P}^I$ or $\mathbf{P}^F$, determined by the row indices $i_1 < i_2 < \cdots < i_t$ and the column indices $j_1 < j_2 < \cdots < j_t$, and denote entry (i, j) of $\mathbf{R}$ as $\mathbf{R}[i, j]$. The determinant of $\mathbf{R}$ is defined by the Leibniz formula:

$$\det(\mathbf{R}) = \sum_{\sigma \in \operatorname{Perm}(t)} \operatorname{sgn}(\sigma) \prod_{l=1}^{t} \mathbf{R}[l, \sigma(l)] = \sum_{\sigma \in \operatorname{Perm}(t)} \operatorname{sgn}(\sigma) \theta^{E_\sigma} \tag{10}$$

$$\text{where} \qquad E_\sigma = \sum_{l=1}^{t} (i_l - 1)(j_{\sigma(l)} - 1), \tag{11}$$

Perm(t) is the set of all permutations on t elements, and $\text{sgn}(\sigma) \in \{-1, 1\}$ is the sign of permutation σ. Clearly, $\det(\mathbf{R})$ defines a univariate polynomial $f_{\mathbf{R}} \in \mathbb{F}_p[\theta]$. We will now show that $\deg(f_{\mathbf{R}}) = \sum_{l=1}^{t}(i_l - 1)(j_l - 1)$ by showing that there is a unique permutation $\sigma^* \in \text{Perm}(t)$ for which E_{σ^*} achieves this value, and that this is the maximum over all permutations in Perm(t). This means that $f_{\mathbf{R}}$ has a leading term of degree E_{σ^*}.

To prove this, we show that any permutation $\sigma \in \text{Perm}(t)\backslash\{\sigma^*\}$ can be modified into a permutation σ' such that $E_{\sigma'} > E_{\sigma}$. Specifically, we show that $\sigma^* = \sigma_{\text{id}}$, the identity permutation. Consider $\sigma \in \text{Perm}(t)\backslash\{\sigma_{\text{id}}\}$: let a be the smallest index such that $\sigma(a) \neq a$, let $b = \sigma^{-1}(a)$, and let $c = \sigma(a)$. Let σ' be such that $\sigma'(a) = a$, $\sigma'(b) = c$, and $\sigma'(d) = \sigma(d)$ for $d \in [t]\backslash\{a, b\}$. In other words, σ' is the result of "swapping" the images of a and b in σ. Notice that $a < b$ and $a < c$. Then, we have that:

$$E_{\sigma'} - E_{\sigma} = (i_a - 1)(j_a - 1) + (i_b - 1)(j_c - 1) - (i_a - 1)(j_c - 1) - (i_b - 1)(j_a - 1) \quad (12)$$

$$= (i_b - i_a)(j_c - j_a) > 0 \quad (13)$$

The last inequality comes from the fact that $a < b$ implies $i_a < i_b$ and $a < c$ implies $j_a < j_c$. Therefore, $\deg(f_{\mathbf{R}}) = \max_{\sigma \in \text{Perm}(t)} E_{\sigma} = E_{\sigma_{\text{id}}}$.

Let $E^*(\lambda, k^I, r^I, r^F)$ be the maximum degree of $f_{\mathbf{R}}$ over all submatrices $\mathbf{R}$ of $\mathbf{P}^I$ or $\mathbf{P}^F$. Then, $E^*(\lambda, k^I, r^I, r^F)$ corresponds to the diagonal with the largest elements in $\mathbf{P}^I$ or $\mathbf{P}^F$. In $\mathbf{P}^F$ this is the diagonal of the square submatrix formed by the bottom r^F rows. In $\mathbf{P}^I$ it can be either the diagonal of the square submatrix formed by the bottom r^I rows, or by the right k^I columns. Thus, we have that:

$$E^*(\lambda, k^I, r^I, r^F) = \max\left\{\sum_{i=0}^{r^F-1} i(\lambda k^I - r^F + i), \sum_{i=0}^{r^I-1} i(k^I - r^I + i), \sum_{i=0}^{k^I-1} i(r^I - k^I + i)\right\}$$

$$= (1/6) \cdot \max\left\{\begin{matrix} r^F(r^F - 1)(3\lambda k^I - r^F - 1), \\ r^I(r^I - 1)(3k^I - r^I - 1), \\ k^I(k^I - 1)(3r^I - k^I - 1) \end{matrix}\right\}.$$

Let $D = E^*(\lambda, k^I, r^I, r^F) + 1$. Then, if $\det(\mathbf{R}) = 0$ for some submatrix $\mathbf{R}$, θ is a root of $f_{\mathbf{R}}$, which is a contradiction since θ is a primitive element and the minimal polynomial of θ over $\mathbb{F}_p$ has degree $D > \deg(f_{\mathbf{R}})$ [34]. ◀

C Proofs of Section 5

C.1 Proof of Theorem 25

Consider the construction Hankel$_s$ described in this section, for some $s \in \{\lambda, \ldots, r^I\}$. The Hankel form of T_m and the manner in which $\mathbf{P}^I$ and $\mathbf{P}^F$ are constructed guarantees that the l-th column of $\mathbf{P}^F$ corresponds to the vertical concatenation of columns $l, l+t, \ldots, l+(\lambda-1)t$ of $\mathbf{P}^I$. Thus, $\mathbf{P}^F$ is r^F-column block-constructible from $\mathbf{P}^I$. Furthermore, since $\mathbf{P}^I$ and $\mathbf{P}^F$ are submatrices of T_m, they are superregular. Thus $\mathbf{P}^I$ and $\mathbf{P}^F$ satisfy both of the properties laid out in Theorem 19 and hence the convertible code constructed by Hankel$_s$ is access-optimal. ◀

D Algorithm for constructing superregular Hankel triangular arrays

In this appendix we describe the algorithm from [52] for constructing a superregular Hankel triangular array over any finite field. This is supplied as reference and is not necessary for understanding the constructions described in this paper. We note that the algorithm outlined in [52] takes the field size q as input, and generates T_q as the output. It is easy to see that T_q thus generated can be truncated to generate the triangular array T_m for any $m \leq q$.

Let $\mathbb{F}_q$ be a given base field, and let $m \leq q$ be the size of the output triangular array T_m. The triangular array T_m has Hankel form, as shown in Equation (1). Therefore, it suffices to specify the entries $b_1, b_2, \ldots, b_m$ in the first column of T_m. On input $m \leq q$, the algorithm proceeds as follows:

1. Consider the extension field $\mathbb{F}_{q^2}$ and choose an element $\beta \in \mathbb{F}_{q^2}$ such that $\beta^i \notin \mathbb{F}_q$ for $i \in [q]$ and $\beta^{q+1} \in \mathbb{F}_q$. Let $p(x) = x^2 + \mu\, x + \eta$ be the minimal polynomial of β over $\mathbb{F}_{q^2}$.
2. Let $\sigma_{-1}, \sigma_0, \ldots, \sigma_m \in \mathbb{F}_q$ be such that $\sigma_{-1} = -\eta^{-1}, \sigma_0 = 0$, and $\sigma_i = -\mu\, \sigma_{i-1} - \eta\, \sigma_{i-2}$, for $i \in [m]$.
3. Set $b_i = \sigma_i^{-1}$, for all $i \in [m]$.

The resulting triangular array is superregular, that is, every square submatrix taken from T_m is superregular. Please refer to [52] for a proof of this fact.

Incentive Compatible Active Learning

Federico Echenique
Division of the Humanities and Social Sciences, California Institute of Technology, Pasadena, CA, USA
fede@hss.caltech.edu

Siddharth Prasad
Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA
sprasad2@cs.cmu.edu

Abstract

We consider active learning under incentive compatibility constraints. The main application of our results is to economic experiments, in which a learner seeks to infer the parameters of a subject's preferences: for example their attitudes towards risk, or their beliefs over uncertain events. By cleverly adapting the experimental design, one can save on the time spent by subjects in the laboratory, or maximize the information obtained from each subject in a given laboratory session; but the resulting adaptive design raises complications due to incentive compatibility. A subject in the lab may answer questions strategically, and not truthfully, so as to steer subsequent questions in a profitable direction.

We analyze two standard economic problems: inference of preferences over risk from multiple price lists, and belief elicitation in experiments on choice over uncertainty. In the first setting, we tune a simple and fast learning algorithm to retain certain incentive compatibility properties. In the second setting, we provide an incentive compatible learning algorithm based on scoring rules with query complexity that differs from obvious methods of achieving fast learning rates only by subpolynomial factors. Thus, for these areas of application, incentive compatibility may be achieved without paying a large sample complexity price.

2012 ACM Subject Classification Theory of computation → Models of learning

Keywords and phrases Active Learning, Incentive Compatibility, Preference Elicitation

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.67

Funding *Federico Echenique*: Echenique thanks the NSF for support through the grants SES-1558757 and CNS-1518941.
Siddharth Prasad: Prasad thanks the Caltech SURF Program for support.

1 Introduction

We study active learning under incentive compatibility constraints. Consider a learner: Alice, who seeks to elicit the parameters governing the behavior of a human subject: Bob. The chief application of our paper is to the design of laboratory experiments in economics. In such applications, Alice is an experimenter observing choices made by Bob in her laboratory. The active learning paradigm seeks to save on the number of questions posed by Alice by making the formulation of each question dependent on Bob's answers to previous questions [8, 21]. Now, Bob may misrepresent his answers to some of Alice's questions so as to guide Alice's line of questioning in a direction that he can benefit from.

Our setting differs from standard applications of active learning in computer science, in that data are labeled by a self-interested human agent (in our story, Bob). Computer scientists have thought of active learning as applied to, for example, combinatorial chemistry, or image detection. A learner then makes queries that are always truthfully answered. In economic settings, in contrast, one must recognize the role of incentives.

The existing literature on applications of passive learning to preference elicitation (see for example [12, 30, 11, 17]) does not have to deal with agents' incentives to manipulate the learning mechanism, but active learning does, because an agent who understands the learner's algorithm may answer strategically early on in the experiment so as to influence the questions he faces later in the experiment.

We should emphasize that experimental orthodoxy in economics requires that subjects (such as Bob) know as much as possible about the experimental design. No deception is allowed in economic experiments. In addition, subjects' participation is almost universally incentivized: Bob gets a payoff that depends on his answers to Alice's questions. Our model relates to a long-standing interest among economists for adaptive experimental design, see [23, 32, 16, 29].

Consider a concrete example. Bob has a utility function x^σ over money, so that if he faces a random amount of money X, his expected utility is $\mathbf{E}[X^\sigma]$. In other words, Bob has a utility of the "constant relative risk aversion" (CRRA) form, and Alice wants to learn the value of the parameter σ –Bob's relative risk aversion coefficient.[1] A standard procedure for estimating σ is a *multiple price list*.[2]

In a multiple-price list (MPL), Alice successively asks Bob to choose between a sure payoff of x dollars and a fixed lottery L, for example a lottery that flips a fair coin and pays 0 dollars if the coin turns up Heads, and 1 dollar if it turns up Tails. Alice would first ask Bob to choose between a very small amount x (almost zero) and L. Then Alice would raise x a little and ask Bob to choose again. The procedure is repeated, each time increasing the amount x, until reaching a number equal to, or close to 1. At some value x, Bob would switch from preferring the lottery to preferring the fixed amount of money. Then Alice would solve the equation

$$x^\sigma = (1/2)0^\sigma + (1/2)1^\sigma = (1/2) \tag{1}$$

to find the value of σ. Now, it is important to explain how the experiment is incentivized: When the experiment is over, Alice will actually implement one of the choices made by Bob. Conventional experimental methodology dictates [7] that she chooses one of the questions at random and implements it.

A proponent of active learning will immediately remark that the MPL design asks too many questions. Alice only needs to know the value of x at which Bob is indifferent between x and the lottery L. We can thus imagine an adaptive design, where Alice raises x until Bob switches from L to x, and stops the experiment when that happens. This design will result in strictly fewer questions than the passive (supervised) learning design.

Bob, however, understands that Alice stops raising x when he declares indifference to L. So he will manipulate Alice into offering him values of x *beyond* what he truly views as indifferent to L. Specifically, suppose that Alice raises x continuously (this is a simplifying assumption; see Section 3 for a realistic version of this design), and that if Bob declares indifference at x then the last question is implemented with probability $p(x) \in (0, 1)$. The function p is strictly decreasing since reporting a larger value of x increases the probability that a question for which Bob preferred L will be implemented.

---

[1] The coefficient σ captures Bob's willingness to assume risk. It is a parameter that economic experiments very often seek to measure, even when the experiment is ostensibly about a totally different question. Economic experimentalists want to understand the relation between risk and their general experimental findings, so they include risk elicitation as part of the design.

[2] Multiple price lists are a very common experimental design, first used by [13], and popularized by [28] as a method to estimate σ, as described here.

Bob's payoff from stopping at x is $\pi(x;\sigma) = p(x)x^\sigma + (1-p(x))(1/2)$ because with probability $p(x)$ the last question gets implemented, so he gets the sure amount x, and with the complementary probability one of the other questions is implemented and Bob gets his preference for those questions, namely the lottery L. The expected utility of L is $1/2$.

Then it is clear that Bob would like to stop at an x that is strictly greater than the value at which he would truly be indifferent to L, the value that solves (1). If he stops at the true value of x, he gets for sure something that he values as much as L (either L or the amount x that he values exactly as L). By stopping at a strictly greater x, he has a shot at getting a value of x that he prefers over L.

The situation is, however, far from hopeless. Bob's optimal value of x is strictly increasing in σ.[3] Alice can then undo Bob's strategic choice of x and back out the true value of σ. (Alice's approach is common in applied econometrics, often called the "structural" method.)

In this paper, we prove general possibility results, to illustrate that there are many situations where active learning is consistent with incentive compatibility. In Section 3 we shall present a formal model of multiple price lists, and show that it is possible to learn while satisfying incentive compatibility. In Section 4 we discuss incentive issues in active learning in a more general sense. We present a formal notion of incentive compatible active learning in a general preference elicitation environment, and provide characterizations of the complexity of incentive compatible learning in certain "nice" environments.

A recent and growing body of work studies the problem of inferring models of economic choice from a learning theoretic perspective [30, 12, 35, 9, 11, 17, 10]. The learnability of preference relations has also received very recent attention [11, 17]. Our investigation takes a different angle: in attempting to model an experimental situation where subjects are asked to make choices in an interactive manner, via, e.g., a computer program, or in person, we allow the analyst complete control over the learning data. In the active learning literature, this framework is known as the *membership queries* model. There is also an ongoing line of work that considers learning problems when the data provider is strategic [22, 1, 31, 18]. Finally, the recent work of [27] studies a model where an agent may (at a cost) manipulate the input to a classification algorithm.

The membership query model closely captures an adaptive economic experiment, while in this context the more traditional learning/active learning models (e.g. PAC learning, stream-based active learning) seem to place unnecessary restrictions on how the analyst learns. This notion is briefly discussed in [17], where classical learning theoretic approaches appear to give much weaker complexity guarantees than the membership queries model in learning time-dependent discounted utility preferences. *For the remainder of this paper, whenever we use the phrase "active learning", we refer to the membership query setting – all other forms of learning can be viewed as a special case of membership queries.*

The other important component in modelling an economic experiment is a payment to the agent after the experiment has concluded. Experiments in economics are always *incentivized*, meaning that there are actual material consequences to subjects' decisions in the lab. Subjects are paid for their decisions in the experiment. We incorporate this incentive payment into the execution of the algorithm by which the analyst chooses questions – the analyst implements the outcome chosen by the agent in the final round of the interaction. Thus, rather than treating the payment scheme as a separate problem, we use it to demand a certain level of

[3] If $\pi(x;\sigma) = p(x)x^\sigma + (1-p(x))(1/2)$ and we assume that p is smooth, then $\partial^2\pi(x;\sigma)/\partial\sigma\partial x = p'(x)x^\sigma \ln(x) + (\sigma+1)p(x)x^{\sigma-1} > 0$ as $x \in (0,1]$ and p is decreasing. So π is strictly supermodular. Hence the optimal x is increasing in σ.

robustness from our learning algorithms. As we demonstrate, this precludes the analyst from running naive learning algorithms that, despite achieving good query complexities, allow the agent to strategically and dishonestly answer questions to get offered higher payoff outcomes.

Finally, the framework we introduce engenders the following natural question: is there a combinatorial measure of complexity, akin to VC dimension for PAC learning concept classes, that precisely captures the complexity of incentive compatible learning in preference environments? Our results examine certain sufficient conditions for incentive compatible learning, a potential first step towards better understanding this new and interesting learning model.

Summary of results

We begin by discussing incentive issues in a very common experimental paradigm, that of convex budgets. We present an example to the effect that incentive problem are present and can be critical. Then we turn to the *Multiple Price Lists* (MPL), another very common experimental design used to infer agents' attitudes towards risk. In MPL experiments, an agent is asked to choose between receiving various deterministic monetary amounts and participating in a lottery. The goal of the analyst is to elicit the agent's *certainty equivalent*, i.e. the deterministic quantity at which the agent values the lottery (in our previous discussion, the certainty equivalent is the quantity that solves Equation (1)). We analyze a simple sequential search mechanism that is used in practice – start from the lowest possible deterministic amount and keep increasing the offer until the agent prefers it to the lottery. The analyst pays the agent by implementing the agent's decision on a randomly selected question that was asked. We show that while this mechanism is not incentive compatible, under relatively benign assumptions it satisfies a one-to-one condition where the analyst can accurately infer the agent's true certainty equivalent after learning the agent's reported certainty equivalent. We then show how a modified payment scheme that only depends on the final decision of the agent allows the analyst to do a binary search and retain incentive compatibility, giving a mechanism for learning the certainty equivalent of a strategic agent to within an error of ε using $O(\log 1/\varepsilon)$ questions.

We then turn to an abstract model of learning preference parameters/types. The idea is, as in the MPL, to induce incentive compatibility by incentivizing the payment from the last question asked of the agent. To this end, we coin a formal notion of incentive compatible (IC) learnability. A learning algorithm is simply an adaptive procedure that at each step asks the agent to choose between two outcomes. Informally, the *IC learning complexity* of an algorithm is the number of rounds required to both

1. Accurately learn (with high probability) the agent's type with respect to some specified metric on the type space.
2. Ensure that (with high probability) the payment mechanism of implementing the agent's choice on the final question cannot be strategically manipulated to yield a significant payoff gain over answering questions truthfully.

A simple structural condition allows a strong notion of incentive compatibility to be achieved via a deterministic exhaustive search (truthful reporting is the agent's unique best response), and we give examples of commonly studied economic preference models that fit our condition. We demonstrate that a large class of preference relations over Euclidean space – those exhibiting strict convexity under a condition which we call *hyperplane uniqueness* (detailed in Section 4) – can be learned in an incentive compatible manner.

▶ **Theorem 1** (informal). *Let Θ be a type space such that the preferences induced by each $\theta \in \Theta$ are continuous, strictly convex, and satisfy hyperplane uniqueness. Then, Θ is IC learnable, under a suitably chosen metric.*

However, this strong notion of incentive compatibility comes at a cost – the associated IC learning complexity can be massive (exponential in the preference parameters). In the abstract setting of preferences over outcomes, it is unclear how to obtain a tangible improvement in this complexity (even with randomization), and specifically it would appear that the problem parameters (e.g. the outcome space, the set of possible agent types) require much more structure for any sort of improvement.

We then analyze the specific setting of learning the beliefs of an expected utility agent, where we have the required structure. Here, an agent holds a belief represented by a distribution $\alpha \in \Delta_n$ (there are n uncertain states of the world, and α_i is the probability with which the agent believes state i will occur), and is asked to make choices between vectors of rewards $x \in \mathbb{R}^n$, where the utility an agent of type α enjoys from x is simply $\alpha.x$. We first observe that naive learning algorithms can vastly beat the learning complexity of the general preference framework, but fail to be incentive compatible. Our main result is an incentive compatible learning algorithm for eliciting an agent's beliefs that significantly improves upon the complexity in the general framework, and only differs from the fast naive learning algorithms by subpolynomial factors.

▶ **Theorem 2.** *There is an algorithm for learning the belief of an expected utility agent that when run for*

$$O\left(n^{3/2}\log n \max\left(\log\frac{n}{\varepsilon}, \log\frac{1}{\tau}\right)\right)$$

rounds (with high probability) cannot be manipulated to yield more than a τ increase in payoff, and learns a truthful agent's belief to within total variation distance of ε. [4]

Our algorithm is built upon disagreement based active learning methods that provide learning guarantees, and employs the spherical scoring rule to ensure incentive compatibility properties.

2 Example: Convex budgets

We present a simple example to illustrate how incentive issues can prevent a very popular experimental design from being implementable in an active learning setting.

Consider an experiment on choice under uncertainty, with an adaptive "convex budgets" design. Such designs are ubiquitous in experimental economics: see [5, 19, 2, 25, 4, 6] among (many) others. Convex budgets is very popular as a design because it parallels the most basic model in economic theory, the model of consumer choice.[5]

Bob, a subject in the lab, has expected utility preferences. Specifically, suppose that the experiment involves two possible states of the world, and that Bob chooses among vectors $x = (x_1, x_2) \in \mathbb{R}^2_+$. If Bob chooses the vector (x_1, x_2) and the state of the world turns out to

[4] Typical supervised learning bounds have a logarithmic dependence on the confidence parameter $\frac{1}{\delta}$, and so for the sake of brevity we omit terms depending on δ in our complexity bounds.

[5] Consumer choice is probably the first model a student of economics is ever exposed to. It captures optimal choice from an economic budget sets, defined from linear prices and a maximum expenditure level.

be i, then he is paid x_i. Bob believes that the state of the world i occurs with probability α_i, so his expected utility from choosing x is $\alpha_1 x_1 + \alpha_2 x_2$ (we assume for simplicity that Bob is risk-neutral).

The experiment seeks to learn the subjects' beliefs α with a design that has Bob choosing

$$x \in B(p, I) = \{y \in \mathbb{R}^2_+ : p \cdot y \leq I\},$$

at prices $p \in \mathbb{R}^2_+$ and income $I > 0$. The problem is equivalent to learning the ratio α_1/α_2. It is obviously optimal for Bob to choose $x = (1/p_1, 0)$ if $\alpha_1/p_1 > \alpha/p_2$ and $x = (0, 1/p_2)$ if $\alpha_1/p_1 < \alpha/p_2$.

The experimental design presents the subject with a sequence of prices p and incomes I, and asks him to choose from $B(p, I)$. Usually only one of the choice problems in the sequence will actually be paid off. It is standard practice in experimental economics to pay out only one of the questions posed to a subject. For the purpose of this example, imagine that the sequence has a length of 2: (p^1, I^1) and (p^2, I^2). Moreover, suppose (again for simplicity) that incomes and prices are such that $I^t = p^t \cdot (1/2, 1/2) = 1$, for $t = 1, 2$.

Fix the first price at $p^1 = (1, 1)$. If Alice, the experimenter, observes a choice of $(1/p_1, 0)$ she should conclude that $\alpha_1/\alpha_2 > p^1_1/p^1_2$. And given such an inference, it would not make sense to set the second set of prices so that $p^2_1/p^2_2 < p^1_1/p^1_2$. Alice, following an active learning paradigm of adaptive experimental design, should adjust p_1/p_2 upwards. So let us assume that she decides to adjust the ratio p^1_1/p^1_2 by a factor of 2 *in the direction in which there is something to learn:* If the choice from $B(p^1, I^1)$ is $(1/p^1_1, 0)$, Alice will set $p^2_1/p^2_2 = 2(p^1_1/p^1_2)$. If the choice is $(0, 1/p^1_2)$, she will set $p^2_1/p^2_2 = (1/2)(p^1_1/p^1_2)$.

Now consider the problem facing our subject, Bob. Suppose that Bob's beliefs are such that $\alpha_1 < \alpha_2$, and, to make our calculations simpler, that $\alpha_1/\alpha_2 \leq 1/2$. If he chooses "truthfully" according to his beliefs, he would choose $x = (0, 1/p^1_1) = (0, 1)$ from $B(p^1, I^1)$ and thus face prices $p^2 = (2/3, 4/3)$. This means that the relative price of payoffs in state 2 increase, the state that Bob values the most because he thinks it is the most likely to occur. If instead Bob "manipulates" the experiment by choosing $x = (1/p_1, 0)$, he will face prices $p^2 = (4/3, 2/3)$. It is obvious that Bob is better off in the second choice problem from facing the second budget because he will be able to afford a much larger payoff in state 2. If Alice only incentivizes (pays out) the choice from $B(p_2, I_2)$, then Bob is always better off by misrepresenting his choice from the first budget.

If, instead, Alice incentivizes the experiment by implementing one of the choices made by Bob at random (a common practice in economic experiments, see [7] for a formal justification), then the utility from truthtelling is $(1/2)\alpha_2(1+3/4) = \alpha_2(7/8)$. The utility from manipulation is $(1/2)(\alpha_1 + \alpha_2(3/2))$. As long as $\alpha_1/\alpha_2 \in (1/4, 1/2)$, the manipulation yields a higher utility than truth telling.

The convex budgets example illustrates the perils of active learning as a guide to adaptive experimental design, when human subjects understand how the experiment unfolds conditional on how they make choices. The main result of our paper (see Section 4.2) considers belief elicitation, but proposes an active learning algorithm that is based on pairwise comparisons, not choices from convex budgets.

3 Multiple Price Lists

We begin by formally considering the application in the introduction: the use of *Multiple Price Lists* (MPL) to elicit an agent's preferences over risk. MPL was first proposed by [13], and popularized by [28], who used it to estimate risk attitudes along the lines of the discussion in the sequel.

We shall consider a version of MPL where a lottery with monetary outcomes is fixed, and an agent chooses between a sure (deterministic) monetary payment x or the lottery. More specifically, consider a lottery where a coin is flipped and if the outcome is heads, the payoff is $\overline{x}$ dollars, while if the outcome is tails, the payoff is $\underline{x}$ dollars. An analyst wants to assess an agent's willingness to participate in the lottery when presented with various deterministic alternatives. Denote this lottery by L.

At every round of the experiment, the analyst asks the agent to choose between a deterministic payoff of x or participation in the lottery, and aims to learn the agent's *certainty equivalent*: the deterministic amount that yields indifference. Conventionally (for example, see [28, 3]), the experiment is run by presenting the agent with a list of n pairs (x_i, L). The agent makes a choice from each pair, either the sure amount x_i or the lottery L. Then the experimenter draws one of the n questions at random an pays the agent according to the decision he made for that question. (i.e. if he preferred the deterministic amount x, he is paid x, and otherwise gets to participate in the lottery). We now present a formal model of the MPL experimental design and analyze issues of incentive compatibility.

3.1 The model

We consider a lottery L with a low outcome $\underline{x}$ and a high outcome $\overline{x}$, $\underline{x} < \overline{x}$. The lottery can operate in any number of ways, for example, by a coin flip. The analyst chooses a discretization $\underline{x} = x_0 < x_1, \ldots < x_{n-1} < x_n = \overline{x}$ of the interval $I = [\underline{x}, \overline{x}]$ such that the intervals $I_k = (x_k, x_{k+1}]$ all have equal length $\ell = \frac{\overline{x}-\underline{x}}{n}$. This discretization of I represents the deterministic amounts that the analyst will offer to the agent.

An agent's certainty equivalent is the point $x \in (\underline{x}, \overline{x})$ such that he is indifferent between receiving x versus participating in the lottery. Certainty equivalents will be uniquely determined by an agent's utility over money, as long as his utility function is strictly increasing.

For example, if an agent values money according to $u : \mathbb{R} \to \mathbb{R}$, his certainty equivalent (assuming that L is a coin-flip) would be the point $x \in (\underline{x}, \overline{x})$ such that $u(x) = \frac{1}{2}u(\underline{x}) + \frac{1}{2}u(\overline{x})$. In our model, we consider agents whose utility functions belong to a given family $\mathcal{U}$ of functions such that a given certainty equivalent uniquely determines the utility function of the agent, and vice-versa. For example, if $\mathcal{U} = \{x \mapsto x^\sigma : 0 < \sigma < 1\}$, so utilities take the CRRA form we discussed in the introduction, then σ uniquely determines the point x such that $x^\sigma = \frac{1}{2}\underline{x}^\sigma + \frac{1}{2}\overline{x}^\sigma$.

3.2 Sequential Search

We first consider a simple mechanism that aims to find the agent's certainty equivalent by performing a sequential search on $x_1, \ldots, x_n$. On round t of the experiment, the agent chooses between the lottery and a deterministic payoff of x_t. If he chooses the lottery, the experiment continues, and if he chooses x_t or claims indifference, the experiment terminates. If the experiment terminates at round T, the analyst can conclude that the agent reported a certainty equivalent lying in the interval $(x_{T-1}, x_T]$.

The goal of the analyst is to make a payment to the agent at the end of the experiment such that the agent is incentivized to answer questions according to his true certainty equivalent. We analyze a common scheme used in experiments: if the experiment terminates after T rounds, choose t randomly from $\{1, \ldots, T\}$, and pay the agent based on his preference on the tth question: so if $t = T$, the agent receives x_T, otherwise the agent receives a payment that is the outcome of the lottery. However, as discussed in the introduction, this scheme

is not incentive compatible. Indeed, if x_T is the agent's true certainty equivalent, he has a profitable deviation to push the experiment to terminate at x_{T+1}. The agent is indifferent between receiving x_T and participating in the lottery, so by declaring a certainty equivalent that is higher than x_T he may possibly win an amount larger than x_T, and which he values strictly more than the lottery.

We now show that under some simplifying assumptions, this kind of payment scheme can at least be implemented in a manner such that the agent's true certainty equivalent can be accurately inferred based on his report. Let $\mathcal{U}$ be a family of utility functions such that each $u \in \mathcal{U}$ satisfies an inverse Lipschitz condition with constant K_u: for all x, x^*, $|u(x) - u(x^*)| > K_u|x - x^*|$. Let $K = \sup_{u\in\mathcal{U}} K_u$. Finally, let $M = \sup_{u\in\mathcal{U}}(u(\overline{x}) - u(\underline{x}))$.

For $t \in \{1, \dots, n\}$ let p_t denote the probability that the agent is paid the deterministic amount x_t if the experiment stops on round t (so the agent participates in the lottery with probability $1 - p_t$). An agent with true certainty equivalent x and corresponding utility function u has an expected payoff of

$$\text{Payoff}(x, x_t) = p_t u(x_t) + (1 - p_t)u(x)$$

for reporting a certainty equivalent in $(x_{t-1}, x_t]$.

Let $r : (\underline{x}, \overline{x}) \to \{x_1, \dots, x_n\}$ be the best response of an agent with certainty equivalent x:

$$r(x) = \text{argmax}_{x_t} \text{Payoff}(x, x_t).$$

We refer to r as the report function.

We now show that when $p_1, \dots, p_n$ satisfy $p_{t+1} < p_t$[6] and $p_{t+1} < \frac{K\ell}{2M} p_t$, the analyst can recover the agent's true certainty equivalent up to some low error via the sequential search mechanism.

We should emphasize that the agent will not be truthful, in the sense of reporting their true certainty equivalent. However, we are still able to back out the true certainty equivalent from understanding the agents strategic incentives.

We proceed in steps. First, we characterize the best responses for agents with certainty equivalents belonging to $\{x_1, \dots, x_{n-1}\}$. We show that the report function is one to one. This implies that $r(x_t) = x_{t+1}$, since $r(x_t) > x_t$, for each $t = 1, \dots, n-1$.

▶ **Proposition 3.** *For $x_t \in \{x_1, \dots, x_{n-1}\}$, $r(x_t) = x_{t+1}$.*

Proof. Note that since $r(x_{n-1}) = x_n$, it suffices to show that r is injective on $\{x_1, \dots, x_{n-1}\}$.

Suppose $r(x_{t_1}) = r(x_{t_2})$, and without loss of generality let $t_1 \geq t_2$. Let $r(x_{t_1}) = r(x_{t_2}) = x_t$. Since any agent is incentivized to report higher than their true certainty equivalent, $t > t_1, t_2$, so in particular $t \geq t_1 + 1$. Let u_1 denote the utility function of the agent with true type x_{t_1}, u_2 that of the agent with true type x_{t_2}.

For any $s \neq t$ we have (since r gives the best response):

$$p_t u_1(x_t) - p_t u_1(x_{t_1}) > p_s u_1(x_s) - p_s u_1(x_{t_1}),$$
$$p_t u_2(x_t) - p_t u_2(x_{t_2}) > p_s u_2(x_s) - p_s u_2(x_{t_2}).$$

Adding the two inequalities and rearranging gives

$$\frac{p_t}{p_s} > \frac{u_1(x_s) - u_1(x_{t_1}) + u_2(x_s) - u_2(x_{t_2})}{u_1(x_t) - u_1(x_{t_1}) + u_2(x_t) - u_2(x_{t_2})}. \tag{2}$$

[6] This is true for the standard uniform randomization scheme, as $p_t = 1/t$. More generally, without this condition the agent will have incentives to report high certainty equivalents as this is not penalized by lower probabilities of winning the certain amount.

At $s = t_1$ (we know $t_1 \neq t$, since $r(x_{t_1}) > x_{t_1}$), Equation 2 simplifies to

$$\frac{p_t}{p_{t_1}} > \frac{u_2(x_{t_1}) - u_2(x_{t_2})}{u_1(x_t) - u_1(x_{t_1}) + u_2(x_t) - u_2(x_{t_2})} > \frac{u_2(x_{t_1}) - u_2(x_{t_2})}{2M},$$

so

$$u_2(x_{t_1}) - u_2(x_{t_2}) < 2M\frac{p_t}{p_{t_1}} \leq 2M\frac{p_{t_1+1}}{p_{t_1}} < K\ell.$$

The inverse Lipschitz condition on u_2 then implies that $|x_{t_1} - x_{t_2}| < \ell$, which cannot happen unless $t_1 = t_2$. ◀

Thus, if an agent's true certainty equivalent happens to coincide with one of the points of the discretization, the agent will answer questions as if his certainty equivalent is the next point in the discretization.

For the next step, we need an additional Lipschitz type condition on utility functions. Suppose there are constants C_1 and C_2 such that for any $x, x^* \in (\underline{x}, \overline{x})$, with u, u^* the corresponding utility functions, and for any $x, x'' \in (\underline{x}, \overline{x})$,

$$|u(x') - u^*(x'')| \leq C_1|x - x^*| + C_2|x' - x''|.$$

Moreover, let

$$\lambda = \inf_{x \in (\underline{x}, \overline{x})} \min_{s,t} |\text{Payoff}(x, x_s) - \text{Payoff}(x, x_t)|,$$

be the smallest possible deviation in payoff obtained by changing one's report.

We also require the assumption that if $x^* \geq x$, then $u^*(x') \geq u(x')$ for any x', where u^* and u are the utility functions corresponding to certainty equivalents x^* and x, respectively. This is an intuitive condition stating that agents with a higher certainty equivalent value money more than agents with a lower certainty equivalent (note that the CRRA utilities discussed previously satisfy this property). This in particular implies that $u^*(x^*) \geq u^*(x) \geq u(x)$, so $\text{Payoff}(x^*, x_k) \geq \text{Payoff}(x, x_k)$ for any x_k in the discretization. We will use this in the proof of the following proposition, which establishes that r satisfies a certain weak monotonicity property.

▶ **Proposition 4.** *Let $x_{t-1} < x < x^* \leq x_t$ with $x^* - x < \frac{\lambda}{2(2C_1+C_2)}$, and suppose $r(x^*) \leq x_{t+1}$. Then, $r(x) \leq x_{t+1}$.*

Proof. Let u, u^* be the utility functions corresponding to certainty equivalents x and x^*, respectively. We first bound the increase in payoff an agent of type x^* experiences over an agent of type x for making the same report. For any x_k, we have

$$\begin{aligned}
\text{Payoff}(x^*, x_k) - \text{Payoff}(x, x_k) &= p_k(u^*(x_k) - u(x_k)) - p_k(u^*(x^*) - u(x)) + (u^*(x^*) - u(x)) \\
&< p_k(u^*(x_k) - u(x_k)) + (u^*(x^*) - u(x)) \\
&< (u^*(x_k) - u(x_k)) + (u^*(x^*) - u(x)) \\
&\leq C_1\ell + (C_1 + C_2)\ell \\
&\leq \frac{\lambda}{2}
\end{aligned}$$

As $r(x^*) \leq x_{t+1}$, either $r(x^*) = x_{t+1}$ or $r(x^*) = x_t$. Suppose $r(x^*) = x_{t+1}$. We show that an agent of type x cannot increase his payoff by reporting above x_{t+1}. Let $s > t + 1$.

Plugging x_{t+1} into the above bound gives

$$\text{Payoff}(x^*, x_{t+1}) \leq \frac{\lambda}{2} + \text{Payoff}(x, x_{t+1}),$$

and the definition of λ gives that

$$\text{Payoff}(x^*, x_{t+1}) \geq \text{Payoff}(x^*, x_s) + \lambda.$$

Combining the two inequalities yields

$$\begin{aligned}\text{Payoff}(x, x_{t+1}) \geq \text{Payoff}(x^*, x_{t+1}) - \frac{\lambda}{2} &> \text{Payoff}(x^*, x_s) + \frac{\lambda}{2} \\ &> \text{Payoff}(x^*, x_s) \\ &> \text{Payoff}(x, x_s),\end{aligned}$$

so $r(x) \leq x_{t+1}$.

In the case that $r(x^*) = x_t$, we similarly get $r(x) \leq x_t$. ◀

We can then repeatedly apply this proposition starting with $r(x_t) = x_{t+1}$ to conclude that for any $x_{t-1} < x \leq x_t$, we have $r(x) \leq x_{t+1}$.

Putting things together, we get:

▶ **Theorem 5.** *If $r(x) = x_{t+1}$, then $x_{t-1} < x < x_{t+1}$.*

Thus, to learn the agent's true certainty equivalent to within ε-error, the analyst chooses a discretization with $\frac{\overline{x}-\underline{x}}{n} \leq \frac{\varepsilon}{2}$, and runs a sequential search over the discretization. The number of questions the analyst asks is $O(\frac{1}{\varepsilon})$.

Of course, to lower the number of questions asked, the analyst could instead perform a binary search. It is easy to see that, like in the sequential search mechanism, simply implementing a uniformly random question is not incentive compatible. For example, consider a discretization with deterministic amounts $x_1, \ldots, x_7$, and consider an agent with true certainty equivalent at x_3. For simplicity, we assume that if when presented with (x_i, L) the agent is indifferent between x_i and L, he chooses x_i. If the agent answers truthfully, the pairs offered by a binary search would be (x_4, L), (x_2, L), and (x_3, L), and his choices would have been x_4, L, and x_3, respectively. The agent's expected payoff is $(1/3)u(x_4) + (1/3)u(L) + (1/3)u(x_3) = (1/3)u(x_4) + (2/3)u(L)$. Suppose instead the agent answers as if his true certainty equivalent is x_5. Then, the pairs he gets offered would be (x_4, L), (x_6, L), and (x_5, L), and his choices would have been L, x_6, and x_5, respectively. His expected payoff is then $(1/3)u(L) + (1/3)u(x_6) + (1/3)u(x_5)$, which is clearly a profitable deviation.

It is unclear if this scheme can be directly modified to satisfy incentive compatibility properties, but since the payments in the sequential search mechanism only depended on the last question asked, we can use the same payment scheme here so that Theorem 5 holds. So now the analyst can learn the agent's certainty equivalent to within an error of ε with $O(\log 1/\varepsilon)$ questions.

4 General Preference Elicitation

Our discussion so far has focused on a specific, albeit ubiquitous, preference elicitation environment. In the rest of the paper we introduce a general model of incentive compatible active learning. We introduce the idea of incentive compatible query complexity: the sample size that guarantees some learning objective while maintaining incentive compatibility.

The main application of our tools will be to expected utility theory. We shall introduce a learning algorithm that is incentive compatible for learning the beliefs of an agent that has expected utility preferences.

We focus on learning an agent's preferences. The agent will be modeled as having a utility function parameterized by some type, which generates the agent's choices, that the learner wishes to infer. To this end, Θ is a type space equipped with a metric $d : \Theta \times \Theta \to \mathbb{R}_{\geq 0}$ that is bounded with respect to d. $\mathcal{O}$ is the space of possible outcomes. An agent of type $\theta \in \Theta$ has utility $u(\theta, o)$ if the outcome is $o \in \mathcal{O}$. θ induces a preference relation $\succsim$ over $\mathcal{O}$ defined by $o \succsim o' \iff u(\theta, o) \geq u(\theta, o')$.

An analyst aims to learn the agent's type by asking him to make a sequence of choices between pairs of outcomes.[7] The agent makes choices among the pairs presented to him.

The agent's choices can be thought of as the result of a strategy. Formally, a strategy σ is a mapping

$$\sigma : \bigcup_t \{((o_1, o_1'), \mathbf{1}_{o_1 \succsim o_1'}), \ldots, ((o_t, o_t'), \mathbf{1}_{o_t \succsim o_t'}), (o_{t+1}, o_{t+1'})\} \to \Delta\{0, 1\}$$

that dictates a (potentially randomized) response for every possible history of the interaction up to any given time. Let Σ denote the collection of all possible consistent strategies (a strategy is consistent if its outputs up to any given time are consistent with some preference relation in the type space).

For any strategy σ, let $\hat{\sigma}$ denote an oracle with memory that responds to queries of the form "is o preferred to o'?" according to σ given the history of previous queries made so far. Let $\hat{\Sigma} = \{\hat{\sigma} : \sigma \in \Sigma\}$ denote the collection of oracles corresponding to all possible strategies. For a type $\theta \in \Theta$, let $\hat{\theta} \in \hat{\Sigma}$ denote the oracle that responds truthfully according to θ (i.e. on query (o, o') it returns $\mathbf{1}_{u(\theta,o) \geq u(\theta,o')}$).

We imagine the oracle playing the role of the agent: in an interaction with the analyst, an agent of true type θ chooses to act as an oracle for some strategy σ (departing from standard terminology, we allow the oracle to have randomized responses).

The analyst implements a learning mechanism, which consists of the following steps:

1. Run a (potentially randomized) learning algorithm $\mathcal{A} : \hat{\Sigma} \to \Theta$ that has access to oracle $\hat{\sigma}$ and can make queries to $\hat{\sigma}$ of the form (o, o') for $o, o' \in \mathcal{O}$.
2. Arrive at a hypothesis $\theta^h \sim \mathcal{A}(\hat{\sigma})$ for the agent's type.
3. Implement the agent's response on the last query.[8]

We now establish the notion of learnability that we work with. This definition is not concerned with issues of incentive compatibility: it is simply a refinement of the standard notion of a learning algorithm that stipulates that we learn a truthfully reported hypothesis accurately. Since in our setting the analyst has full control over the data he learns from, our requirements on the error of the algorithm are with respect to the metric d on the space of types Θ.

[7] One can imagine many other protocols for learning. We constrain ourselves to protocols that are based on a sequence of pairwise comparisons. Such protocols are common in practice, and are the obvious empirical counterpart to the decision theory literature in economics and statistics. This stands in contrast with the literature on scoring rules, which allows for richer message spaces.

[8] Within adaptive experimental design, the idea of making a last choice on behalf of the agent is due to Ian Krajbich.

▶ **Definition 6.** *$\mathcal{A} : \hat{\Sigma} \to \Theta$ is an (ε, δ)-learning algorithm if for all $\theta \in \Theta$,*

$$\Pr_{\theta^h \sim \mathcal{A}(\hat{\theta})} [d(\theta, \theta^h) \leq \varepsilon] \geq 1 - \delta.$$

The number of queries made by $\mathcal{A}$ to the oracle $\hat{\theta}$ is the query complexity *of $\mathcal{A}$, denoted by $q(\varepsilon, \delta)$.*

Next, we define what it means for a learning algorithm to be incentive compatible. Intuitively, we require that if the learning algorithm is terminated on round T, and the analyst implements the agent's preferred outcome on the Tth query (o_T, o'_T), then the agent (with high probability) cannot gain a non-negligible advantage over truthfully reporting by attempting to answer questions strategically. Let $\mathcal{A}_T(\hat{\sigma}) = (o_T, o'_T) \in \Delta(\mathcal{O} \times \mathcal{O})$ denote the Tth query to $\hat{\theta}$ made by an execution of $\mathcal{A}(\hat{\sigma})$.

▶ **Definition 7.** *$\mathcal{A} : \hat{\Sigma} \to \Theta$ is (τ, ν)-incentive compatible if there exists a $T(\tau, \nu) \in \mathbb{N}$ such that for all $T \geq T(\tau, \nu)$, the following holds for any type θ and strategy σ:*

$$\Pr_{\substack{(o_T, p_T) \sim \mathcal{A}_T(\hat{\theta}) \\ (o'_T, p'_T) \sim \mathcal{A}_T(\hat{\sigma})}} [u(\theta, q_T) \geq u(\theta, q'_T) - \tau] \geq 1 - \nu,$$

where q_T (q'_T) is the preferred outcome between o_T and p_T (o'_T and p'_T) according to oracle $\hat{\theta}$ ($\hat{\sigma}$). The quantity $T(\tau, \nu)$ is the IC complexity *of $\mathcal{A}$.*[9]

Our goal is to design mechanisms that learn the agent's true type in an incentive compatible manner.

▶ **Definition 8.** *$\mathcal{A} : \hat{\Sigma} \to \Theta$ is an $(\varepsilon, \delta, \tau, \nu)$-IC learning algorithm if it is an (ε, δ)-learning algorithm that is (τ, ν)-IC. We refer to the quantity $\max(q(\varepsilon, \delta), T(\tau, \nu))$ as the* IC learning complexity *of $\mathcal{A}$.*

4.1 An incentive compatible exhaustive search

We first give a very simple method of achieving incentive compatible learning in the general framework introduced in Section 4. The method proceeds by exhaustively searching over the type space, and requires a simple structural assumption. The assumption connects agents' payoffs to the distance metric used by the learner to assess learning accuracy. In a sense, this lines up the agent's incentives with the learner's objective, and makes it easy to obtain a satisfactory algorithm.

Suppose there exists a one-to-one assignment of outcomes to types $s : \Theta \to \mathcal{O}$ such that

$$u(\theta, s(\theta')) > u(\theta, s(\theta'')) \iff d(\theta, \theta') < d(\theta, \theta''),$$

so in particular $\theta = \operatorname{argmax}_{\theta'} u(\theta, s(\theta'))$. In the literature on scoring rules, s is called *effective* with respect to d [24].

The following is an incentive compatible learning algorithm. Recall that an ε-cover of a subset K of a metric space (M, d) is a set of points C such that for every $x \in K$, there is an $x^* \in C$ such that $d(x, x^*) \leq \varepsilon$.

1. Initialize an ε-cover $\{\theta_1, \ldots\}$ of Θ with respect to d.
2. Initialize $\theta^h \leftarrow \theta_1$.

[9] In this definition, (o_T, p_T) and (o'_T, p'_T) are drawn from independent executions of $\mathcal{A}$.

3. For $t = 1$ to T:
   a. Query $(s(\theta_{t+1}), s(\theta^h))$.
   b. If $s(\theta_{t+1})$ is preferred, $\theta^h \leftarrow \theta_{t+1}$.
4. Output θ^h.
5. Pay the agent $s(\theta^h)$.

By definition of the function s, allowing the algorithm to exhaustively search over all points of the cover will yield a θ^h that is the most preferred point in the cover and is also the closest point in the cover to the report. So reporting θ yields $d(\theta, \theta^h) \leq \varepsilon$. Moreover, this (deterministic) algorithm satisfies $(0, 0)$-incentive compatibility for any runtime T, since lying at any round would simply reduce the payoff of stopping at any round. The learning complexity is the covering number $N_\varepsilon(\Theta)$ of the type space (which is finite as Θ is bounded).

We now present some natural preference environments in which such an assignment function can be constructed. In the following discussion, the outcome space is $\mathcal{O} = \mathbb{R}^n$, and Θ is assumed to be bounded so that the search above terminates.

- *Euclidean preferences.* Each agent has an "ideal point" $\theta \in \mathbb{R}^n$, and $u(\theta, x) \geq u(\theta, y)$ iff $||x - \theta|| \leq ||y - \theta||$. Let $s : \mathbb{R}^n \to \mathbb{R}^n$ be the identity.
- *Linear preferences.* The type of an agent is a vector $\theta \in \mathbb{R}^n$ and $u(\theta, x) \geq u(\theta, y)$ iff $\theta.x \geq \theta.y$ (in order for preferences to be distinguishable we assume that no two $\theta, \theta' \in \Theta$ are scalar multiples of one another, and so for simplicity we normalize so that all types have the same length). The indifference sets of an agent of type θ are the hyperplanes $\{x : \theta.x = k\}$, for $k \in \mathbb{R}$. For each θ, there is a unique indifference set that is tangent to the unit $(n-1)$-sphere S^{n-1}. Let $s(\theta)$ be that tangent point.

Euclidean and linear preferences are characterized by natural axioms for preference relations [14].

More generally, suppose the preferences of each agent are continuous and strictly convex, which we define as the upper contour sets $C(x) = \{y : y \succsim x\}$ being closed, convex, and any supporting hyperplane of $C(x)$ being unique, for all x. For a type θ, real number k, and outcome $x \in \mathbb{R}^n$ such that $u(\theta, x) = k$, let $H_k^\theta(x)$ denote the supporting hyperplane of the upper contour set $\{y : u(\theta, y) \geq k\}$ at x.

Suppose that the following uniqueness requirement holds: for every pair of types $\theta \neq \theta'$, real number k, and outcome $x \in \mathbb{R}^n$ such that $u(\theta, x) = k$, if k' is such that $u(\theta', x) = k'$, it holds that $H_k^\theta(x) \neq H_{k'}^{\theta'}(x)$. We call this property *hyperplane uniqueness.*[10] Then, the argument for IC learnability in the case of linear preferences can be adapted to this setting as well. Though the assignment function s we construct may not necessarily be effective, we show that exhaustively searching over a sufficiently fine cover is nevertheless incentive compatible.

▶ **Theorem 1.** *Let Θ be a type space such that the preferences induced by each $\theta \in \Theta$ are continuous, strictly convex, and satisfy hyperplane uniqueness. Then, there exists a metric on Θ with respect to which Θ is $(\varepsilon, 0, 0, 0)$-IC learnable.*

Proof. For each θ, let $s(\theta)$ be the unique maximizer of $u(\theta, x)$ over the unit $(n-1)$-sphere S^{n-1}; uniqueness follows from the strict convexity of preferences. Note that S^{n-1} and $C = \{y : u(\theta, y) \geq k\}$, with $k = u(\theta, s(\theta))$, are on differing sides of the supporting hyperplane of C at $s(\theta)$. Hyperplane uniqueness ensures that s is one-to-one. Let d be the metric where $d(\theta, \theta')$ is the Euclidean distance between $s(\theta)$ and $s(\theta')$.

[10] Hyperplane uniqueness is reminiscent of the single-crossing property in mechanism design.

Let $C^\theta_{k'} = \{y : u(\theta, y) > k'\}$ with $k' < k$ and let $N^\theta_{k'} = C_{k'} \cap S^{n-1}$. Clearly the diameter of $N^\theta_{k'}$ converges to 0 as $k' \uparrow k$. Therefore, for each θ and $\varepsilon > 0$, there is an open neighborhood $N^\theta_\varepsilon \subset B_\varepsilon(s(\theta))$ of $s(\theta)$ such that $u(\theta, x) > u(\theta, y)$ for all $x \in N^\theta_\varepsilon$ and all $y \in S^{n-1} \setminus N^\theta_\varepsilon$ (where N^θ_ε is of the form $N^\theta_{k'}$, for k' sufficiently close to k). [11]

Now, fix the learning parameter ε, and let $\eta > 0$ be sufficiently small such that if K is an η-cover of S^{n-1}, $K \cap N^\theta_\varepsilon \neq \emptyset$ for all θ. Then, any $x \in K \cap N^\theta_\varepsilon$ satisfies $u(\theta, x) > u(\theta, y)$, for all $y \in S^{n-1} \setminus B_\varepsilon(s(\theta))$. Thus, the most preferred point of an agent of type θ is contained in $K \cap N^\theta_\varepsilon \subset B_\varepsilon(s(\theta))$, and so exhaustively searching over this η-cover is an ε-learning algorithm with respect to d that is incentive compatible. ◀

It is an interesting question to see what structural conditions one can impose on the type space, the outcome space, etc. to write down better learning mechanisms. For example, one might hope to achieve a learning complexity that is logarithmic in the size of the cover $N_\varepsilon(\Theta)$. As we will see in the case of expected utility, naive learning algorithms achieve this sample complexity, but fail to be incentive compatible. More generally one can ask if there is a combinatorial complexity measure (such as VC dimension in the case of PAC learning) that characterizes the complexity of incentive compatible learning.

4.2 The expected utility model of choice under uncertainty

We now turn to the case of belief elicitation for an expected utility agent. Belief elicitation has a long history in experimental economics, and in the theoretical literature on scoring rules (e.g. [15]; see [20] for a survey). A major difference with the theory of scoring rules is that we shall take as given a protocol that is based on pairwise comparisons among uncertain prospects.[12] The case of passive learning was studied in [11].

There are n states of the world, indexed by $i = 1, \ldots, n$. An agent has a subjective belief $\alpha \in \Delta_n$, where α_i is the probability the agent assigns to state i occurring. The agent evaluates the payoff of a vector of rewards $x \in \mathbb{R}^n$ by computing expectation according to α. An agent's belief α defines a preference relation $\succsim \subseteq \mathbb{R}^n \times \mathbb{R}^n$, where

$$x \succsim y \iff \alpha.x \geq \alpha.y.$$

An analyst would like to learn α by asking the agent to make several choices between vectors of rewards. The analyst presents the agent with a sequence of pairs (x, y) and if the agent chooses x she infers that $(x - y).\alpha \geq 0$. So the problem is related to that of learning half spaces, but with the added complication of having to respect incentive compatibility. An important assumption is that the analyst is able to simulate the states of the world and observe a state according to the "ground truth" process governing the states (so for example if the states were "rain", "snow", and "shine", the analyst could simply observe the weather on the given day).

Using the notation of the previous section, $\Theta = \Delta_n$, $\mathcal{O} = \mathbb{R}^n$, and $u(\alpha, x) = \mathbf{E}_{i \sim \alpha}[x] = \alpha.x$.

In the context of learning the agent's true belief, the analyst uses total variation distance $||\alpha - \beta||_{TV} = \frac{1}{2} \sum_{i=1}^n |\alpha_i - \beta_i|$ to measure accuracy/error.

[11] The neighborhoods and balls are with respect to the subspace topology on S^{n-1}.

[12] This follows experimental practice, as well as the standard model of choice under uncertainty; starting from von-Neumann and Morgenstern [34] and Savage [33]. In the scoring rule model, subjects are asked to report beliefs rather than carrying out a sequence of binary choices. In any case we shall use scoring rules in our solution, just not by asking subjects to report their beliefs.

4.2.1 Naive algorithms are not incentive compatible

First, to illustrate the restrictions of our definitions, we write down a naive algorithm for eliciting α that achieves a good query complexity, but is not incentive compatible.

Consider a mechanism that tries to elicit each α_i by performing a search (sequential or binary) on each state. That is, for each state i, the algorithm makes queries $(e_i, c_i\vec{1})$, varying c_i over a $\frac{2\varepsilon}{n}$-cover of $[0,1]$ to find the indifference points, which reveal α_i to within an error of $\frac{2\varepsilon}{n}$. So, for example, a binary search uses $O(n \log \frac{n}{\varepsilon})$ questions to arrive at a hypothesis within total variation distance ε from α. Note that a $\frac{2\varepsilon}{n}$-cover of the simplex Δ_n with respect to total variation distance contains $O((n/\varepsilon)^n)$ elements, so performing a state-wise binary search exponentially improves upon a search over the entire cover.

However, incentive compatibility is broken rather easily, since the agent has a great deal of control over what questions the agent asks (in a similar manner to the situation in MPL). Consider the following simple example: suppose the analyst fixes a discretization of $[0,1]$ with sure amounts $x_1, \ldots, x_7$, as in the binary search MPL example from Section 3, and suppose an agent has a true belief α, with $\alpha_n \leq x_6$. If, instead of α, the agent reports an α' with $\alpha'_n \in (x_6, x_7)$, the final question he would get asked would be $(e_n, x_7\vec{1})$. The agent would prefer $x_7\vec{1}$, and thus would get paid off a sure amount of x_7. It is clear that truthfully reporting yields a strictly lower payoff than the misrepresentation. Notice that this situation is even worse than that of MPL, since if the binary search ends on state n, then regardless of the probabilities an agent assigns to states $1, \ldots, n-1$, he will want to answer questions as if he assigns most weight to state n – so there is no hope of backing-out an agent's true belief using this kind of scheme.

A strategic agent can easily outwit minor modifications to this scheme: for example if the analyst does the binary searches in a random order over the states, the agent can adaptively report a belief that assigns most weight to the last state over which the analyst performs a binary search.

4.2.2 A mechanism based on scoring rules.

In this section we present an IC learning algorithm with IC learning complexity

$$O\left(n^{3/2} \log n \max\left(\log \frac{n}{\varepsilon}, \log \frac{1}{\tau}\right)\right).$$

The algorithm is based on ideas from active learning, and specifically leverages convergence bounds on so-called disagreement based methods. Let $||\cdot||$ denote the L^2 norm, let S^{n-1} denote the unit $(n-1)$-sphere, and let $\rho : \mathbb{R}^n \to S^{n-1}$ denote the projection map onto the unit sphere defined by $\rho(\alpha) = \frac{\alpha}{||\alpha||}$.

We now present an incentive compatible learning algorithm that we henceforth refer to as $\mathcal{A}$.

1. Initialize $\mathcal{H}^0 \leftarrow \Delta_n$.
2. For $t = 1$ to T:
   a. Choose v uniformly at random from S^{n-1}. If the hyperplane $\{x : v.x = 0\}$ does not intersect $\mathcal{H}^{t-1}$, resample.
   b. Let β^1, β^2 be any elements of $\mathcal{H}^{t-1}$ such that $\rho(\beta^1) - \rho(\beta^2)$ is a scalar multiple of v.
   c. Query oracle on pair $(x_t = \rho(\beta^1), y_t = \rho(\beta^2))$.
   d. $\mathcal{H}^t \leftarrow \mathcal{H}^{t-1} \cap \{\beta \in \Delta_n : \beta \text{ is consistent with label on } (x_t, y_t)\}$
3. Output any $\beta^h \in \mathcal{H}^T$.
4. Pay the agent off based on preference from (x_T, y_T). If z_T is the preferred vector, simulate states of the world, and pay $(z_T)_i$ if state i occurs.

Before analyzing the algorithm, let us briefly remark that the analyst can always find β^1, β^2 satisfying the required conditions to query the agent. Let normal vector $v \in S^{n-1}$ define a hyperplane $v.x = 0$ that cuts through the projection $\rho(\mathcal{H}) \subset S^{n-1}$ of the current hypothesis set onto the unit sphere. Let w be a point in the interior of $\rho(\mathcal{H})$ such that $v.w = 0$.[13] We can find an open ball $B(w, r)$ (with respect to the subspace topology on S^{n-1} induced by $\mathbb{R}^n$) of radius r centered at w such that $B(w, r) \subset \rho(\mathcal{H})$. Then, take a point $x \in B(w, r)$ in the positive v direction from w and $y \in B(w, r)$ in the negative v direction from w such that $||x - w|| = ||y - w||$. Then, $x - y = v||x - y||$.

Choosing β^1 and β^2 in this manner has no effect on the analysis of the learning rate, but is the main ingredient in achieving incentive compatibility. The learning guarantees we obtain are due to standard bounds on the label complexity of disagreement based active learning.

▶ **Theorem 9.** *$\mathcal{A}$ is a learning algorithm of query complexity $O(n^{3/2} \log n \log \frac{n}{\varepsilon})$ with respect to total variation distance.*

Proof. Suppose $\mathcal{A}$ receives as input an oracle $\hat{\alpha}$. If on a given round we sample a normal vector v and correspondingly query points (x_t, y_t), the truthful agent's/oracle's preference from (x_t, y_t) precisely reveals $\text{sgn}(v.\alpha)$ – this is simply because $x_t - y_t$ and v determine the same hyperplane.

The VC dimension of the expected utility model is linear (Theorem 2 of [11]), and the disagreement coefficient of the class of homogeneous linear separators with respect to the uniform distribution over normal vectors is bounded above by $\pi\sqrt{n}$ (Theorem 1 of [26]). Standard convergence results in active learning (see, e.g., [21]) then imply that with $O(n^{3/2} \log n \log \frac{1}{\eta})$ queries, it holds with high probability that $\text{err}_\alpha(\alpha^h) \leq \eta$ for all α^h in the final hypothesis set, where

$$\text{err}_\alpha(\beta) = \Pr_{v \sim S^{n-1}}[\text{sgn}(v.\alpha) \neq \text{sgn}(v.\beta)] = \frac{\arccos(\rho(\alpha).\rho(\beta))}{\pi}.$$

For $\varepsilon > 0$, let $\eta = \frac{2\varepsilon}{\pi n} < \frac{1}{\pi} \arccos\left(1 - \frac{2\varepsilon^2}{n^2}\right)$, so $\cos(\pi\eta) > 1 - \frac{2\varepsilon^2}{n^2}$.
Running $\mathcal{A}$ for $O(n^{3/2} \log n \log \frac{n}{\varepsilon})$ rounds yields that for any hypothesis $\alpha^h \in \mathcal{H}^T$,

$$\begin{aligned} ||\rho(\alpha) - \rho(\alpha^h)|| &= \sqrt{(\rho(\alpha) - \rho(\alpha^h)).(\rho(\alpha) - \rho(\alpha^h))} \\ &= \sqrt{2 - 2\rho(\alpha).\rho(\alpha^h)} \\ &\leq \sqrt{2 - 2\cos(\pi\eta)}, \end{aligned}$$

which is at most $2\varepsilon/n$ (where the final inequality is with high probability over the execution of $\mathcal{A}$). Thus $||\alpha - \alpha^h|| \leq \frac{2\varepsilon}{n}$, and so $||\alpha - \alpha^h||_{TV} \leq \varepsilon$. [14] ◀

We now analyze incentive compatibility properties of the algorithm. The main ingredient is in using the mapping $(\alpha, i) \mapsto \rho(\alpha)_i = \frac{\alpha_i}{||\alpha||}$ to choose what questions to ask. This mapping is known as the *spherical scoring rule*, and incentivizes truthful forecasts, in the sense that $\alpha = \text{argmax}_\beta \mathbf{E}_{i \sim \alpha}[\rho(\beta)_i]$. The spherical scoring rule satisfies the geometric property that

[13] The interior of $\rho(\mathcal{H})$ can be written as $\rho(\{\beta \in \Delta_n : v_1.\beta > 0, \ldots, v_T.\beta > 0\})$ for some $v_1, \ldots, v_T$, which is a non-empty intersection of open half-spaces as the agent's responses are required to be consistent.

[14] $||\alpha - \alpha^h|| \leq \frac{2\varepsilon}{n} \implies \sum_{i=1}^n (\alpha_i - \alpha_i^h)^2 \leq \frac{4\varepsilon^2}{n^2}$, so $(\alpha_i - \alpha_i^h) \leq \frac{2\varepsilon}{n}$ for each i, which implies that $||\alpha - \alpha^h||_{TV} \leq \varepsilon$.

$\mathbf{E}_{i\sim\alpha}[\rho(\beta)_i] = ||\alpha|| \cos(\alpha, \beta)$, where $\cos(\alpha, \beta)$ is the cosine of the angle formed by vectors α, β. Moreover, the spherical scoring rule is *effective* with respect to the renormalized L^2 metric, i.e. $\mathbf{E}_{i\sim\alpha}[\rho(\beta)_i] > \mathbf{E}_{i\sim\alpha}[\rho(\beta')_i]$ if and only if $||\rho(\alpha) - \rho(\beta)|| < ||\rho(\alpha) - \rho(\beta')||$. Note that the spherical scoring rule plays the role of the assignment function s in the more general preference framework.

We use the following straightforward observation bounding the deviation from the maximum possible payoff in terms of the renormalized L^2 distance from the true type.

▶ **Lemma 10.** *Let* $||\rho(\alpha) - \rho(\alpha')|| \leq \lambda$. *Then*

$$\mathbf{E}_{i\sim\alpha}[\rho(\alpha')_i] \geq \mathbf{E}_{i\sim\alpha}[\rho(\alpha)_i] - \frac{1}{2}\lambda^2.$$

Proof. We can write $||\rho(\alpha) - \rho(\alpha')||^2 = 2(1 - \cos(\alpha, \alpha'))$, so

$$\mathbf{E}_{i\sim\alpha}[\rho(\alpha')_i] = ||\alpha|| \cos(\alpha, \alpha') = ||\alpha|| \left(1 - \frac{1}{2}||\rho(\alpha) - \rho(\alpha')||^2\right) \geq \mathbf{E}_{i\sim\alpha}[\rho(\alpha)_i] - \frac{1}{2}\lambda^2. \quad \blacktriangleleft$$

▶ **Theorem 2.** *The IC learning complexity of* $\mathcal{A}$ *is* $O\left(n^{3/2} \log n \max\left(\log \frac{n}{\varepsilon}, \log \frac{1}{\tau}\right)\right)$.

Proof. Suppose we run $\mathcal{A}$ for T rounds to achieve (ε, δ)-learning. By Theorem 9, it holds with high probability that the hypothesis set obtained will be contained inside a small ball with respect to renormalized L^2 distance. More precisely, if $\mathcal{A}$ is given access to oracle $\hat{\alpha}$, and $\lambda = 2\varepsilon/n$, then

$$\Pr[\mathcal{H}^T(\alpha) \subseteq B(\alpha, \lambda)] \geq 1 - \delta,$$

where $\mathcal{H}^T(\alpha)$ is shorthand to denote a hypothesis set drawn from an execution of $\mathcal{A}_T(\hat{\alpha})$ and $B(\alpha, \lambda) = \{\alpha' : ||\rho(\alpha) - \rho(\alpha')|| \leq \lambda\}$.

By Lemma 10,

$$\Pr\left[\forall \alpha^h \in \mathcal{H}^T(\alpha), \mathbf{E}_{i\sim\alpha}[\rho(\alpha^h)_i] \geq \mathbf{E}_{i\sim\alpha}[\rho(\alpha)_i] - \frac{1}{2}\lambda^2\right] \geq \Pr[\mathcal{H}^T(\alpha) \subseteq B(\alpha, \lambda)] \geq 1 - \delta,$$

so any strategy can yield an advantage of at most $\frac{1}{2}\lambda^2 = \frac{2\varepsilon^2}{n^2}$ over truthful reporting. For $\varepsilon < n\sqrt{\tau}$ and $\delta = \nu$ we get (τ, ν)-incentive compatibility. The IC complexity is the query complexity of $(n\sqrt{\tau}, \nu)$-learning, which is $O(n^{3/2} \log n \log \frac{1}{\tau})$.

Thus, the number of queries required to simultaneously achieve (ε, δ)-learning and (τ, ν)-incentive compatibility is $O\left(n^{3/2} \log n \max\left(\log \frac{n}{\varepsilon}, \log \frac{1}{\tau}\right)\right)$. ◀

Our notion of incentive compatibility is approximate, and allows for small gains to the agent from misrepresenting their beliefs. We now demonstrate that, even though Theorem 2 allows for the possibility of gaining some advantage by playing strategically, we can ensure that with high probability any type learned by the analyst as a result of a strategic interaction will be sufficiently close to the true type that the analyst accurately learns the true type nonetheless.

Suppose the analyst wants to achieve ε learning accuracy, and additionally wants to guarantee that with probability at least $1 - \delta$, any best-responding agent will report a belief, or type, that is within ε total variation distance to the true type (note that this is a slightly different notion of incentive compatibility than the previous one). As usual, let α denote the agent's true type. Let $\varepsilon_0 < \frac{\varepsilon}{3}$, $\lambda = \frac{2\varepsilon_0}{n}$, $\delta_0 < 1 - \sqrt{1 - \delta}$, and run the algorithm to achieve $(\varepsilon_0, \delta_0)$-learning.

We first show that any misreport that is sufficiently far from the true type yields, with high probability, a strictly worse payoff than truthful reporting. Recall that $B(\alpha, \lambda)$ denotes the closed ball of radius λ centered at α with respect to the renormalized L^2 distance.

Suppose that $||\rho(\alpha) - \rho(\beta)|| > 2\lambda$, so that $B(\alpha, \lambda) \cap B(\beta, \lambda) = \emptyset$. Then, as the spherical scoring rule is effective with respect to renormalized L^2-distance,

$$\begin{aligned}
&\Pr\left[\forall \alpha^h \in \mathcal{H}^T(\alpha), \forall \beta^h \in \mathcal{H}^T(\beta), \mathbf{E}_{i\sim\alpha}[\rho(\alpha^h)_i] > \mathbf{E}_{i\sim\alpha}[\rho(\beta^h)_i]\right] \\
&\quad \geq \Pr[\mathcal{H}^T(\alpha) \subseteq B(\alpha, \lambda) \wedge \mathcal{H}^T(\beta) \subseteq B(\beta, \lambda)] \\
&\quad \geq (1 - \delta_0)^2 \\
&\quad \geq 1 - \delta,
\end{aligned}$$

so it holds with high probability that any such misreport yields a strictly worse payoff.

The remaining misreports are sufficiently close to the true type such that the analyst does not care if these allow the agent to increase his payoff. Indeed, if $||\rho(\alpha) - \rho(\beta)|| \leq 2\lambda$, since $\mathcal{H}^T(\alpha) \subseteq B(\alpha, \lambda)$ and $\mathcal{H}^T(\beta) \subseteq B(\beta, \lambda)$ with high probability, the triangle inequality yields

$$||\rho(\alpha) - \rho(\beta^h)|| \leq ||\rho(\alpha) - \rho(\beta)|| + ||\rho(\beta) - \rho(\beta^h)|| \leq 3\lambda,$$

so $||\alpha - \beta^h||_{TV} \leq 3\varepsilon_0 < \varepsilon$. Thus with probability at least $1 - \delta$, the analyst will learn a β^h such that $||\alpha - \beta^h||_{TV} \leq \varepsilon$ for any such misreport.

To summarize, the algorithm can be run for $O(n^{3/2} \log n \log \frac{n}{\varepsilon})$ rounds (the exact number of rounds would just be a small constant factor more than that required by the vanilla learning requirement) such that regardless of what strategy an agent may use to best respond during the interaction, with high probability the analyst will end up accurately learning the agents true type.

5 Concluding remarks

We have analyzed the incentive compatibility of active learning using data labeled by human subjects. Our results are directly applicable to the adaptive design of economic experiments that seek to estimate subjects' preference parameters. Our paper has discussed some of the leading areas of economic experimentation: estimation of risk aversion from multiple price lists, and belief elicitation using convex budgets and scoring rules. We highlight some challenges in making active learning compatible with incentives, but for the most part we offer satisfactory algorithmic solutions to the specific areas of experimentation we have focused on.

There are, of course, many other areas of application of active learning, and incentive issues will be important as long as the required training data is labeled by human subjects under incentivized conditions. To this end, we have introduced a general model of active learning under incentives: we believe that we are the first to do so, and we expect our findings to motivate additional investigations of the problems at the intersection of learning and incentives.

References

1 Jacob Abernethy, Yiling Chen, Chien-Ju Ho, and Bo Waggoner. Low-cost learning via active data procurement. In *Proceedings of the Sixteenth ACM Conference on Economics and Computation*, pages 619–636. ACM, 2015.

2 David S. Ahn, Syngjoo Choi, Douglas Gale, and Shachar Kariv. Estimating Ambiguity Aversion in a Portfolio Choice Experiment. *Quantitative Economics*, 5(2):195–223, 2014.

3 Steffen Andersen, Glenn W Harrison, Morten Igel Lau, and E Elisabet Rutström. Elicitation using multiple price list formats. *Experimental Economics*, 9(4):383–405, 2006.

4 James Andreoni, Marco Castillo, and Ragan Petrie. What Do Bargainers' Preferences Look Like? Experiments with a Convex Ultimatum Game. *American Economic Review*, 93(3):672–685, 2003.

5 James Andreoni and John Miller. Giving According to GARP: An Experimental Test of the Consistency of Preferences for Altruism. *Econometrica*, 70(2):737–753, 2002.

6 Ned Augenblick, Muriel Niederle, and Charles Sprenger. Working over Time: Dynamic Inconsistency in Real Effort Tasks *. *The Quarterly Journal of Economics*, 130(3):1067–1115, May 2015. doi:10.1093/qje/qjv020.

7 Yaron Azrieli, Christopher P Chambers, and Paul J Healy. Incentives in experiments: A theoretical analysis. *Journal of Political Economy*, 126(4):1472–1503, 2018.

8 Maria-Florina Balcan, Alina Beygelzimer, and John Langford. Agnostic active learning. *Journal of Computer and System Sciences*, 75(1):78–89, 2009.

9 Maria-Florina Balcan, Amit Daniely, Ruta Mehta, Ruth Urner, and Vijay V Vazirani. Learning economic parameters from revealed preferences. In *International Conference on Web and Internet Economics*, pages 338–353. Springer, 2014.

10 Pathikrit Basu. Learnability and Stochastic Choice. SSRN 3338991, 2019.

11 Pathikrit Basu and Federico Echenique. Learnability and Models of Decision Making under Uncertainty. In *Proceedings of the 2018 ACM Conference on Economics and Computation*, pages 53–53. ACM, 2018.

12 Eyal Beigman and Rakesh Vohra. Learning from revealed preference. In *Proceedings of the 7th ACM Conference on Electronic Commerce*, pages 36–42. ACM, 2006.

13 Hans P Binswanger. Attitudes toward risk: Theoretical implications of an experiment in rural India. *The Economic Journal*, 91(364):867–890, 1981.

14 Christopher P Chambers and Federico Echenique. Spherical Preferences. *arXiv preprint*, 2019. arXiv:1905.02917.

15 Christopher P Chambers and Nicholas S Lambert. Dynamic belief elicitation, 2017.

16 Jonathan Chapman, Erik Snowberg, Stephanie Wang, and Colin Camerer. Loss Attitudes in the US Population: Evidence from Dynamically Optimized Sequential Experimentation (DOSE). Technical report, National Bureau of Economic Research, 2018.

17 Zachary Chase and Siddharth Prasad. Learning Time Dependent Choice. In *10th Innovations in Theoretical Computer Science Conference (ITCS 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

18 Yiling Chen, Chara Podimata, Ariel D Procaccia, and Nisarg Shah. Strategyproof linear regression in high dimensions. In *Proceedings of the 2018 ACM Conference on Economics and Computation*, pages 9–26. ACM, 2018.

19 Syngjoo Choi, Raymond Fisman, Douglas Gale, and Shacher Kariv. Consistency and Heterogeneity of Individual Behavior under Uncertainty. *American Economic Review*, 97(5):1921–1938, 2007.

20 Vincent Conitzer. Prediction markets, mechanism design, and cooperative game theory. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pages 101–108. AUAI Press, 2009.

21 Sanjoy Dasgupta. Two faces of active learning. *Theoretical computer science*, 412(19):1767–1781, 2011.

22 Ofer Dekel, Felix Fischer, and Ariel D Procaccia. Incentive compatible regression learning. *Journal of Computer and System Sciences*, 76(8):759–777, 2010.

23 Mahmoud A El-Gamal, Richard D McKelvey, and Thomas R Palfrey. A Bayesian sequential experimental study of learning in games. *Journal of the American Statistical Association*, 88(422):428–435, 1993.

24 Daniel Friedman. Effective scoring rules for probabilistic forecasts. *Management Science*, 29(4):447–454, 1983.

25 Daniel Friedman, Sameh Habib, Duncan James, and Sean Crockett. Varieties of Risk Elicitation. Unpublished manuscript, 2018.

26 Steve Hanneke. A bound on the label complexity of agnostic active learning. In *Proceedings of the 24th international conference on Machine learning*, pages 353–360. ACM, 2007.

27 Moritz Hardt, Nimrod Megiddo, Christos Papadimitriou, and Mary Wootters. Strategic classification. In *Proceedings of the 2016 ACM conference on innovations in theoretical computer science (STOC)*, pages 111–122. ACM, 2016.

28 Charles A Holt and Susan K Laury. Risk aversion and incentive effects. *American economic review*, 92(5):1644–1655, 2002.

29 Taisuke Imai and Colin F Camerer. Estimating Time Preferences from Budget Set Choices Using Optimal Adaptive Design. Technical report, Caltech Working Paper, 2016.

30 Gil Kalai. Learnability and rationality of choice. *Journal of Economic theory*, 113(1):104–117, 2003.

31 Yang Liu and Yiling Chen. Machine-learning aided peer prediction. In *Proceedings of the 2017 ACM Conference on Economics and Computation*, pages 63–80. ACM, 2017.

32 Debajyoti Ray, Daniel Golovin, Andreas Krause, and Colin Camerer. Bayesian rapid optimal adaptive design (broad): Method and application distinguishing models of risky choice. California Institute of Technology working paper, 2012.

33 Leonard J Savage. *The foundations of statistics*. Courier Corporation, 1972.

34 John Von Neumann and Oskar Morgenstern. *Theory of games and economic behavior (commemorative edition)*. Princeton university press, 1953.

35 Morteza Zadimoghaddam and Aaron Roth. Efficiently learning from revealed preference. In *International Workshop on Internet and Network Economics*, pages 114–127. Springer, 2012.

Pseudorandomness and the Minimum Circuit Size Problem

Rahul Santhanam
Department of Computer Science, University of Oxford, United Kingdom
rahul.santhanam@cs.ox.ac.uk

Abstract

We explore the possibility of basing one-way functions on the average-case hardness of the fundamental Minimum Circuit Size Problem ($\mathsf{MCSP}[s]$), which asks whether a Boolean function on n bits specified by its truth table has circuits of size $s(n)$.

1. (Pseudorandomness from Zero-Error Average-Case Hardness) We show that for a given size function s, the following are equivalent: Pseudorandom distributions supported on strings describable by $s(O(n))$-size circuits exist; Hitting sets supported on strings describable by $s(O(n))$-size circuits exist; $\mathsf{MCSP}[s(O(n))]$ is zero-error average-case hard. Using similar techniques, we show that Feige's hypothesis for random k-CNFs implies that there is a pseudorandom distribution (with constant error) supported entirely on satisfiable formulas. Underlying our results is a general notion of *semantic sampling*, which might be of independent interest.
2. (A New Conjecture) In analogy to a known universal construction of succinct hitting sets against arbitrary polynomial-size adversaries, we propose the Universality Conjecture: there is a universal construction of succinct pseudorandom distributions against arbitrary polynomial-size adversaries. We show that under the Universality Conjecture, the following are equivalent: One-way functions exist; Natural proofs useful against sub-exponential size circuits do not exist; Learning polynomial-size circuits with membership queries over the uniform distribution is hard; $\mathsf{MCSP}[2^{\epsilon n}]$ is zero-error hard on average for some $\epsilon > 0$; Cryptographic succinct hitting set generators exist.
3. (Non-Black-Box Results) We show that for weak circuit classes $\mathfrak{C}$ against which there are natural proofs [44], pseudorandom functions secure against poly-size circuits in $\mathfrak{C}$ imply superpolynomial lower bounds in P against poly-size circuits in $\mathfrak{C}$. We also show that for a certain natural variant of MCSP, there is a polynomial-time reduction from approximating the problem well in the worst case to solving it on average. These results are shown using non-black-box techniques, and in the first case we show that there is no black-box proof of the result under standard crypto assumptions.

2012 ACM Subject Classification Theory of computation → Computational complexity and cryptography

Keywords and phrases Minimum Circuit Size Problem, Pseudorandomness, Average-case Complexity, Natural Proofs, Universality Conjecture

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.68

Funding This work was supported in part by the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2014)/ERC Grant Agreement No. 615075.

Acknowledgements Discussions with Shuichi Hirahara and Igor Carboni Oliveira were very helpful at an early stage of this research. Thanks to Igor for his detailed comments on an early draft of this work. Thanks to Shuichi for telling me about his independent work [23] and for alerting me to the relevance of auxiliary-input one-way functions. Thanks also to Andrej Bogdanov and Hoeteck Wee for e-mail correspondence about cryptographic hitting set generators. Conversations with Marco Carmosino, Manuel Sabin and Prashant Nalini Vasudevan were useful. Part of this work was done while participating in the Simons Institute Semester on Lower Bounds in Computational Complexity. I wish to thank the Simons Institute for their hospitality.

1 Introduction

We investigate the relationship between the complexity of the Minimum Circuit Size Problem (MCSP) [2] and the existence of various kinds of pseudorandom objects, such as hitting sets and pseudorandom sets, succinctly describable or not, in the cryptographic regime. There are two broad regimes of pseudorandom constructions: the complexity-theoretic regime, where the generator has more resources than the adversary, and the cryptographic regime, where the adversary can have more resources than the generator. In the complexity-theoretic regime, there is a beautiful theory [40, 31] giving equivalences between the existence of hard problems in E (linear exponential time), and explicit constructions of hitting set generators and pseudorandom generators.

In the cryptographic regime, the celebrated result of [22] gives an equivalence between pseudorandom generators and one-way functions, which are in many ways the fundamental cryptographic primitive. However, as we point out, there are several gaps in our understanding of the relationships between hardness and pseudorandomness in the cryptographic regime. To some extent, these gaps reflect an imperfect knowledge of average-case hardness for NP in general. In this paper, we make the argument that the average-case complexity of MCSP in particular is deeply relevant to pseudorandomness in the cryptographic regime.

We first discuss the questions that motivate us.

1.1 Our Questions

1.1.1 One-way Functions

From a cryptographer's point of view, one-way functions are an extremely robust and useful primitive, forming the basis for a range of important crypo constructions [34]. However, one-way functions do not fit very neatly into the complexity landscape. The hardness assumptions required to do cryptography seem stronger than the assumptions traditionally studied in complexity theory, and bridging this gap is an important open problem.

▶ **Question 1.** *Is there a natural well-studied problem in* NP *whose average-case hardness with respect to some natural distribution is equivalent to the existence of one-way functions?*

An attempt to base one-way functions on a standard complexity assumption was made by Ostrovsky and Wigderson [42], who showed that a weak variant of one-way functions called auxiliary-input one-way functions follow from the assumption that ZK (computational zero-knowledge) is not contained in BPP. It is unknown whether the existence of one-way functions is equivalent to the existence of auxiliary-input one-way functions. An equivalence result would imply that one-way functions could be based on worst-case hardness of ZK, which "would have a major impact on cryptography" [10].

▶ **Question 2.** *Are one-way functions equivalent to auxiliary-input one-way functions?*

One-way functions are also relevant to proof complexity. The main result of [44] is that if one-way functions exist, "natural proofs" of lower bounds against superpolynomial-size circuits do not. Here a "natural proof" is a property of Boolean functions that is satisfied with significant probability by a random Boolean function, and is moreover easy to check. Most standard algebraic and combinatorial lower bound techniques yield properties of this form.

Thus one-way functions rule out natural proofs of lower bounds, but could it be the case that neither one-way functions exist nor natural proofs? Both objects are useful, the first for cryptographic applications and the second for proof theory, so it would be nice to have at least one of them available, even if it is impossible to have both.

▶ **Question 3.** *Can we base one-way functions on the non-existence of natural proofs?*

1.1.2 Pseudorandomness

In many contexts where pseudorandomness is relevant, there is a distinction between a one-sided "hitting" notion where the goal is to find an explicit set of points that hits every "easy" dense set and a two-sided "pseudorandom" notion, where the question is to find an explicit distribution on points that approximates every easy set. The support of a pseudorandom distribution is a hitting set, but it is not clear in general how to get pseudorandom distributions from hitting sets. Sometimes the two notions coincide, and this makes for a cleaner theory. This is the case for example with the complexity-theoretic regime, where the results of [40, 31] imply that hitting set generators (HSGs) and pseudorandom generators (PRGs) are equivalent.

How about the cryptographic regime? What is the power of HSGs in this setting? Surprisingly, this question doesn't seem to have received much attention, though it seems a very natural one.

Part of the reason might be that, in contrast to PRGs, HSGs are not obviously robust. For instance, while we can increase the "stretch" of a PRG simply by iterating it, this does not seem to work with HSGs.

▶ **Question 4.** *Can HSGs be stretched in the cryptographic regime? For example, does a HSG with seed length $N^{0.5}$ (where N is the output length) imply one with seed length $N^{0.25}$?*

Question 4 might seem purely curiosity-driven, but in fact it has relevance to proof complexity and cryptography against non-deterministic adversaries, as suggested by Rudich [45]. The notion of cryptographic pseudorandomness does not make sense against non-deterministic adversaries, as a non-deterministic adversary could break the PRG by simply guessing the seed and checking that it maps to his/her input. However, a crucial observation is that the notion of HSG still does make sense against non-deterministic adversaries. Rudich defined the notion of a "demi-bit" to capture hitting sets in this setting, and asked questions about whether they can be stretched. Indeed, Question 4 is a version of his question for deterministic adversaries.

Cryptographic PRGs are equivalent to the existence of one-way functions, as shown by [22]. Is there a comparable connection for HSGs?

▶ **Question 5.** *In the cryptographic regime, is the existence of HSGs equivalent to some notion of average-case hardness for a decision or search problem?*

Finally, an ideal situation in terms of the cleanness of the resulting theory would be if HSGs are simply equivalent to PRGs.

▶ **Question 6.** *Are HSGs equivalent to PRGs in the cryptographic regime?*

1.1.3 The Minimum Circuit Size Problem and Learning

The Minimum Circuit Size Problem (MCSP), which asks if a given string is the truth table of a function with small circuits, is both fundamental and elusive. It asks a very natural question about the complexity of a string, and the naturalness and importance of the problem have been clear since work by Soviet researchers in the 50s and 60s [46]. However, the problem has also been elusive in terms of classifying its complexity. It is said that Levin delayed publication of his seminal paper on NP-completness because he was hoping to show MCSP is NP-complete [8]. Nearly 50 years later, we still lack even a clear belief about whether MCSP is NP-complete or not.

Natural proofs are essentially zero-error algorithms for MCSP on average over the uniform distribution [44, 24], so it is no surprise that the "natural proofs" paper revived work on MCSP and its place in the complexity of landscape. There has been a long line of recent works on the problem [33, 5, 3, 4, 7, 39, 53, 6, 25, 26, 24, 41]. In fact, the work by Williams [53] implicitly studies connections between MCSP and derandomization, but he is interested in connections to complexity-theoretic derandomization, while we are interested in the cryptographic regime, which raises some very different issues.

Despite recent work, major questions still remain about how the problem relates to other problems. Indeed, a number of the cited works deal with the difficulties of constructing reductions to MCSP.

▶ **Question 7.** *Is* MCSP *or one of its variants equivalent in complexity to some other complexity class or notion of independent interest?*

MCSP comes supplied with a parameter - the size s for which we wish to know whether the input truth table has circuits of that size. Often complexity results about MCSP are fairly robust to this parameter, but there is no formal justification known for this. It was suggested in [24] that the problem might be easier to show robust to its parameter in terms of average-case complexity.

▶ **Question 8.** *Does the average-case easiness of* MCSP *with parameter* $2^{n/4}$ *imply the average-case easiness of* MCSP *with parameter* $2^{n/2}$*?*

The MCSP problem is closely connected to Valiant's PAC learning model [52]. Learning algorithms can be thought of as methods to solve the *search* version of MCSP - they are given access to the truth table of a Boolean function with small circuits, and need to efficiently find a small circuit that approximates the truth table well. Valiant observed in his original paper that polynomial-size circuits are not learnable in his model under cryptographic assumptions, and it was observed in [43] that non-learnability follows from the assumption that one-way functions exist. However the precise complexity of learning in various models is still not known despite more than three decades of work [52, 35, 11, 10, 15, 50]. In particular, one can ask if there is a converse to the hardness result of [52, 43] for some natural learning model.

▶ **Question 9.** *For some natural learning model, does non-learnability of polynomial-size circuits imply the existence of one-way functions?*

MCSP has stubbornly resisted attempts to show that it is NP-complete. It is natural to wonder if it has structural features that distinguish it from other NP-complete problems, such as for example random self-reducibility or a worst-case to average-case reduction. This might give some evidence that MCSP is not NP-complete - it is known under standard complexity assumptions that NP-complete problems do not have black-box worst-case to average-case reductions [17, 12].

▶ **Question 10.** *Is there a worst-case to average-case reduction for* MCSP*? Or to ask a more relaxed question, does average-case easiness of* MCSP *imply non-trivial approximations for the problem?*

1.2 Results

1.2.1 Pseudorandomness from Zero-Error Average-Case Hardness

We begin by showing connections between zero-error average-case hardness of MCSP, and the existence of *succinct* hitting sets and pseudorandom distributions. As mentioned before, we are inspired by the connections between these notions in the complexity-theoretic setting [40, 31], and are looking for analogous results in the cryptographic setting.

Let us first explain what we mean by zero-error average-case hardness of MCSP. Given any size function $s : \mathbb{N} \to \mathbb{N}$, MCSP$[s]$ has a very natural distribution on inputs associated with it, namely the uniform distribution. MCSP$[s]$ is heavily biased over the uniform distribution since the overwhelming majority of truth tables correspond to hard Boolean functions. So it does not make sense to study bounded-error notions of average-case hardness over the uniform distribution - the trivial algorithm that always says "no" will do very well. Instead, we consider zero-error algorithms, i.e., algorithms that always output the correct answer or "?", and output the correct answer with noticeable probability over the uniform distribution.

It turns out that this notion of average-case hardness is fairly robust. It was shown in [24] that average-case hardness is essentially equivalent to the non-existence of Razborov-Rudich natural proofs. Thus the main result of Razborov and Rudich [44] showing that natural proofs don't exist if one-way functions exist can also be interpreted as showing zero-error average-case hardness of MCSP$[s]$ for any $s = 2^{\Omega(n)}$ under the assumption that one-way functions exist. A major motivation for this paper is the question of whether the *converse* holds, i.e., whether one-way functions can be based on zero-error average-case hardness of MCSP$[s]$ for reasonable size functions s. Indeed, by the connection we just mentioned with natural proofs, this is equivalent to Question 3.

It is known in the cryptographic setting that the existence of pseudorandom functions of circuit complexity $2^{\epsilon n}$ for any $\epsilon > 0$ is equivalent to the existence of one-way functions [22, 20]. In order to make progress toward basing one-way functions on zero-error average-case hardness of MCSP, we first show how to derive weaker versions of pseudorandom functions. Succinct hitting sets and succinct pseudorandom distributions are examples of such weaker objects, as we explain below.

A succinct set (resp. distribution) is a set of (resp. distribution over) strings, each of which, when interpreted as the truth table of a Boolean function, has circuits that are not too large. Succinct hitting sets are simply succinct sets that are hitting sets against arbitrary poly-size adversaries. A succinct pseudorandom distribution is a succinct distribution that is pseudorandom against arbitrary poly-size adversaries.

To compare these notions with pseudorandom functions, we note that pseudorandom functions are essentially equivalent to succinct PRGs, i.e., succinct pseudorandom distributions that are efficiently samplable. Similarly, a succinct HSG is an efficiently computable function whose range is a collection of succinct hitting sets.

It is fairly straightforward to show that zero-error average-case hardness of MCSP$[\tilde{O}(s)]$, $\tilde{O}(s(n))$-succinct hitting sets and $\tilde{O}(s(n))$-succinct HSGs are all equivalent. We give this argument in Subsection 3.1 of Section 3. The equivalence between the second and third notions follows from the existence of *universal succinct HSGs*, i.e., a fixed polynomial-time construction that is guaranteed to be a succinct HSG if succinct hitting sets exist. This is essentially folklore - the idea is to use the mapping from circuits to the truth tables of functions they compute. Indeed, this has been a very fruitful technique in complexity theory - the "easy witness" method of [32, 29].

What seems less straightforward is to get a connection between succinct hitting sets and succinct pseudorandom sets. This is what we manage to show.

▶ **Theorem 1** (Informal Statement). *The following are equivalent:*

1. *For all $\epsilon > 0$, there are succinct hitting sets supported on truth tables of circuit complexity $2^{\epsilon n}$.*
2. *For all $\epsilon > 0$,* MCSP *with parameter $2^{\epsilon n}$ is zero-error hard on average.*
3. *For all $\epsilon > 0$, there are succinct pseudorandom distributions supported on truth tables of circuit complexity $2^{\epsilon n}$.*

It is perhaps surprising that based on a *zero-error* average-case notion of hardness for MCSP, we are able to get pseudorandom distributions that are indistinguishable from random with respect to *bounded two-sided error*.

The proof of Theorem 1 proceeds via a certain Sampler-Distinguisher game we define here, inspired by the PRF-Distinguisher game in [41]. The analysis of the game uses the approximate Min-Max theorem of [9, 38] and is fairly general. We note that the approximate Min-Max theorem has been used in many different contexts in complexity theory and pseudorandomness (see [51]), but as far as we are aware, our use of it here to derive pseudorandomness from zero-error average-case hardness is novel.

Indeed our techniques can be used to establish an analogous result for Satisfiability based on Feige's hypothesis [16] that random k-CNFs of linear density are hard to refute. Feige's hypothesis is essentially a hypothesis about zero-error average-case hardness of k-SAT under a natural distribution on k-CNFs, just as the non-existence of natural proofs is a hypothesis about zero-error average-case hardness of MCSP. One way of stating the hypothesis is that errorless polynomial-time algorithms, which always output the correct answer or "?" and output the correct answer with high probability over randomly chosen k-CNFs of linear density, do not exist.

In this case again, we get a consequence for pseudorandom distributions. Just as the pseudorandom distributions obtained in Theorem 1 are succinct, i.e., supported on YES instances of $\mathsf{MCSP}[s]$, here the pseudorandom distributions obtained are supported on satisfiable instances.

▶ **Theorem 2** (Informal Statement). *For any fixed integer k, if random k-CNFs with any linear number of clauses are hard to refute for non-uniform poly-time algorithms, then for each $\epsilon > 0$ there is a pseudorandom distribution over satisfiable formulas with error ϵ.*

We abstract out the main idea behind the proofs of Theorems 1 and 2 to show a more general connection between zero-error average-case hardness for a language $Q \subseteq \{0,1\}^*$ and the existence of pseudorandom distributions supported on a language $Q' \subseteq \{0,1\}^*$. We show that such a connection exists whenever there are samplers satisfying a certain *semantic* condition. Recall that a sampler f from n bits to m bits with seed length ℓ is a polynomial-time computable function such that for any bounded function g on m bits, for *most* inputs x of length n, the expectation of $g(f(x, U_\ell))$ is close to the expectation of $g(U_m)$. Samplers are closely related to randomness extractors [49]. We consider samplers with an additional condition: whenever $x \in Q$, for every y of length ℓ, $f(x, y) \in Q'$. We call such samplers (Q, Q')-semantic samplers, and show that the existence of semantic samplers with good parameters implies a strong connection between zero-error average-case hardness and pseudorandomness.

We note that this connection between samplers and pseudorandomness is *different* to the well-known connection of Trevisan [47] between hardness-based pseudorandom generators and extractors. Indeed, Trevisan's connection does not involve any semantic property of the extractor.

1.2.2 A New Conjecture and Its Consequences

Theorem 1 partly addresses Questions 5 and 6 in Subsection 1.1, but it is not clear what it says about the others. Motivated by a belief that the questions in Subsection 1.1 are all connected and part of a unified picture, we propose a natural conjecture about universal succinct PRGs that results in such a picture.

A universal succinct PRG is a fixed polynomial-time computable function f such that if succinct pseudorandom distributions exist, then f induces such distributions on uniformly chosen seeds.

▶ **Universality Conjecture** (Informal Statement). *There exist universal succinct PRGs.*

We briefly discuss the context for the Conjecture. Universality is a phenomenon that is widely observed in complexity theory and the foundations of cryptography. For example, there are universal one-way functions [36], and via the equivalence between one-way functions and PRGs [22], there are universal PRGs in the cryptographic setting. In the complexity-theoretic setting, universal HSGs and PRGs follow from the equivalence of HSGs, PRGs and circuit lower bounds for $\mathsf{E} \stackrel{\text{def}}{=} \mathsf{DTIME}(2^{O(n)})$, together with the existence of complete problems for $\mathbb{E}$. Given the variety of such universal examples, and the variety of ways for establishing that they exist, it is perhaps not unreasonable to expect them in the context of succinct PRGs.

An even closer analogy is to the case of succinct HSGs, which are the one-sided error version of succinct PRGs. As described in the previous subsection, universal succinct HSGs can be shown to exist by interpreting circuits succinctly describing hitting sets as seeds to a hitting set generator. This is a folklore argument that we formalize in Section 3 as Proposition 9. The Universality Conjecture is simply the analogue of Proposition 9 for pseudorandom distributions.

Next we turn to the consequences of our Conjecture for the questions raised in the introduction. We make a few clarifications about notation in the informal statement of our main result below. N always refer to the output size of some generator and is a power of 2, and $n \stackrel{\text{def}}{=} \log(N)$. By default, we take the succinctness parameter (i.e., the size of circuits representing each element of the range) to be the same as the seed length. To clarify any further notational issues, please refer to Section 2.

▶ **Theorem 3** (Informal Statement). *If the Univerality Conjecture is true, then the following hold:*

1. *One-way functions exist iff there is an $\epsilon < 1$ such that* MCSP *with parameter $2^{\epsilon n}$ is zero-error hard on average.*
2. *One-way functions exist iff auxiliary one-way functions exist.*
3. *One-way functions exist iff natural proofs against* SIZE(poly) *do not exist.*
4. *For any $0 < \epsilon < \delta < 1$, a succinct HSG with seed length N^{δ} implies a succinct HSG with seed length N^{ϵ}.*
5. *For any $\epsilon < 1$, a succinct HSG with seed length N^{ϵ} exists iff* MCSP *with parameter $2^{\epsilon n}$ is zero-error hard on average.*
6. *For any $\epsilon < 1$, a succinct HSG with seed length N^{ϵ} exists iff a PRG with seed length N^{ϵ} exists.*
7. *For any $0 < \epsilon < \delta$,* MCSP *with parameter $2^{\epsilon n}$ is zero-error hard on average iff* MCSP *with parameter $2^{\delta n}$ is zero-error hard on average.*
8. *One-way functions exist iff polynomial-size circuits cannot be learned with membership queries under the uniform distribution in polynomial time.*

Items 1 to 3 of Theorem 3 answer Questions 1 to 3. Items 4 to 6 answer Questions 4 to 6, but for *succinct* HSGs rather than HSGs. Question 7 is also answered by Item 1. Questions 8 and 9 are answered by Items 7 and 8.

Of the items in Theorem 3, it is perhaps Item 1 which is of most interest. Most candidate one-way functions are based on problems in $\mathsf{NP} \cap \mathsf{coNP}$. MCSP, however, is believed by many to be NP-complete, and therefore unlikely to be in coNP. Indeed, a conjecture of Rudich [45] even asserts that MCSP is hard on average for coNP.

Item 1 also has implications for Impagliazzo's "Five Worlds" [28]. In particular, if the Conjecture were true and we could base one-way functions on average-case hardness of MCSP, we would get evidence against the existence of Pessiland: a world where there are problems that are hard on average, but one-way functions do not exist.

We briefly explain how the Conjecture helps to show these results. For every item in Theorem 3, one of the two directions of the equivalence was known before, and it is the Conjecture, together with Theorem 1, that enables us to show the reverse direction. The crucial aspect of the Conjecture is that it allows us to derive succinct PRGs from succinct pseudorandom distributions. Once we have a PRG, we are able to stretch the PRG using standard techniques, and this enables us to close the chain of implications between average-case hardness of MCSP, succinct HSGs and succinct PRGs.

We discuss some reasons why it might be useful to consider the Universality Conjecture.

First note that common beliefs in the crypto and complexity communities support the Conjecture. Indeed, most complexity theorists and cryptographers believe that one-way functions exist. If one-way functions exist, by [22] and [20], pseudorandom function generators exist, and any pseudorandom function generator is trivially a universal succinct PRG.

Of course, the issue is not simply *truth* but also *provability*. Is it likely the Universality Conjecture will be proved in the near future? We do not have a strong belief about this, but there is at least some reason to hope that more sophisticated versions of the proof technique of Theorem 1 might help. More precisely, understanding uniformity and succinctness of approximate strategies for Min-Max games [9, 38] is a possible direction.

Regardless of whether the Conjecture will be settled in the near future, it could function as an organizing principle connecting various fundamental phenomena that we still don't understand well, including the complexity of MCSP, the hardness of learning, the relationship between uniform and non-uniform versions of one-way functions, the structure of zero knowledge, reducibilities between average-case problems over the uniform distribution, and the role of pseudorandomness in proof complexity, among others.

1.2.3 Non-Black-Box Results

Finally, we show a couple of *non-black-box* results about MCSP. As a meta-complexity problem, i.e., a problem whose instances themselves encode computations, MCSP seems particularly amenable to non-black-box techniques. Hopefully this amenability will come in useful in establishing strong unconditional connections between the hardness of MCSP and the existence of one-way functions. Basing one-way functions on NP-hardness in a black-box fashion has unlikely consequences [1], so if MCSP did turn out to be NP-complete and we wished to base one-way functions on its *worst-case* hardness, we would need to use non-black-box techniques.

Our first result is about circuit classes that are weak in the sense that there are natural proofs useful against them. For such circuit classes (which include AC^0, $\mathsf{AC}^0[p]$ for prime p, etc.), we establish a surprising implication from lower bounds for MCSP to lower bounds for P.

▶ **Theorem 4** (Informal Statement). *Let $\mathfrak{D}$ be any circuit class closed under projections for which there are natural proofs against $\mathfrak{D}$. If there is a constant k such that* MCSP *with parameter n^k is zero-error hard on average against $\mathfrak{D}$, then* P *is not contained in $\mathfrak{D}$.*

The proof of Theorem 4 is non black-box, i.e., it does not work when the circuit class $\mathfrak{D}$ against which we are arguing is given access to an oracle. Indeed, we can prove that under standard crypto assumptions, there is no black-box implication of the sort we show.

The reason we find the implication in Theorem 4 somewhat surprising is that the hypothesis is about the hardness of a problem in NP and unlikely to be in P, indeed even believed by many to be NP-complete. Yet the conclusion is about super-polynomial lower bounds within P!

Our next result addresses the relaxed version of Question 10. We observe that a recent search-to-decision reduction of [14] for a slight variant of MCSP called AveMCSP (where the question is whether there is a small circuit that computes the input truth table correctly on a 0.9 fraction of inputs) actually gives an approximation to average-case reduction. We note that Shuichi Hirahara [23] independently observed an analogous approximation to average-case reduction for the standard version of MCSP, based on [13], however the approximation factor there is much larger.

▶ **Theorem 5** (Informal Statement). *There is an approximation to average-case reduction for* AveMCSP.

Theorem 5 has implications for the NP-hardness of problems such as MCSP and AveMCSP. For several NP-hard problems, the theory of probabilistically checkable proofs establishes strong inapproximability results under the NP $\neq$ P assumption. Thus an approximation to average-case reduction can be considered morally similar to a worst-case to average-case reduction. It is known under standard complexity assumptions that NP-complete problems do not have black-box worst-case to average-case reductions [17, 12].

Does this suggest that AveMCSP is unlikely to be NP-complete? Not quite! The proof of Theorem 5 is *also* non black-box, even though the ideas are quite different from the proof of Theorem 4.

1.3 Related Work

The first paper to connect the complexity of MCSP with the existence of one-way functions was the "natural proofs" paper of Razborov and Rudich [44]. Razborov and Rudich do not explicitly consider MCSP - indeed, this problem was only defined in subsequent work of Kabanets and Cai [33]. The main result of [44] states that exponentially-hard one-way functions imply the non-existence of natural proofs with poly-size constructibility that are useful against polynomial-size circuits. This can be re-interpreted [24] as saying that if exponentially-hard one-way functions exist, then MCSP[poly(n)] is zero-error hard on average. Thus an average-case algorithm for MCSP can be used to break *any* one-way function. The main question we consider in this paper is whether the converse holds.

Various works have attempted to connect the existence of one-way functions and their variants with other complexity notions. Ostrovsky and Wigderson [42] showed that if ZK $\neq$ BPP, then auxiliary-input one-way functions exist. They also showed that if ZK is hard for BPP on average (in the bounded-error sense), then one-way functions exist. [30] and [11] (see also [10]) show equivalences between the existence of one-way functions and certain average-case hardness assumptions for learning.

More recently, interest in the complexity of MCSP and its connections with learning and pseudorandomness has been re-awakened by the result of [13] which shows that natural proofs useful against a circuit class $\mathfrak{C}$ imply efficient learning algorithms for $\mathfrak{C}$, under certain reasonable conditions on $\mathfrak{C}$. Building on this, [41] show equivalences between various forms of learning, as well as a dichotomy between learning and pseudorandomness in a certain parameter regime.

A very recent work of Hirahara [23] (who obtained his results independently from ours) builds on [13] to show that solving MCSP on average in the zero-error sense implies efficient non-trivial approximability of the minimum circuit size. Thus, if MCSP were NP-hard even to approximate non-trivially, the existence of hard-on-average problems in NP would follow. This suggests the possibility of excluding Impagliazzo's Heuristica world [28] where NP is hard in the worst case but not hard on average by studying MCSP and showing that it is NP-hard to approximate.

In contrast to [23], our focus in this paper is on finding evidence against the existence of Pessiland: another world of Impagliazzo's [28] where there are hard-on-average problems but one-way functions do not exist.

2 Preliminaries

2.1 Notation

For convenience, we use $f\colon K \to N$ to denote a function mapping K bits to N bits, i.e., $f\colon \{0,1\}^K \to \{0,1\}^N$. We let $\mathcal{F}_{K\to N}$ denote the family of all functions $f\colon K \to N$. We sometimes view a boolean function $f\colon N \to 1$ as a subset of $\{0,1\}^N$.

Throughout this paper, we use capital letters to denote the input length and output length of a function when we are interested in interpreting the output as the truth table of a Boolean function. In such a case, we will use a lowercase letter to denote the logarithm of the number represented by the corresponding uppercase letter.

We let $\mathcal{U}_L$ denote the uniform distribution over $\{0,1\}^L$. We occasionally abuse notation and identify a set S with the uniform distribution over the set. We say that a function $f\colon K \to N$ ε-fools a family of functions $\mathcal{D} \subseteq \mathcal{F}_{N\to 1}$ if $|\Pr[g(f(\mathcal{U}_K)) = 1] - \Pr[g(\mathcal{U}_N) = 1]| \leq \varepsilon$ for every $g \in \mathcal{D}$.

A function $g\colon N \to 1$ is γ-dense if $\Pr[g(\mathcal{U}_N) = 1] \geq \gamma$. We say that $f \in \mathcal{F}_{K\to N}$ *hits* g is $f(\{0,1\}^K) \cap g^{-1}(1) \neq \emptyset$.

Given a Boolean function $f \in \mathcal{F}_{n\to 1}$, $\mathtt{tt}(f)$ is the 2^n-bit string which represents the truth table of f in the standard way, and conversely, given a string $y \in \{0,1\}^{2^n}$, $\mathtt{fn}(y)$ is the Boolean function in $\mathcal{F}_{n\to 1}$ whose truth table is represented by y.

Given a circuit class $\mathfrak{C}$, we use $\mathfrak{C}[s(n)]$ to refer to the class of functions computed by $\mathfrak{C}$-circuits of size $s(n)$. Given a language $L \subseteq \{0,1\}^*$, co-L denotes the complement of L.

We say a circuit class $\mathfrak{C}$ is standard if there is a quasi-linear time Turing machine which, given a representation $\tilde{C}$ of a circuit C from the class and an input x, computes $C(x)$. All commonly studied circuit classes contained in the class of general Boolean circuits are standard.

For a size bound $s\colon \mathbb{N} \to \mathbb{N}$, we use quasi-$s(n)$ to denote a function of growth rate $s(n) \cdot \mathsf{poly}(\log s(n))$.

The density of a set $A \subseteq \{0,1\}^N$ is $|A|/2^N$. Given a language L and an integer n, $L_n = L \cap \{0,1\}^n$ denotes the slice of L at length n.

2.2 Pseudorandomness and Average-Case Hardness

Let $K : \mathbb{N} \to \mathbb{N}$ be a function such that $K(N) < N$ for all N, and $\epsilon : \mathbb{N} \to [0,1]$ be a function. Moreover, let $\mathfrak{C}$ be a complexity class and $\mathfrak{D}$ be a class of functions. A $\mathfrak{C}$-PRG against $\mathfrak{D}$ with seed length K and error ϵ is a sequence $f = \{f_N\}_{N\in\mathbb{N}}$ of functions such that (i) $f \in \mathcal{F}_{K(N)\to N}$; (ii) $f \in \mathfrak{C}$; and (iii) the output of f_N $\epsilon(N)$-fools every function in $\mathcal{F}_{N\to 1} \cap \mathfrak{D}$, whenever N is sufficiently large. If ϵ is left unspecified, it will be taken to be $1/N^{\omega(1)}$ by default. Similarly $\mathfrak{C}$ will be taken to be $\mathsf{DTIME}(\mathrm{poly}(N))$ by default, and $\mathfrak{D}$ will be taken to be $\mathsf{SIZE}(\mathrm{poly}(N))$ by default. Note that in this default situation, the generator is computable in a *fixed* polynomial time in its output length, but must be secure against arbitrary poly-size distinguishers.

We say $f = \{f_N\}_{N\in\mathbb{N}}$ is a $\mathfrak{C}$-HSG against $\mathfrak{D}$ if instead of condition (iii) above the output of f_N hits every set $A_N \subseteq \{0,1\}^N$ of density $\epsilon(N)$ in $\mathfrak{D}$. Again the error parameter is taken to be $1/N^{\omega(1)}$ by default.

It is instructive to consider the following examples: standard cryptographic PRGs are default PRGs with seed length $K(N) = N^{\Omega(1)}$, while complexity-theoretic (Nisan-Wigderson) PRGs are E-PRGs against $\mathsf{SIZE}(N)$ with error $1/N$, where the seed length can be anything between $\log N$ and $N-1$.

We say the seed length K is non-trivial if $K(N) < N$.

We will be interested in *succinct* pseudorandom distributions and hitting sets. Let $\mathfrak{C}$ be a circuit class and $\epsilon : \mathbb{N} \to [0,1]$ be an error bound. We say that there are $\mathfrak{C}$-succinct pseudorandom distributions against $\mathfrak{D}$ with error ϵ if there is a set $S \subseteq \{0,1\}^*$ and a probability distribution μ_S supported on S such that for each N a power of 2, (i) For each $y \in S_N$, $\mathtt{fn}(y) \in \mathfrak{C}$; and (ii) $|\Pr_{y\leftarrow\mu_S}[g(y)=1] - \Pr_{y\leftarrow\mathcal{U}_N}[g(y)=1]| \leq \epsilon(N)$ for every $g \in \mathcal{D}$. By default, we will take $\epsilon(N)$ to be $1/N^{\omega(1)}$ as usual.

Similarly, we say that there are $\mathfrak{C}$-succinct hitting sets against $\mathfrak{D}$ with error ϵ if there is a set $S \subseteq \{0,1\}^*$ such that for each N a power of 2, (i) For each $y \in S_N$, $\mathtt{fn}(y) \in \mathfrak{C}$; and (ii) For each set $A_N \subseteq \{0,1\}^N$ of density $\epsilon(N)$, S_N has non-empty intersection with A_N.

A $\mathfrak{C}$-succinct PRG is simply a PRG that induces a $\mathfrak{C}$-succinct pseudorandom distributions when a seed of any given length is chosen uniformly at random, and a $\mathfrak{C}$-succinct HSG is a HSG whose range is a collection of $\mathfrak{C}$-succinct hitting sets.

▶ **Definition 6** (Universal Succinct Generators). *Let $\mathfrak{C}$ be a circuit class and $K : \mathbb{N} \to \mathbb{N}$ be a function. A universal $\mathfrak{C}$-succinct HSG with seed length K is a poly-time computable sequence $f = \{f_N\}_{N\in\mathbb{N}}$ of functions from $K(N)$ bits to N bits such that if there are $\mathfrak{C}$-succinct hitting sets against $\mathsf{SIZE}(\mathrm{poly})$ with error $\epsilon(N)$, then f is a $\mathfrak{C}$-succinct HSG with error $\epsilon(N)$. A universal $\mathfrak{C}$-succinct PRG with seed length K is a poly-time computable family f of functions from $K(N)$ bits to N bits such that if there are $\mathfrak{C}$-succinct pseudorandom distributions against $\mathsf{SIZE}(\mathrm{poly})$ with error $\epsilon(N)$, then f is a $\mathfrak{C}$-succinct PRG with error $\epsilon(N)$. A non-uniform universal $\mathfrak{C}$-succinct PRG with seed length K is a poly-size computable family f of functions from $K(N)$ bits to N bits such that if there are $\mathfrak{C}$-succinct pseudorandom distributions against $\mathsf{SIZE}(\mathrm{poly})$ with error $\epsilon(N)$, then f is a $\mathfrak{C}$-succinct $\mathsf{SIZE}(\mathrm{poly})$-PRG with error $\epsilon(N)$.*

Given $\epsilon : \mathbb{N} \to \mathbb{R}$, a one-way function with security $\epsilon(N)$ is a polynomial-time computable sequence of functions $f = \{f_N : K(N) \to N\}$ for some $K(N) = N^{\Omega(1)}$, such that for any sequence $\{C_N\}$ of polynomial-size circuits, for large enough N, $\Pr_{x\sim U_N}[f_N(C_N(1^N, f(x))) = f_N(x)] \leq \epsilon(N)$. A one-way function is called *weak* if it has security $1 - 1/\mathrm{poly}(N)$, and *strong* if it has security $1/N^{\omega(1)}$. One-way functions are often defined against uniform adversaries; in this work, we only consider security against non-uniform adversaries.

Informally, an auxiliary-input one-way function is a poly-time computable sequence of functions f with an "auxiliary" input z such that for any poly-size adversary, there is an infinite set of auxiliary inputs z for which f is hard to invert. For a formal definition and a good discussion of the significance of auxiliary-input one-way functions, see [48].

We require a notion of zero-error average-case hardness for languages. Given a parameter $\epsilon : \mathbb{N} \to [0,1]$, a language $Q \subseteq \{0,1\}^*$, and a circuit class $\mathfrak{D}$, we say that Q is zero-error average-case feasible for $\mathfrak{D}$ with success probability ϵ if there is a sequence of circuits $D_N \in \mathfrak{D}$ such that for each N, D_N always outputs 0,1 or "?", never outputs the wrong answer for any input to Q, and outputs "?" with probability at most $1 - \epsilon(N)$. We say Q is zero-error average-case easy if it is average-case feasible for $\mathsf{SIZE}[\text{poly}]$ with success probability $1/N^{O(1)}$.

We say that Q is zero-error average-case infeasible for $\mathfrak{D}$ with success probability ϵ if for every sequence of circuits $D_N \in \mathfrak{D}$ such that D_N always outputs 0,1 or "?", for large enough N, D_N either outputs the wrong answer for some input to Q or outputs "?" with probability greater than $1 - \epsilon(N)$. We say Q is zero-error average-case hard if it is average-case infeasible for $\mathsf{SIZE}[\text{poly}]$ with success probability $1/N^{O(1)}$.

Note that hardness is not just defined as the complement of easiness - hardness and easiness are both required to hold on almost all input lengths.

2.3 MCSP, Natural Proofs and Learning

$\mathsf{MCSP}[s(n)]$ denotes the Minimum Circuit Size Problem for a size parameter $s(n)$, where the truth table given as input to the problem has size 2^n. In other words, an input truth table is a positive instance if and only if it is computed by a circuit of size at most $s(n)$. This notation is adopted for convenience. Thus the most interesting functions $s(n)$ would satisfy $n \leq s(n) \leq 2^n$. Given a class $\mathfrak{C}$, $\mathfrak{C}$-$\mathsf{MCSP}[s(n)]$ denotes MCSP for $\mathfrak{C}$-circuits. In case the input size N is not a power of two, we consider the input truth table to be the first $2^{\lfloor \log(N) \rfloor}$ bits of the input.

We also require some variants of MCSP. Given an approximation parameter $\delta : \mathbb{N} \to [0,1]$, and size functions $c, s : \mathbb{N} \to \mathbb{N}$ with $n \leq c(n) \leq s(n) \leq 2^n$ for all n, δ-$\mathsf{AveMCSP}[c,s]$ is the following promise problem: YES instances are truth tables of Boolean functions that can be $\delta(n)$-approximated by circuits of size at most $c(n)$, and NO instances are truth tables of Boolean functions that cannot be δ-approximated by circuits of size at most $s(n)$. When δ is left unspecified, we take it to be 0.9 for convenience.

We say that $\mathfrak{R} = \{\mathcal{R}_n\}_{n \in \mathbb{N}}$ is a *combinatorial property* (of boolean functions) if $\mathcal{R}_n \subseteq \mathcal{F}_n$ for all n. We use $L_{\mathfrak{R}}$ to denote the language of truth-tables of functions in $\mathfrak{R}$. Formally, $L_{\mathfrak{R}} = \{y \mid y = \mathtt{tt}(f) \text{ for some } f \in \mathcal{R}_n \text{ and } n \in \mathbb{N}\}$.

▶ **Definition 7** (Natural Properties [44]). *Let $\mathfrak{R} = \{\mathcal{R}_n\}$ be a combinatorial property, $\mathfrak{C}$ a circuit class, and $\mathfrak{D}$ a (uniform or non-uniform) complexity class. We say that $\mathfrak{R}$ is a $\mathfrak{D}$-natural property useful against $\mathfrak{C}[s(n)]$ if there is $n_0 \in \mathbb{N}$ such that the following holds*:

- **(i)** Constructivity. $L_{\mathfrak{R}} \in \mathfrak{D}$.
- **(ii)** Density. *For every $n \geq n_0$,* $\Pr_{f \sim \mathcal{F}_n}[f \in \mathcal{R}_n] \geq 1/2$.
- **(iii)** Usefulness. *For every $n \geq n_0$, we have $\mathcal{R}_n \cap \mathcal{C}_n[s(n)] = \emptyset$.*

By default, $\mathfrak{D}$ is taken to be $\mathsf{SIZE}[\text{poly}]$.

It has been observed that the density parameter in the definition above can be amplified using a straightforward reduction (cf. [13]).

▶ **Proposition 8** ([24]). *Let $s : \mathbb{N} \to \mathbb{N}$ be a size function.* $\mathsf{MCSP}[s(n)]$ *is zero-error easy on average iff there are* $\mathsf{SIZE}(\text{poly})$*-natural properties useful against* $\mathsf{SIZE}[s(n)]$.

We give only a brief description of learning in Valiant's PAC model. We will be concerned here only with learning using membership queries under the uniform distribution. A learning algorithm in this context is a probabilistic oracle algorithm making queries to an oracle for some function f. Given a circuit class $\mathfrak{C}$ and a function $T : \mathbb{N} \to \mathbb{N}$, a time T learner for $\mathfrak{C}$ is an oracle algorithm which given input 1^n and oracle access to a function f from $\mathcal{C}$ halts in time $T(n)$ and outputs with high probability a circuit computing f correctly on a 0.9 fraction of inputs of length n.

3 Hitting Sets, Pseudorandom Distributions and the Hardness of MCSP

3.1 Succinct HSGs and MCSP

We first observe that there is a universal succinct HSG.

▶ **Proposition 9.** *Let $\mathfrak{C}$ be any standard circuit class, and $s : \mathbb{N} \to \mathbb{N}$ be any function such that $n \leq s(n) \leq 2^n$ for all n. There is a universal $\mathfrak{C}[s(n)]$-succinct HSG U with seed length* quasi-$s(n)$.

Proof. The idea of the proof is simple. We define a HSG U with seed length quasi-$s(n)$ that interprets its seed as the representation of a $\mathfrak{C}$-circuit of size $s(n)$, and outputs the truth table of the function computed by this circuit. U can be computed in time 2^nquasi-$s(n)$, which is quasi-quadratic in $N = 2^n$.

We need to show that U is a universal $\mathfrak{C}[s(n)]$-succinct HSG. The $\mathfrak{C}[s(n)]$-succinctness is immediate from the definition of the HSG - every output of the HSG is the truth table of a Boolean function in $\mathfrak{C}[s(n)]$. For the universality, suppose that there is some collection H of $\mathfrak{C}[s(n)]$-succinct hitting sets. Every string in this collection is the truth table of a Boolean function in $\mathfrak{C}[s(n)]$. Hence H is contained in the range of U, from which it follows that the range of U is a collection of hitting sits. ◀

Next we observe that the existence of succinct hitting set generators is equivalent to hardness for MCSP. This observation is closely related to similar observations in the theory of Boolean or arithmetic circuit complexity and meta-mathematics of lower bounds [53, 21, 18].

▶ **Proposition 10.** *Let $\mathfrak{C}$ be a standard circuit class and $\mathfrak{D}$ be a family of functions. Moreover, let $s(n) \geq n$. For every large n, the following are equivalent:*

1. *There are* $\mathfrak{C}$[quasi-$s(n)$]-succinct *hitting sets against $\mathfrak{D}$*
2. *There is a* $\mathfrak{C}$[quasi-$s(n)$]-succinct *HSG with seed length* quasi-$s(n)$ *against $\mathfrak{D}$*
3. $\mathfrak{C}$-MCSP[quasi-$s(n)$] *is zero-error average-case hard against $\mathfrak{D}$.*

Proof. The second item follows from the first by Proposition 9.

Next we show that the third item follows from the second. Let H be a $\mathfrak{C}$[quasi-$s(n)$]-succinct HSG against $\mathfrak{D}$. Assume for the sake of contradiction that $\mathfrak{C}$-MCSP[quasi-$s(n)$] is zero-error average-case solvable with success probability $1/N^{O(1)}$ in $\mathfrak{D}$, where $N = 2^n$. This implies that there is a subset A of co-$\mathfrak{C}$-MCSP[quasi-$s(n)$] with density $1/N^{O(1)}$. For each large N, A_N has no strings of $\mathfrak{C}$-circuit complexity $\leq$ quasi-$s(n)$, hence A_N does not intersect the range of H_N for such N. Yet the density of A_n is at least $1/N^{O(1)}$, which contradicts the assumption that H is a HSG against $\mathfrak{D}$.

Finally we show that the first item follows from the third. Suppose $\mathfrak{C}$-MCSP[quasi-$s(n)$] is zero-error average-case hard against $\mathfrak{D}$. Consider the set of truth tables of functions with $\mathfrak{C}$-complexity at most quasi-$s(n)$. We show that this is a collection of $\mathfrak{C}$[quasi-$s(n)$]-succinct

hitting sets against $\mathfrak{D}$. The succinctness condition follows from the definition of the set. To show that this is a collection of hitting sets, suppose to the contrary that there is a set $A \in \mathfrak{D}$ such that A_N has density $1/N^{O(1)}$ for each N a power of 2, and A does not intersect the collection. Then we can define a zero-error average-case algorithm which outputs 0 for $x \in A$ and "?" otherwise. This algorithm only outputs 0 on truth tables that do not have quasi-$s(n)$ size $\mathfrak{C}$-circuits, hence it always outputs the correct answer when it does not output "?". By the density condition on A, the algorithm has success probability $1/N^{O(1)}$. ◀

3.2 From Zero-Error Average-Case Hardness to Succinct Pseudorandomness

In this section, we show that zero-error average-case hardness for MCSP in fact yields succinct *pseudorandom* distributions, where the complexity parameter of the succinctness is slightly worse than that of the MCSP problem we assume to be hard.

In fact, we show something more general for languages $Q, Q' \subseteq \{0,1\}^*$: assuming the existence of certain "semantic" samplers, zero-error average-case hardness over the uniform distribution for a problem Q implies the existence of pseudorandom distributions supported on YES instances of Q'. The main idea for showing the implication to pseudorandomness is to analyze a family of *Sampler-Distinguisher* zero-sum games, which we introduce in this work. This is inspired by, but different from, the PRF-Distinguisher game analyzed in [41]. The strategies of the row player in the Sampler-Distinguisher game are YES instances of a given length for the problem Q we wish to solve. The strategies of the column player are circuits from some circuit class $\mathfrak{D}$, corresponding to the class against which we are analyzing average-case hardness.

The payoff corresponding to a row (instance) x and column (circuit) D is defined as the average of $D(x_j), j = 1 \dots k$ minus the expectation of D on inputs of length $|x_j|$. Here the strings x_j are generated from x to have the following properties. First, when x is a YES instance of Q, the x_j's are YES instances of Q'. Second, when y is random, the set $\{y_i\}$ is a good sampler with high probability, meaning that it can be used to estimate the expectation of any bounded function to within reasonable error.

We are able to argue that when the Row player wins the game, there are succinct pseudorandom distributions against $\mathfrak{D}$, and when the column player wins the game, there is a zero-error average case algorithm for Q. The argument in the second case relies on the approximately optimal succinct strategies for zero-sum games given by [9, 38].

▶ **Lemma 11** ([9, 38]). *Let M be a $r \times c$ matrix with entries in $[-1,1]$ representing the payoffs of a zero-sum game. Let $v(M)$ denote the value of the game. Let $\delta < 1$ be a parameter. Then there is a strategy for the row (Min) player supported uniformly on at most $10\log(c)/\delta^2$ pure strategies that guarantees her a payoff at most $v(M) + \delta$ and a strategy for the column (Max) player supported uniformly on at most $10\log(r)/\delta^2$ pure strategies that guarantees here a payoff at least $v(M) - \delta$.*

We will need a special case of the standard Hoeffding inequalities.

▶ **Proposition 12** ([27]). *Let $X_1 \dots X_n$ be independent random variables taking values in $[-1,1]$. Let $\bar{X} \stackrel{\text{def}}{=} \sum_i X_i/n$ denote the empirical mean of these variables. Then, for any $t > 0$, $\Pr(|\bar{X} - E(\bar{X})| \geq t) \leq 2e^{-t^2 n/2}$.*

We now define the notion of semantic sampler we require.

▶ **Definition 13.** *Let $Q, Q' \subseteq \{0,1\}^*$ be languages, $\ell, m : \mathbb{N} \to \mathbb{N}$ and $\epsilon, \delta : \mathbb{N} \to [0,1]$ be functions. A poly-time computable sequence of functions $f = \{f_N : \{0,1\}^N \times \{0,1\}^{\ell(N)} \to \{0,1\}^{m(N)}\}$ is a (Q, Q')-semantic sampler f with seed length ℓ, output length m, accuracy ϵ and error δ if:*

1. *(Semantic condition) For large enough $N \in \mathbb{N}$, for all $x \in \{0,1\}^N$ and $y \in \{0,1\}^{\ell(N)}$, $x \in Q$ implies $f_N(x,y) \in Q'$.*
2. *(Sampling condition) For large enough $N \in \mathbb{N}$, for every function $g : \{0,1\}^{m(N)} \to [-1,1]$, for all but a $\delta(N)$ fraction of N-bit strings x, $|E_{z \in U_{m(N)}} g(z) - E_{y \in U_{\ell(N)}} g(f_N(x,y))| \leq \epsilon(N)$.*

We are ready to prove our general connection between zero-error average-case hardness and pseudorandomness.

▶ **Theorem 14.** *Let $Q, Q' \subseteq \{0,1\}^*$ be languages, $m : \mathbb{N} \to \mathbb{N}$ be a surjective function such that $m(N) \leq N$ for all $N \in \mathbb{N}$, and $\epsilon, \delta : \mathbb{N} \to [0,1]$ be functions. Suppose there is a (Q, Q')-semantic sampler f with output length m, accuracy ϵ and error δ. If Q is zero-error average-case infeasible for* SIZE[poly] *with success probability $1 - \delta(N)$, then there are pseudorandom distributions with error $30\epsilon(N)$ against* SIZE[poly] *that are supported on Q'.*

Proof. Let $t : \mathbb{N} \to \mathbb{N}$ be a size function, and f be the (Q, Q')-semantic sampler in the statement of the theorem. We define the (f,t) Sampler-Distinguisher family of zero-sum games at level N by their payoff functions as follows. For each positive integer N, the set A_N corresponding to strategies of the Row (Min) player is defined to be the set of N-bit strings x such that $x \in Q$. The set B_N corresponding to strategies of the Column (Max) player is defined to be the set of Boolean circuits with input length $m(N)$ and size at most $t(N)$.

Now we define the Column player's payoff function $P_N : A_N \times B_N \to [-1,1]$ as follows. Given a string $x \in A_N$ and a circuit $D \in B_N$, $P_N(x,D) = E_{y \in U_{\ell(N)}} D(f_N(x,y)) - E_{z \in U_{m(N)}} D(z)$. Since the class of Boolean circuits is closed under complement, we can assume wlog that the value of the game is non-negative for each N.

Let $\gamma : \mathbb{N} \to [0,1]$ be a monotonically decreasing error parameter to be specified later. There are two cases: either for every polynomially bounded t, for almost all N, the value of the (f,t) Sampler-Distinguisher game at level N is at most $\gamma(N)$, or there exists some polynomially bounded t such that for infinitely many N, the value of the game at level N is greater than $\gamma(N)$.

In the first case, using Lemma 11, for each $t(N)$, for N large enough, there is a $\gamma(N)/2$-approximately optimal strategy of the Row player for the game at level N that is a uniform distribution on a multiset R of strings x of length N in Q.

Now the induced distribution on strings $z = f_N(x,y)$ obtained by picking x uniformly from R and y uniformly from strings of length $\ell(N)$ is a pseudorandom distribution with error at most $3\gamma(N)/2$ against circuits with input length $m(N)$ and size at most $t(N)$, just by linearity of expectation. Then we have that there are pseudorandom distributions with error at most $3\gamma(N)/2$ against SIZE(poly) that are supported on Q', using the fact that $R \subseteq Q$ and that $f(x,y) \in Q'$ whenever $x \in Q$ and $y \in \{0,1\}^{\ell(N)}$. We will eventually choose $\gamma(N) = 20\epsilon(N)$, which yields pseudorandom distributions with error at most $30\epsilon(N)$.

In the second case, let t be polynomially bounded in N and I' be an infinite set of input lengths such that the value of the (f,t) Sampler-Distinguisher game is at least $\gamma(N')$ on each $N' \in I'$. Define the set I of input lengths as follows: $N \in I$ iff $m(N) \in I'$. Since I' is infinite and m is surjective, I is also infinite. We show that there is a sequence of circuits $\{C_N\}$ of polynomial size such that for each input length $N \in I$, C_N solves Q well on average, contradicting the assumption that Q is average-case hard against SIZE[poly].

Since the value of the game is at least $\gamma(N')$ on each $N' \in I'$, we have that for each $N' \in I'$, there is an approximately optimal strategy for the Column player supported uniformly on at most $O(N'/\gamma(N')^2)$ pure strategies that guarantees her a payoff at least $\gamma(N')/2$. Note that each pure strategy is simply a circuit of size $O(t)$. For each $N' \in I'$, define the probabilistic circuit $C'_{N'}$ on input z of length N' as follows. $C'_{N'}$ samples uniformly a circuit D from the set of circuits $S_{N'}$ in the support of the approximate strategy for the Column player, and outputs $D(z)$. $C'_{N'}$ can be implemented straightforwardly to have size at most $O(tN'/\gamma(N')^2)$. Let v denote the value $E_{z \leftarrow U_{N'}, D \leftarrow S_{N'}} D(z)$.

Next we define the circuit C_N. We first construct a probabilistic circuit and then show how to fix the randomness. On input x of length N, the circuit samples y uniformly at random from $\{0,1\}^{\ell(N)}$ and computes $f_N(x,y)$. C_N runs the probabilistic circuit C' independently $100N/(\gamma(N))^2$ times on $f_N(x,y)$ for uniformly and independently chosen $y \in \{0,1\}^{\ell(N)}$. It computes the average v' of the outputs obtained by C' over all these runs, and checks if $v' - v \geq \epsilon/4$. In order to implement this check, the value v (which does not depend on x) is hardcoded into C. If the majority of checks succeed, C_N outputs "?", else it outputs 0.

Let $N \in I$ and $N' = m(N)$. We show that if x of length N is a YES instance of Q, C_N always outputs "?" with probability $> 1 - 2^{-N}$, and for a $1 - o(1)$ fraction of inputs of length N, C_N outputs 0 with probability $> 1 - 2^{-N}$. By fixing the randomness of C_N using Adleman's trick, we get deterministic circuits which *always* output "?" on YES inputs, and output 0 on almost all NO instances. Moreover, they output either "0" or "?" on every NO instance. This implies that for each $N \in I$, C_N is a circuit that solves Q well on average.

We first establish our claim for YES instances. Suppose x of length N is in Q. This implies that for all $y \in \{0,1\}^{\ell(N)}$, $f_N(x,y) \in Q'$ by the semantic condition on the (Q,Q')-semantic sampler f. Hence, for each $y \in \{0,1\}^{\ell(N)}$, $f_N(x,y)$ is a pure strategy for the Row player in the (s'',t) Sampler-Distinguisher game at level N'. Since $N' \in I'$, the game at level N' has value greater than $\gamma(N')$. In particular, this means that the Row player's strategy of playing $f(x, U_{\ell N})$ yields payoff at least $\gamma(N')/2$ in expectation to the Column player when the Column player plays the approximately optimal strategy corresponding to the probabilistic circuit C'. Thus, over the randomness of C' and random choice of $y \in \{0,1\}^{\ell(N)}$, $E[C'(f_N(x,y))] > v + \gamma(N')/2$. Since γ is a monotonically decreasing function, and $N' \leq N$, we have that $E[C'(f_N(x,y))] > v + \gamma(N)/2$. Applying Proposition 12, when C' is simulated $100N/(\gamma(N))^2$ times independently on uniformly chosen $y \in \{0,1\}^{\ell(N)}$ to compute the empirical average v', $v' > v + \gamma(N)/4$ with probability $> 1 - 2^{-N}$. Hence C_N outputs "?" with at least this probability, as claimed.

Next we argue that when $\gamma(N)$ is chosen to be $20\epsilon(N)$, for all but approximately $\delta(N)$ fraction of inputs of length N, C_N outputs 0 with probability $> 1 - 2^{-N}$. Intuitively, this follows from the fact that a large enough randomly chosen set of strings is a good sampler for a function with range $[-1,1]$.

More formally, consider the quantity v defined above as the expectation of $C'(z)$ over uniformly chosen N'-bit z and randomness of the circuit C'. We upper bound the probability, for a uniformly chosen x of length N, that the empirical average v' of the outputs obtained by $100N/(\gamma(N))^2$ independent runs of $C'(f_N(x,y))$ for uniformly chosen y, is greater than $v + \gamma(N)/4$.

By the sampling condition for the (Q,Q')-semantic sampler f, with probability at least $1 - \delta(N)$ over uniformly chosen x of length N, $|E_{y \in U_{\ell(N)}} C'(f_N(x,y)) - v| \leq \epsilon(N)$. Call a string x "good" if this inequality is satisfied. Applying Proposition 12 again, for any x, with probability $> 1 - 2^{-N}$ over the internal randomness of C, $|v' - E_{y \in U_{\ell(N)}} C'(f_N(x,y))| \leq \gamma(N)/5$. In particular, for good x, by the triangle inequality, we have that with probability $> 1 - 2^{-N}$, $|v' - v| < \gamma/5 + \epsilon(N) \leq \gamma(N)/5 + \gamma(N)/20 = \gamma(N)/4$. Thus, for all but a $\delta(N)$ fraction of strings x of length N, C_N outputs 0 with probability $> 1 - 2^{-N}$, as claimed. ◀

We now show how to use Theorem 14 to derive pseudorandom distributions from zero-error average-case hardness for MCSP and SAT, by showing the existence of appropriate semantic samplers. Indeed, in both cases, we will use the simple sampler defined below.

▶ **Definition 15.** *Given a function $q : \mathbb{N} \to \mathbb{N}$ such that $q(N) \leq N$ is a power of two for all N, we define the $q(N)$-projection sampler to be the function sequence $Proj = \{Proj_N : \{0,1\}^N \times \{0,1\}^{\log(q(N))} \to \{0,1\}^{\lfloor N/q(N) \rfloor}$, where for any string $x = x_1 x_2 \ldots x_N$ of length N and a string y of length $\log(q(N))$ interpreted in the standard way as an integer in $[q(N)]$, $Proj_N(x,y) = x_{y\lfloor N/q(N) \rfloor} \ldots x_{(y+1)\lfloor N/q(N) \rfloor - 1}$. In other words, $Proj_N(x,y)$ is the contiguous substring of x of length $\lfloor N/q(N) \rfloor$ beginning at index $y\lfloor N/q(N) \rfloor$.*

We first show how to apply Theorem 14 in the case of MCSP.

▶ **Theorem 16.** *Let $\mathfrak{C}$ be any circuit class closed under projections. Let $s : \mathbb{N} \to \mathbb{N}$ be a size function, and $q : \mathbb{N} \to \mathbb{N}$ be any super-constant poly-time computable function such that $q(N) \leq N$ is a power of two for all N. Let $\epsilon : \mathbb{N} \to \mathbb{R}$ be any poly-time computable monotonically decreasing error function such that $\epsilon(N) = \omega(1/\sqrt{q(N)})$. If $\mathfrak{C}$-MCSP$[s(n)]$ is zero-error average-case infeasible for* SIZE[poly] *with success probability $1 - o(1)$, then there are $\mathfrak{C}[s']$-*succinct *pseudorandom distributions with error $O(\epsilon(N))$ against* SIZE[poly]*, where $s' : \mathbb{N} \to \mathbb{N}$ is any size function such that $s'(n - \log(q(2^n))) \geq s(n)$.*

Proof. Let Q be $\mathfrak{C}$-MCSP$[s(n)]$, and Q' be $\mathfrak{C}$-MCSP$[s'(n)]$, where s' is any size function such that $s'(n - 2\log(q(2^n))) \geq s(n)$, as in the statement of the theorem. We show that the $q(N)$-projection sampler $Proj$ is in fact a (Q, Q')-semantic sampler with error $o(1)$ and accuracy $\epsilon(N)$, and then apply Theorem 14.

First we show the semantic condition. Let x of length N be the truth table of a Boolean function with $\mathfrak{C}$-circuits of size at most $s(n)$, where $n = \log(N)$. We show that for any $y \in \{0,1\}^{\log(q(n)}$, $Proj_N(x,y)$ is the truth table of a Boolean function with $\mathfrak{C}$-circuits of size at most $s'(n)$. Indeed, $Proj_N(x,y)$ is the truth table of a function on $n - \log(q(N))$ bits, obtained from x by fixing the first $\log(q(n))$ input bits. Let C_N be a $\mathfrak{C}$-circuit of size $s(n)$ computing $\mathtt{fn}(x)$. By fixing $\log(q(n))$ input bits of C_N to constants, and using the fact that $\mathfrak{C}$ is closed under projections, we obtain a $\mathfrak{C}$-circuit of size at most $s(n)$ computing $\mathtt{fn}(Proj(x,y))$, which is a function with $n - \log(q(N))$ input bits. Since $s'(n - \log(q(N)) \geq s(n)$, we have that $\mathtt{fn}(Proj(x,y))$ is in $\mathfrak{C}$-MCSP$[s'(n)]$.

Next we establish the sampling condition. Note that for x uniformly chosen in $\{0,1\}^N$, the $q(N)$ strings $Proj_N(x,y), y \in [q(N)]$ are uniformly and independently distributed in $\{0,1\}^{\lfloor N/q(N) \rfloor}$. Applying Proposition 12, we have that with for any function $g : \{0,1\}^{\lfloor N/q(N) \rfloor} \to [-1,1]$, for at least a $1 - o(1)$ fraction of x, $|E_{z \in U_{\lfloor N/q(N) \rfloor}} g(z) - E_{y \in U_{\log(q(N))}} g(Proj_N(x,y))| \leq \epsilon(N)$, where $\epsilon(N)$ is as in the statement of the theorem.

Now applying Theorem 14, the statement of the theorem follows immediately. ◀

Theorem 16 gives a connection from zero-error average-case hardness of MCSP to succinct pseudorandom distributions. It is perhaps surprising that zero-error average-case hardness in this context implies the existence of pseudorandom distributions that approximate any poly-size circuit well with respect to *two-sided error*. This is reminiscent of the situation with complexity-theoretic pseudo-random generators, where the generator is allowed to run in time exponential in the seed length. Known equivalences between worst-case hardness of exponential time and existence of complexity-theoretic PRGs [40, 31] also imply an equivalence between complexity-theoretic PRGs and complexity-theoretic hitting set generators. Such an equivalence is unknown in the cryptographic setting, and is a major motivation for our work. Theorem 16 provides partial unconditional evidence for such an equivalence extending to the

cryptographic setting. Indeed, combining Theorem 16 with Proposition 10 for appropriately chosen parameters, we get the following equivalence between succinct hitting sets and succinct pseudorandom distributions in the medium-error regime.

▶ **Corollary 17.** *Let $\mathfrak{C}$ be a standard circuit class closed under projections. Moreover, let $s(n) \geq n$. For every large n, the following are equivalent:*

1. *For every $\delta < 1/2$, there are $\mathfrak{C}[s(O(n))]$-succinct hitting sets with error $N^{-\delta}$ against* SIZE[poly]
2. *For every $\delta < 1/2$, $\mathfrak{C}$-MCSP$[s(O(n))]$ is zero-error average-case infeasible for* SIZE[poly] *with success probability $N^{-\delta}$.*
3. *For every $\delta < 1/2$, there are $\mathfrak{C}[s(O(n))]$-succinct pseudorandom distributions with error $N^{-\delta}$ against* SIZE[poly]

Proof. The second item follows from the first using the proof idea of Proposition 10. By using a direct implication from the first item to the third item of Proposition 10, we avoid the quasi-linear blow-up in the parameter, and preserve the error to within a neglibible additive term.

The third item follows from the second by using Theorem 16 with $\epsilon(N) = N^{\delta}$ and $k(N) = N^{2\delta+\gamma}$ for some γ such that $2\delta + \gamma < 1$. Note that for this parameter choice of k, $s'(N) = s(O(N))$, as desired.

The first item follows trivially from the third, since pseudorandom sets are also hitting sets. ◀

▶ Remark. For MCSP, the equivalence above can be extended to the setting of negligible error by using samplers more sophisticated than the projection sampler used in the proof of Theorem 16, such as samplers corresponding to the Nisan-Wigderson generator. However, this extension comes at the cost of generality - it does not work for $\mathfrak{C}$-MCSP for arbitrary $\mathfrak{C}$ closed under projections.

Corollary 17 implies Theorem 1 by choosing s appropriately.

We next show how to apply Theorem 14 to SAT, by showing an analogous construction of pseudorandom satisfiable formulas based on Feige's hypothesis [16] that random k-CNF formulas with a linear number of clauses are hard to refute. In our terminology, Feige's hypothesis states that random k-SAT is zero-error average-case hard. Here, as elsewhere in this paper, we consider non-uniform algorithms rather than uniform ones.

We would like to construct a pseudorandom distribution supported on satisfiable formulae based on Feige's hypothesis, similar to the construction of succinct pseudorandom distribution in Theorem 16 based on the zero-error average-case hardness of MCSP. However, we need to be careful in how we encode our formulas. With an arbitrary encoding, it might be the case that satisfiable formulas are easily distinguishable from random strings. This would the case, for example, if any valid encoding of any formula began with a 1. To solve this issue, we specify a natural information-theoretically efficient encoding of formulas as follows.

We consider k-CNF formulas on N variables, where k is a constant and N is a power of 2. We encode a formula ϕ with cN clauses by using $(n+1)kcN$ bits of information. Each clause is encoded by $(n+1)k$ bits, n bits to encode each variable, and 1 bit to encode whether the variable is positive or negative. These blocks of bits are simply concatenated together. Thus every string of $(n+1)kcN$ represents a unique k-CNF in a valid way.

Assuming the encoding of inputs to SAT above, we can show the following result.

▶ **Theorem 18.** *Fix a positive integer k. If for every $c > 0$, k-SAT on cN clauses is zero-error infeasible for* SIZE[poly] *with success probability $\Omega(1)$, then for every $\epsilon > 0$ there are pseudorandom distributions with error $O(\epsilon)$ supported entirely on satisfiable formulas.*

Proof. We provide a sketch, since the proof is quite similar to that of Theorem 16.

Let $\epsilon > 0$ be any constant, and q be a power of two to be chosen large enough as a function of ϵ. Let Q be k-SAT with cN clauses for some c a large enough power of two, encoded as described above, and Q' be k-SAT with cN/q clauses. We show that that the q-projection sampler is a (Q, Q')-semantic sampler with error ϵ and accuracy ϵ.

For the semantic condition, we simply note that any sub-formula of a satisfiable formula is itself satisfiable, and therefore the q-projection sampler maps satisfiable formulas to satisfiable formulas in our encoding, whenever q is a power of two. The sampling condition is easy to check by again using Proposition 12.

Now the theorem follows by applying Theorem 14. ◀

Theorem 18 is a formal statement of Theorem 2.

We use the simple projection sampler in the proof of Theorem 18. We might be able to reduce the error for the pseudorandom sets by using more sophisticated samplers, however it is tricky to ensure that satisfiable formulas remain satisfiable when the sampler is applied.

4 On Universal Succinct PRGs

We first state our main conjecture formally.

▶ **Conjecture 19** (Universality Conjecture). *For every $\epsilon < 1$, there is a universal* $\mathsf{SIZE}(2^{\epsilon n})$-succinct *PRG with non-trivial seed length.*

We need the standard fact that a PRG can be stretched by iteration.

▶ **Lemma 20.** *For each $0 < \epsilon < 1$, if there is a PRG with non-trivial seed length, there is a PRG with seed length N^{ϵ}.*

We also need the construction of succinct PRGs from PRGs due to [20].

▶ **Lemma 21** ([20]). *For each $\epsilon > \delta < 1$, if there is a PRG with seed length N^{δ}, there is a* $\mathsf{SIZE}[2^{\epsilon n}]$*-succinct PRG*

▶ **Theorem 22.** *Under Conjecture 19, the following are equivalent:*

1. *There is a one-way function secure against* $\mathsf{SIZE}(\mathrm{poly})$.
2. *There is an non-uniform one-way function secure against* $\mathsf{SIZE}(\mathrm{poly})$.
3. *There is a* $\mathsf{SIZE}[2^{\delta n}]$*-succinct HSGs secure against* $\mathsf{SIZE}[\mathsf{poly}(N)]$*, for some $0 < \delta < 1$.*
4. *There are* $\mathsf{SIZE}[2^{\varepsilon n}]$*-succinct HSGs secure against* $\mathsf{SIZE}[\mathsf{poly}(N)]$*, for any $0 < \varepsilon < 1$.*
5. *There is a PRG secure against* $\mathsf{SIZE}[\mathsf{poly}(N)]$ *with non-trivial seed length.*
6. $\mathsf{MCSP}[2^{\delta n}]$ *is hard on average against* $\mathsf{SIZE}[\mathsf{poly}(N)]$ *for some $0 < \delta < 1$.*
7. $\mathsf{MCSP}[2^{\varepsilon n}]$ *is hard on average against* $\mathsf{SIZE}[\mathsf{poly}(N)]$ *for every $0 < \varepsilon < 1$.*
8. *There are no* $\mathsf{SIZE}(\mathrm{poly})$*-natural proofs against* $\mathsf{SIZE}(2^{\epsilon n})$ *for any $\epsilon > 0$.*
9. *Polynomial-size circuits cannot be PAC-learned with membership queries over the uniform distribution in polynomial time.*

Proof. We establish the equivalence through a series of implications.

(3) implies (6): Follows from Proposition 10.

(6) implies (5): By setting $k(N)$ to be a sufficiently small power of N in Theorem 16, we get that there are $\mathsf{SIZE}[2^{\delta' n}]$-succinct pseudorandom distributions with error $1/N^{\gamma}$ for some $\gamma > 0$. Using Conjecture 19 we get that there is a PRG G with non-trivial seed length and error $1/N^{\gamma}$. The PRG G yields a weak one-way function, and by using the standard conversion from weak to strong one-way functions [54, 19], and then applying the HILL construction [22], we get a PRG with non-trivial seed length and negligible error.

(5) implies (4): By using Lemma 20 and Lemma 21, we get $\mathsf{SIZE}[2^{\varepsilon n}]$-succinct PRGs for any $\varepsilon > 0$. This trivially implies $\mathsf{SIZE}[2^{\varepsilon n}]$-succinct HSGs for any $\varepsilon > 0$.

(4) implies (7): Again follows from Proposition 10.

(7) implies (3): Trivial.

(7) equivalent to (8): Shown in [24].

(1) equivalent to (5): Shown in [22].

(1) implies (2): Trivial.

(2) implies (8): Given any infinite set I of auxiliary inputs, by applying the constructions of [22] and [20] to auxiliary-input one-way functions, for any $\varepsilon > 0$, we get a $\mathsf{SIZE}[2^{\varepsilon n}]$-succinct distribution which can be distinguished from uniform by SIZE(poly)-natural proofs against $\mathsf{SIZE}(2^{\epsilon n})$. This implies that the auxiliary-input one-way function can be inverted on I, in contradiction to the assumption that for each poly-size adversary, there is some infinite set of inputs on which the function is hard to invert.

(6) equivalent to (8): Shown in [13]. ◀

Theorem 22 can easily be seen to imply all the items in Theorem 3.

The equivalences above give a fairly clean picture of connections between various fundamental notions, modulo Conjecture 19. We now discuss some of the more interesting individual connections in more detail.

The connection between auxiliary-input one-way functions and one-way functions has been an important question in cryptography since the former notion was introduced in [42]. The notion of auxiliary-input one-way functions has played an important role in the study of zero-knowledge [42, 48] and learning [10]. In particular, it follows from the main result of [42] that under Conjecture 19, if there is a language with zero-knowledge proofs that is not in polynomial size, then one-way functions exist[1].

The notion of HSGs has not been much studied in cryptography, and this is perhaps because it is not obvious how to use HSGs in crypto applications. One of the issues is that it is not clear how to stretch a HSG, i.e., increase the gap between seed length and output length. As a consequence of the Conjecture, succinct HSGs are stretchable. It remains unclear whether the same is true for standard HSGs under plausible assumptions.

One of the main questions about MCSP is how robust its complexity is with respect to the size parameter s. Known results about the complexity of the problem are not very sensitive to the size parameter, but there are no known equivalences between the complexity of $\mathsf{MCSP}[s]$ and the complexity of $\mathsf{MCSP}[s']$ for s and s' that are different. As a consequence of the Conjecture, we get such an equivalence in the average-case setting. It would be interesting to try to establish this equivalence unconditionally.

The equivalence we get between hardness of learning and one-way functions (modulo the Conjecture) is the first such equivalence of which we are aware for a natural worst-case notion of learning. It is shown in [11] that hardness of PAC-learning *on average* implies the existence of one-way functions. The question of whether the hardness of PAC-learning (over any distribution, and without membership queries) implies the existence of one-way functions is posed in [10]. It would be nice if we could use the Conjecture to resolve this question.

Perhaps the most interesting connection is the equivalence between the average-case hardness of MCSP over the uniform distribution and the existence of one-way functions. This has potential applications for the construction of a natural universal one-way function [37]. One also wonders if under the Conjecture, there are other natural problems and distributions

[1] Note that we are using security against non-uniform adversaries throughout our work.

such that the average-case hardness of the problem under the distibution is equivalent to the existence of one-way functions. One would like a richer theory of reducibility between average-case problems, and equivalences of this sort might help.

5 MCSP and Circuit Lower Bounds against Weak Classes

Pseudorandomness is intimately connected to circuit lower bounds. Complexity-theoretic PRGs are equivalent to circuit lower bounds for E, and cryptographic PRGs are equivalent to one-way functions and hence imply circuit lower bounds for NP. In this section, we consider "weak" circuit classes, i.e., circuit classes $\mathfrak{C}$ for which there are natural proofs useful against $\mathfrak{C}[\text{poly}]$. We show that for weak circuit classes $\mathfrak{C}$, succinct hitting sets imply lower bounds for P against $\mathfrak{C}[\text{poly}]$. Using the connection between succinct hitting sets and average-case hardness for MCSP, we show that zero-error average-case lower bounds for $\mathsf{MCSP}[\text{poly}]$ against $\mathfrak{C}[\text{poly}]$ imply lower bounds for P against $\mathfrak{C}[\text{poly}]$. This is surprising in that we establish a hardness consequence for P based on a hardness assumption about a problem not believed to be in P. Indeed, we show that our result is inherently non-black box if one-way functions exist.

▶ **Lemma 23.** *Let $\mathfrak{C}$ be a weak circuit class closed under projections. If there is a constant k such that there are $\mathsf{SIZE}(n^k)$-succinct hitting sets against $\mathfrak{C}[\text{poly}]$, then $\mathsf{P} \not\subseteq \mathfrak{C}[\text{poly}]$*

Proof. Let $\mathfrak{C}$ be a weak circuit class closed under projections. Suppose there is a constant k such that there are $\mathsf{SIZE}(n^k)$-succinct hitting sets against $\mathfrak{C}[\text{poly}]$. Using Proposition 9, we have that there is a $\mathsf{SIZE}(n^k)$-succinct HSG U with seed length $\mathsf{quasi}\text{-}n^k$ against $\mathfrak{C}[\text{poly}]$. Now consider the following function $f(x,i)$, where i is of length n and x is of length $\mathsf{quasi}\text{-}n^k$. $f(x,i)$ is defined to be 1 iff the i'th bit of $G(x)$=1. Now, since Boolean circuits are a standard class, given seed x to G and index i of $G(x)$, the i'th bit of $G(x)$ is computable in time $\mathsf{quasi}\text{-}s(n)$, which is $\mathsf{quasi}\text{-}n^k$. Thus $f \in \mathsf{P}$.

Now we use a win-win analysis. If $\mathsf{P} \not\subseteq \mathfrak{C}[\text{poly}]$, we are done. Hence we can assume that $\mathsf{P} \subseteq \mathfrak{C}[\text{poly}]$. This implies that $f \in \mathfrak{C}[n^{k'}]$ for some constant k'. Since $\mathfrak{C}$ is closed under projections, it follows that every string in the range of U is the truth table of a function with $\mathfrak{C}$-circuits of size at most $n^{k'}$.

Since $\mathfrak{C}$ is weak, there are natural proofs useful against $\mathfrak{C}[\text{poly}]$. This implies that there is a set $A \subseteq \mathsf{SIZE}(\text{poly})$ of density $1/2$ such that no string y for which $\mathtt{fn}(y)$ is in $\mathfrak{C}[n^{k'}]$ belongs to A .

Now again using the assumption that $\mathsf{P} \subseteq \mathsf{C}[\text{poly}]$, we have that $\mathsf{SIZEpoly} \subseteq \mathfrak{C}[\text{poly}]$, and hence $A \in \mathfrak{C}[\text{poly}]$. But now A is a set in $\mathfrak{C}[\text{poly}]$ of density $1/2$ which does not intersect the range of U non-trivially, contradicting the assumption that U is a hitting set generator against $\mathfrak{C}[\text{poly}]$. ◀

The smallest complexity class within which weakness of $\mathfrak{C}$ is known to imply a super-polynomial lower bound against $\mathfrak{C}$ is ZPEXP [41]. Lemma 23 shows that if additionally there are succinct hitting sets against $\mathfrak{C}[\text{poly}]$, the function for which we get a lower bound is much more explicit - it is in P.

Next we combine Lemma 23 with Proposition 10 to get a surprising implication.

▶ **Theorem 24.** *Let $\mathfrak{C}$ be a weak circuit class closed under projections. If there is a constant k such that $\mathsf{MCSP}[n^k]$ is zero-error average-case hard against $\mathfrak{C}[\text{poly}]$, then P is not contained in $\mathfrak{C}[\text{poly}]$.*

Proof. if there is a constant k such that $\mathsf{MCSP}[n^k]$ is zero-error average-case hard against $\mathfrak{C}[\mathrm{poly}]$, then by Proposition 10, we get that there are $\mathsf{SIZE}(n^k)$-succinct hitting sets against $\mathfrak{C}[\mathrm{poly}]$. Now applying Lemma 23, we get the desired consequence. ◀

Theorem 24 is essentially a restatement of Theorem 4.

Theorem 24 is a partial converse to the following corollary to Theorem 2 from [41].

▶ **Theorem 25.** *Let $\mathfrak{C}[\mathrm{poly}]$ be any circuit class closed under composition with poly-size AC^0. If there is a language in P that cannot be approximated on $1/2 + 1/\mathrm{poly}(n)$ fraction of inputs by $\mathfrak{C}$-circuits of polynomial size, then $\mathsf{MCSP}[2^{n/2}]$ is zero-error average-case hard against $\mathfrak{C}[\mathrm{poly}]$.*

Note that there are examples of weak circuit classes such as AC^0 and $\mathsf{AC}^0[p]$ which satisfy the condition in Theorem 25.

Theorem 24 is a rare example of a non-black-box reduction between two problems - the reduction does not work when the circuit class against which we are arguing is given access to an oracle. Indeed, under standard crypto assumptions, there is no black-box reduction from $\mathsf{MCSP}[n^k]$ to P, for k chosen large enough.

▶ **Theorem 26.** *Let $\mathfrak{C}$ be any Boolean circuit class which contains the projection functions. If there are one-way functions of exponential hardness, there is a constant k for which there is no black-box reduction from zero-error average-case hardness of $\mathsf{MCSP}[n^k]$ against $\mathfrak{C}[\mathrm{poly}]$ to $\mathsf{P} \not\subseteq \mathfrak{C}[\mathrm{poly}]$.*

Proof. If there were such a black-box reduction, then for each oracle A, $\mathsf{P} \subseteq \mathfrak{C}^A[\mathrm{poly}]$ would imply $\mathsf{MCSP}[n^k]$ is zero-error easy on average for $\mathfrak{C}^A[\mathrm{poly}]$. But now consider an oracle A that is complete for P under projections. The antecedent trivially holds for such an oracle, but if the consequent held, we would have $\mathsf{MCSP}[n^k]$ is zero-error easy on average for $\mathsf{SIZE}[\mathrm{poly}]$ for any k, which by the "natural proofs" argument of Razborov and Rudich [44] would invert any one-way function in sub-exponential time. ◀

6 An approximation to average-case reduction for AveMCSP

Finally, we observe that a certain search-to-decision reduction for a variant of MCSP given in recent work [14] actually yields a non black-box approximation to average-case reduction. We will use the following variant of the Nisan-Wigderson generator [14], for which the output of the generator has small average-case circuit complexity for most seeds when the function on which the generator is based has small average-case circuit complexity.

▶ **Theorem 27** ([40, 14])**.** *There is a fixed constant $d > 1$ such that for any constant $c > d$ there is a sequence of functions $\{G_m : \{0,1\}^{m^c} \times \{0,1\}^{O(\log(m))} \to \{0,1\}^m\}$ computable in polynomial time such that*

1. *Given $y \in \{0,1\}^{m^c}$ and for any $t = \log(m)^{\omega(1)}$, if $\mathtt{fn}(y)$ can be computed correctly on 0.9 fraction of inputs by circuits of size t, then with probability $1 - o(1)$ over choice of the seed $z \in \{0,1\}^{O(\log(m))}$, $\mathtt{fn}(G_m(y,z))$ can be computed correctly on 0.9 fraction of inputs by circuits of size t^2.*
2. *Given $y \in \{0,1\}^{m^c}$, if $\mathtt{fn}(y)$ cannot be computed correctly on 0.9 fraction of inputs by circuits of size m^d, then $G_m(y,\cdot)$ is a PRG with error $1/m$ against $\mathsf{SIZE}(m)$.*

▶ **Theorem 28.** *For any $\delta > 0$ and $k > 0$, there is $\epsilon > 0$ such that if $\mathsf{AveMCSP}[2^{n/2}]$ is zero-error easy on average for circuits of size N^k, then $\mathsf{AveMCSP}[2^{\epsilon n}, 2^{\delta n}]$ has polynomial-size circuits.*

Proof. Suppose there is $k > 0$ such that $\mathsf{AveMCSP}[2^{n/2}]$ is zero-error easy on average for circuits of size N^k. Without loss of generality, this means that there exists a set $A \subseteq \{0,1\}^*$ with circuits of size N^k such that A has density at least 0.01 and A does not contain any strings y for which $\mathtt{fn}(y)$ has circuits of size at most $2^{n/2}$.

We show how to solve $\mathsf{AveMCSP}[2^{\epsilon n}, 2^{\delta n}]$ in polynomial size, for some ϵ fixed during the argument which depends on δ and k. Given an input z of length N, our non-uniform polynomial time algorithm runs as follows. First it finds m such that $m^c = N$, where d is the fixed constant given by Theorem 27, and c is chosen to be $2kd/\delta$ in the construction of Theorem 27. It then computes $G_{m^k}(z,x)$ for each $x \in \{0,1\}^{O(\log(m))}$ in polynomial time, truncating each output string to its first m bits. It calculates the fraction η of these strings that belong to A. If $\eta \geq 0.005$, it rejects, else it accepts. This algorithm can clearly be implemented by circuits of polynomial size.

In the following argument, we use "average-case circuit complexity" to mean the size of the smallest circuit computing the function correctly on a 0.9 fraction of inputs.

We argue that when $\mathtt{fn}(z)$ has average-case circuit complexity at least N^δ, the algorithm rejects, and that when $\mathtt{fn}(z)$ has average-case circuit complexity at most N^ϵ for appropriately chosen ϵ, the algorithm accepts. For the first item, note that when $\mathtt{fn}(z)$ has average-case circuit complexity at least N^δ, by the second part of Theorem 27, $G_{m^k}(z,\cdot)$ is a $1/m^k$-error PRG against circuits of size m^k. Since A has circuits of size m^k and also has density at least 0.01, this means that $G_{m^k}(z,\cdot)$ fools A, and hence the fraction η calculated by the algorithm in this case is at least $0.01 - o(1)$, which is at least 0.005 for large enough N.

Now if $\mathtt{fn}(z)$ has average-case circuit complexity at most N^ϵ, where we are yet to fixe ϵ, we have from the first part of Theorem of Theorem 27, for at least a $1 - o(1)$ fraction of seeds x, $G_{m^k}(z,x)$ has average-case circuit complexity at most $N^{2\epsilon}$. Note that we are truncating the output of the PRG to the first m bits, hence the function represented by the output has length $m = N^{\delta/2kd}$. If ϵ is chosen so that $\epsilon < \delta/8kd$, we have that with probability $1 - o(1)$ over choice of seed x, the truncated output of the PRG has average-case circuit complexity at most $2^{n/2}$. Since A does not contain any strings y for which $\mathtt{fn}(y)$ has circuits of size at most $2^{n/2}$, we have that the fraction η is calculated to be $o(1) < 0.005$ for large enough N. ◀

Theorem 28 is a formal version of Theorem 5.

Note that the approximation to average-case reduction in Theorem 28 has a very unusual feature - the gap in the approximation version depends on the *complexity* of the average-case algorithm. In particular, this reduction is not black-box, meaning that the reduction does not extend to the case where the average-case algorithm uses an oracle.

7 Future Work

The Universality Conjecture opens up several directions for future work. The first is to derive further interesting implications from the Conjecture. For example, is the Learning is Hard assumption of [10] equivalent to the non-existence of natural proofs under the Conjecture?

The second is to establish the connections and equivalences we seek unconditionally, or under weaker forms of the Conjecture. Our unconditional results such as Theorem 1 are a step in this direction.

The third is to develop approaches to proving the Conjecture. A useful step here would be to come up with an explicit candidate for universal succinct PRGs.

A fourth direction is to explore consequences of the Conjecture being false. As mentioned before, the Conjecture holds if one-way functions exist, and therefore the failure of the Conjecture would imply the non-existence of one-way functions. But perhaps even stronger consequences follow from the failure of the Conjecture, lending further support to it?

More generally, there are other pathways to connecting average-case hardness of MCSP to the existence of one-way functions. In this work, we have explored average-case hardness in the zero-error sense. But perhaps it would be easier to use average-case hardness of MCSP in the bounded-error sense to construct one-way functions. One question here is to come up with a natural distribution under which MCSP is hard on average in the bounded-error sense – clearly the uniform distribution is not a valid candidate.

References

1 Adi Akavia, Oded Goldreich, Shafi Goldwasser, and Dana Moshkovitz. On basing one-way functions on NP-hardness. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing, Seattle, WA, USA, May 21-23, 2006*, pages 701–710, 2006.

2 Eric Allender. The Complexity of Complexity. In *Computability and Complexity - Essays Dedicated to Rodney G. Downey on the Occasion of His 60th Birthday*, pages 79–94, 2017.

3 Eric Allender and Bireswar Das. Zero Knowledge and Circuit Minimization. In *Symposium on Mathematical Foundations of Computer Science* (MFCS), pages 25–32, 2014.

4 Eric Allender, Joshua A. Grochow, and Cristopher Moore. Graph Isomorphism and Circuit Size. *CoRR*, abs/1511.08189, 2015. `arXiv:1511.08189`.

5 Eric Allender, Lisa Hellerstein, Paul McCabe, Toniann Pitassi, and Michael E. Saks. Minimizing Disjunctive Normal Form Formulas and AC0 Circuits Given a Truth Table. *SIAM J. Comput.*, 38(1):63–84, 2008.

6 Eric Allender and Shuichi Hirahara. New Insights on the (Non-)Hardness of Circuit Minimization and Related Problems. In *International Symposium on Mathematical Foundations of Computer Science* (MFCS), pages 54:1–54:14, 2017.

7 Eric Allender, Dhiraj Holden, and Valentine Kabanets. The Minimum Oracle Circuit Size Problem. In *International Symposium on Theoretical Aspects of Computer Science* (STACS), pages 21–33, 2015.

8 Eric Allender, Michal Koucký, Detlef Ronneburger, and Sambuddha Roy. The pervasive reach of resource-bounded Kolmogorov complexity in computational complexity theory. *J. Comput. Syst. Sci.*, 77(1):14–40, 2011.

9 Ingo Althofer. On Sparse Approximations to Randomized Strategies and Convex Combinations. *Linear Algebra and its Applications*, 199:339–355, 1994.

10 Benny Applebaum, Boaz Barak, and David Xiao. On Basing Lower-Bounds for Learning on Worst-Case Assumptions. In *Proceedings of 49th Annual IEEE Symposium on Foundations of Computer Science*, pages 211–220, 2008.

11 Avrim Blum, Merrick Furst, Michael Kearns, and Richard Lipton. Cryptographic Primitives Based on Hard Learning Problems. In *Proceedings of 13th Annual International Cryptology Conference*, pages 278–291, 1993.

12 Andrej Bogdanov and Luca Trevisan. On Worst-Case to Average-Case Reductions for NP Problems. *SIAM J. Comput.*, 36(4):1119–1159, 2006.

13 Marco L. Carmosino, Russell Impagliazzo, Valentine Kabanets, and Antonina Kolokolova. Learning Algorithms from Natural Proofs. In *Conference on Computational Complexity* (CCC), pages 10:1–10:24, 2016.

14 Marco L. Carmosino, Russell Impagliazzo, Valentine Kabanets, and Antonina Kolokolova. Agnostic Learning from Tolerant Natural Proofs. In *Approximation, Randomization, and Combinatorial Optimization*, pages 35:1–35:19, 2017.

15 Amit Daniely, Nati Linial, and Shai Shalev-Shwartz. From average case complexity to improper learning complexity. In *Proceedings of 46th Annual Symposium on Theory of Computing*, pages 441–448, 2014.

16 Uriel Feige. Relations between average case complexity and approximation complexity. In *Proceedings on 34th Annual ACM Symposium on Theory of Computing*, pages 534–543, 2002.

17 Joan Feigenbaum and Lance Fortnow. Random-Self-Reducibility of Complete Sets. *SIAM J. Comput.*, 22(5):994–1005, 1993.

18 Michael A. Forbes, Amir Shpilka, and Ben Lee Volk. Succinct hitting sets and barriers to proving algebraic circuits lower bounds. In *Proceedings of the 49th Annual ACM Symposium on Theory of Computing*, pages 653–664, 2017.

19 Oded Goldreich. *The Foundations of Cryptography - Volume 1, Basic Techniques*. Cambridge University Press, 2001.

20 Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986.

21 Joshua A. Grochow, Mrinal Kumar, Michael E. Saks, and Shubhangi Saraf. Towards an algebraic natural proofs barrier via polynomial identity testing. *Electronic Colloquium on Computational Complexity (ECCC)*, 24:9, 2017.

22 Johan Håstad, Russell Impagliazzo, Leonid Levin, and Michael Luby. A Pseudorandom Generator from any One-way Function. *SIAM J. Comput.*, 28(4):1364–1396, 1999. doi:10.1137/S0097539793244708.

23 Shuichi Hirahara. Non-Black-Box Worst-Case to Average-Case Reductions within NP. In *59th IEEE Annual Symposium on Foundations of Computer Science*, pages 247–258, 2018.

24 Shuichi Hirahara and Rahul Santhanam. On the Average-Case Complexity of MCSP and Its Variants. In *Computational Complexity Conference* (CCC), pages 7:1–7:20, 2017.

25 Shuichi Hirahara and Osamu Watanabe. Limits of Minimum Circuit Size Problem as Oracle. In *Conference on Computational Complexity* (CCC), pages 18:1–18:20, 2016.

26 John M. Hitchcock and Aduri Pavan. On the NP-Completeness of the Minimum Circuit Size Problem. In *Conference on Foundation of Software Technology and Theoretical Computer Science* (FSTTCS), pages 236–245, 2015.

27 Wassily Hoeffding. Probability Inequalities for Sums of Bounded Random Variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

28 Russell Impagliazzo. A Personal View of Average-Case Complexity. In *Proceedings of the Tenth Annual Structure in Complexity Theory Conference*, pages 134–147, 1995.

29 Russell Impagliazzo, Valentine Kabanets, and Avi Wigderson. In Search of an Easy Witness: Exponential Time vs. Probabilistic Polynomial Time. In *Proceedings of the 16th Annual IEEE Conference on Computational Complexity*, pages 2–12, 2001.

30 Russell Impagliazzo and Leonid A. Levin. No Better Ways to Generate Hard NP Instances than Picking Uniformly at Random. In *31st Annual Symposium on Foundations of Computer Science*, pages 812–821, 1990.

31 Russell Impagliazzo and Avi Wigderson. P = BPP if E requires exponential circuits: Derandomizing the XOR lemma. In *Proceedings of the 29th Annual ACM Symposium on the Theory of Computing (STOC)*, pages 220–229, 1997.

32 Valentine Kabanets. Easiness Assumptions and Hardness Tests: Trading Time for Zero Error. In *Proceedings of the 15th Annual IEEE Conference on Computational Complexity*, pages 150–157, 2000.

33 Valentine Kabanets and Jin-yi Cai. Circuit minimization problem. In *Symposium on Theory of Computing* (STOC), pages 73–79, 2000.

34 Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography, Second Edition*. CRC Press, 2014.

35 Michael Kearns and Leslie Valiant. Cryptographic Limitations on Learning Boolean Formulae and Finite Automata. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*, pages 433–444, 1989.

36 Leonid Levin. Randomness Conservation Inequalities; Information and Independence in Mathematical Theories. *Information and Control*, 61(1):15–37, 1984.

37 Leonid Levin. The Tale of One-Way Functions. *Problems of Information Transmission*, 39(1):92–103, 2003.

38 Richard J. Lipton and Neal E. Young. Simple strategies for large zero-sum games with applications to complexity theory. In *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing, 23-25 May 1994, Montréal, Québec, Canada*, pages 734–740, 1994.

39 Cody Murray and Ryan Williams. On the (Non) NP-Hardness of Computing Circuit Complexity. In *Conference on Computational Complexity* (CCC), pages 365–380, 2015.

40 Noam Nisan and Avi Wigderson. Hardness vs Randomness. *J. Comput. Syst. Sci.*, 49(2):149–167, 1994.

41 Igor Carboni Oliveira and Rahul Santhanam. Conspiracies Between Learning Algorithms, Circuit Lower Bounds, and Pseudorandomness. In *Computational Complexity Conference* (CCC), pages 18:1–18:49, 2017.

42 Rafail Ostrovsky and Avi Wigderson. One-Way Fuctions are Essential for Non-Trivial Zero-Knowledge. In *Proceedings of Second Israel Symposium on Theory of Computing Systems*, pages 3–17, 1993.

43 Leonard Pitt and Manfred Warmuth. Prediction-Preserving Reducibility. *Journal of Computer and System Sciences*, 41(3):430–467, 1990.

44 Alexander A. Razborov and Steven Rudich. Natural Proofs. *J. Comput. Syst. Sci.*, 55(1):24–35, 1997.

45 Steven Rudich. Super-bits, Demi-bits, and NP/qpoly-natural Proofs. In *Randomization and Approximation Techniques in Computer Science*, pages 85–93, 1997.

46 Boris A. Trakhtenbrot. A Survey of Russian Approaches to Perebor (Brute-Force Searches) Algorithms. *IEEE Annals of the History of Computing*, 6(4):384–400, 1984.

47 Luca Trevisan. Extractors and pseudorandom generators. *J. ACM*, 48(4):860–879, 2001.

48 Salil Vadhan. An Unconditional Study of Computational Zero Knowledge. *SIAM Journal on Computing*, 36(4):1160–1214, 2006.

49 Salil Vadhan. Pseudorandomness. *Foundations and Trends in Theoretical Computer Science*, 7(1-3):1–336, 2012.

50 Salil Vadhan. On Learning vs. Refutation. In *Proceedings of the 30th Conference on Learning Theory*, pages 1835–1848, 2017.

51 Salil Vadhan and Colin Zheng. A Uniform Min-Max Theorem with Applications in Cryptography. In *33rd Annual Cryptology Conference,*, pages 93–110, 2013.

52 Leslie Valiant. A Theory of the Learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

53 R. Ryan Williams. Natural Proofs versus Derandomization. *SIAM J. Comput.*, 45(2):497–529, 2016.

54 Andrew Yao. Theory and Applications of Trapdoor Functions (Extended Abstract). In *Proceedings of 23rd Annual Symposium on Foundations of Computer Science*, pages 80–91, 1982.

Testing Properties of Multiple Distributions with Few Samples

Maryam Aliakbarpour
Massachusetts Institute of Technology, Cambridge, MA 02139, USA
maryama@mit.edu

Sandeep Silwal[1]
Massachusetts Institute of Technology, Cambridge, MA 02139, USA
silwal@mit.edu

Abstract

We propose a new setting for testing properties of distributions while receiving samples from several distributions, but few samples per distribution. Given samples from s distributions, $p_1, p_2, \ldots, p_s$, we design testers for the following problems: (1) Uniformity Testing: Testing whether all the p_i's are uniform or ϵ-far from being uniform in ℓ_1-distance (2) Identity Testing: Testing whether all the p_i's are equal to an explicitly given distribution q or ϵ-far from q in ℓ_1-distance, and (3) Closeness Testing: Testing whether all the p_i's are equal to a distribution q which we have sample access to, or ϵ-far from q in ℓ_1-distance. By assuming an additional natural condition about the source distributions, we provide sample optimal testers for all of these problems.

2012 ACM Subject Classification Mathematics of computing → Hypothesis testing and confidence interval computation

Keywords and phrases Hypothesis Testing, Property Testing, Distribution Testing, Identity Testing, Closeness Testing, Multiple Sources

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.69

Funding *Maryam Aliakbarpour*: MIT-IBM Watson AI Lab (Agreement No. W1771646), NSF grants IIS-1741137, and CCF-1733808.
Sandeep Silwal: NSF Graduate Research Fellowship under Grant No. 4000054330.

Acknowledgements The authors would like to thank Sushruth Reddy, Rikhav Shah, and Greg Valiant for helpful discussions about de Finetti's theorem. The authors would also like to thank Ronitt Rubinfeld for valuable feedback.

1 Introduction

Statistical tests are a crucial tool in scientific endeavors to analyze data: We routinely model data to be a set of samples from an unknown distribution, and use statistical tests to infer or verify the properties of the underlying distribution. While these tests typically operate under the assumption that data points are drawn from a *single* underlying distribution, in applications, usually the data is gathered from multiple sources. Furthermore in many situations, it is the case that the dataset contains only a few data points from each source. For example, an online shop may have the purchase history of thousands of customers while each customer may shop at the store a small number of times. Alternatively, a medical dataset might record the lifestyle behaviors of patients of a particular disease while only having few data points from any specific demographic (such as age).

[1] Corresponding author

On the other hand, data that comes from multiple sources may result in a dataset consisting of a collection of unconnected and unrelated data points. For example, it might not be possible to derive any meaningful conclusions from a dataset that contains the blood pressure of patients with heart diseases, Alzheimer patients, and healthy individuals. However, if there is some consensus among the sources, we may be able to make reasonable inferences based on the data. Therefore, an important question to ask is: how can we mathematically model agreement among the sources such that it is possible to design testers with theoretical guarantees?

In this work, we propose a framework for hypothesis testing, one of the most fundamental problems in statistics, while allowing for the underlying data to be drawn from multiple distributions (sources) and only receiving "few" samples from each distribution. More specifically, we study the following problem: suppose we have s source distributions, $p_1, \ldots, p_s$. We have a distribution q (hypothesis), and we aim to distinguish between the case where all the source distributions are equal to q and the case where all the source distributions are far from q. We propose a *structural condition* in order to model the agreement among the sources to enable us to draw meaningful conclusions.

Our *structural condition* requires all the sources to have the same preference for every element, meaning that for each domain element x, either all the sources assign higher probability than the "speculated" probability, $q(x)$, or all of them assign lower probabilities. However, the sources can go arbitrarily higher or lower than $q(x)$ as long as they stay on the same side of the $q(x)$. For example, suppose one has tried several prize wheels (lottery machines) in a casino. The player spins the wheel and expects to receive one of the prizes uniformly. Given the results of each spin, our goal is to test whether all the machines were fair (i.e., selecting the prize uniformly), or they are far from being fair. In this case, we can naturally assume that if the machines are unfair, the house will assign a lower probability to the expensive prizes, and higher probability to cheap ones. Another example is political affiliation at a local vs. national level. Suppose a political party polls its constituents in a district about their opinion on the most crucial policy and compares it with national polls. It is natural to assume that the policies of national interest will receive the same responses in different districts. On the other hand, if a policy affects the district positively (or negatively), members of the district are more (less) likely to pick them. It is worth noting that if no structural condition is assumed, the problem becomes vacuous even in the simplest cases. The main issue is that two completely different sets of distributions may result in identical set of samples. For example, suppose each p_i is a singleton distribution on a random element $x \in [n]$. If we draw one sample from each distribution, the samples we obtain will be indistinguishable from the samples that are i.i.d. from a uniform distribution over $[n]$. See Section 2.2 for more elaboration.

Given our agreement condition, we consider three different cases for our hypothesis q: (*i*) *Uniformity testing:* q is uniform. (*ii*) *Identity testing (goodness of fit):* q is explicitly known. (*iii*) *Closeness testing (equivalence test):* q is accessible through samples. We require each source distribution to provide *exactly one* sample for uniformity and one sample in expectation for identity and closeness testing. We develop sample optimal testers for all these three problems. In fact, the sample complexity of our testers is exactly equal to the standard versions of these problems when samples are drawn from a single source. These results lead to the belief that our agreement condition provides the same power as the standard setting for designing the testers while operating under a weaker assumption.

Our sample complexity upper bounds are achieved by using variants of testers previously used for distribution testing in the case where samples are drawn from a fixed distribution. The challenge however, lies in analyzing these testers in our more general setting with multiple sources. The sample complexity lower bounds follow directly from the single distribution setting. For a full description of our contributions and approaches, see Section 2.3.

1.1 Necessity of modeling multiple sources

We might hypothesize that data points drawn from different distributions can be thought of as coming from some "average" or "aggregated" distribution. Indeed, we know by de Finetti's theorem that an infinite sequence of exchangeable random variables is actually drawn from a mixture of product distributions. In other words, there is some latent variable such that conditioning on this variable, all the samples are independently drawn from one probability distribution. However in the case that we have finitely many samples (or equivalently finitely many sources), de Finetti type theorems only hold up to some approximation error and in the case where the number of samples is sublinear in the domain size, we give a family of distributions where the sequence of random variables with each sample drawn from a different distribution cannot be seen as a mixture of product distributions. This result implies that modeling data as samples from a single distribution is not sufficient when multiple sources are involved. See Section 2.3.4 for more information.

1.2 Comparison with other models

Studying properties of a collection of distributions has been studied prior to our work in [26, 2, 16]. These papers consider two primary models for sampling a collection of distributions. In the first model, which is called the *query model*, the user can query each distribution and receive a sample from it. In the second model, which is called the *sampling model*, the user does not get to choose the source distribution, but the user receives a pair (i, j) which can be interpreted as a sample from the collection: the first element i indicates that distribution i was selected with a probability proportional to some (known or unknown) weight, and then j is a sample drawn from the i-th distribution.

There are few differences between our model and two models listed above. In these models, there is no limit on the number of samples that can come from a distribution. This is in contrast to our setting where every distribution contributes only one sample in expectation. On the other hand, in these two models, there is no agreement condition imposed between the different distributions, and their goal is to distinguish if all the distributions are equal or their *average distance* from a single distribution is at least ϵ. Considering the average distance essentially turns this problem into testing closeness of a distribution over the domain $[n] \times [s]$ which requires more samples.

While our problems are inherently different, none of the results in the papers cited above solve the problems we consider using a sublinear number of queries. In fact in some regime of the parameters, their algorithms draw $\omega(1)$ samples (even in expectation). In some special case, where the number of samples per distribution is $\Theta(1)$ in expectation, the sample complexity of their algorithm is greatly larger than ours. In particular, suppose we have s distributions over a domain of size n and we draw m samples from them in total. In the query model, the provided algorithms pick a few distributions and draw $O(n^{2/3})$ samples from them which is in contrast to our requirement of one sample per distribution. Moreover for the sampling model, the optimal algorithm needs $m = O(\sqrt{ns}/\epsilon^2 + n^{2/3}s^{1/3}/\epsilon^{4/3})$ samples in total. Roughly speaking, if the number of distributions is asymptotically smaller than n,

i.e., $s = o(n)$, then certainly the number of samples, m, has to be $\omega(s)$ meaning that we need more than $\Theta(1)$ samples per distribution. On the other hand, if the number of distributions, s, is $\Omega(n)$, then the number of samples, m, has to be $\Omega(n/\epsilon^2)$ which is drastically larger than our sample complexity, $O(\sqrt{n}/\epsilon^2 + n^{2/3}/\epsilon^{4/3})$.

In [31, 36], the authors consider a similar setting as our paper. In their setting, they have N distributions over the domain of size two. Each distribution is determined by a parameter which indicates the probability of the first domain element, and the algorithm receives t samples from each distribution. However, these papers consider a very different problem compared to ours as their goal is to optimally learn the histogram of the parameters with approximation error as a function of t.

1.3 Other related work

Distribution property testing is a framework for investigating properties of a distribution(s) upon receiving samples. This framework was first introduced in [21, 6], and it is part of the broader topic of hypothesis testing in statistics [27, 25]. In this framework, we wish to determine if one or more unknown distributions satisfy a certain property or are "far" from satisfying the property. The goal is to obtain an algorithm, or tester, for this task that has the optimal sample complexity. Since its introduction, several properties have been considered. See [30, 9, 20] for a survey of results.

The problems of testing uniformity, identity, and closeness of distributions have first been considered in [22, 6, 5]where it is assumed that samples are always drawn from a fixed distribution. Many subsequent work improved on their results, and eventually testers with optimal sample complexities of $\Theta(\sqrt{n}/\epsilon^2)$ for identity and uniformity testing, and $\Theta(n^{2/3}/\epsilon^{4/3} + \sqrt{n}/\epsilon^2)$ for closeness testing were obtained. See [34, 28, 33, 10, 17, 1, 16, 18, 14, 15, 8, 4]. For a survey of techniques used for these problems, see [9].

1.4 Organization

We start with definitions and preliminaries in Section 2. In Section 3, we study uniformity testing with samples from multiple sources.

In Section 4, we study identity testing with non-identically drawn samples. In Section 5 we study closeness testing with non-identically drawn samples. Finally, we prove Theorem 2 in Appendix 6.

2 Preliminaries

2.1 Notation and Definitions

We use $[n]$ to denote the set $\{1, \cdots, n\}$. We consider discrete distributions over $[n]$, which are non-negative functions $p : [n] \to [0, 1]$ such that $\sum_{i\in[n]} p(i) = 1$. We let $p(i)$ denote the probability assigned to element $i \in [n]$ by a distribution p and for a set $A \subseteq [n]$, we define $p(A) = \sum_{i\in A} p(i)$. For $q \geq 1$, the ℓ_q-norm of distribution q is defined as $\|p\|_q = (\sum_{i\in[n]} p(i)^q)^{1/q}$. Given two distributions p and p', the ℓ_q-distance between them is defined as the ℓ_q-norm of the vector of their differences: $\|p - p'\|_q = (\sum_{i\in[n]} |p(i) - q(i)|^q)^{1/q}$. The total variation distance of two distributions p and p' is defined as $\|p - p'\|_{TV} = \sup_A |p(A) - p'(A)|$ which is known to be equal to $\|p - p'\|_1/2$. We say that two distributions p and p' are ϵ-far in ℓ_q-distance if $\|p - p'\|_q \geq \epsilon$. Otherwise, we say that p and p' are ϵ-close in ℓ_q-distance. In this paper, we primarily focus on ℓ_1-distance. We denote the uniform distribution over $[n]$ by $\mathcal{U}_n$. Also, we refer to a Poisson random variable with parameter λ as $\text{Poi}(\lambda)$.

2.2 The Structural Condition

We introduce the structural condition used in our multiple source distribution testing setting. This condition models the assumption that the different sources have an *agreement* of the preferences which we explain earlier.

▶ **Definition 1** (Structural Condition). *Given a sequence of distributions $p_1, p_2, \cdots$ over $[n]$ and another distribution q over $[n]$, we say that $\{p_i\}_{i\geq 1}$ satisfy the structural condition if there exist sets $A \subset [n], B = [n] \setminus A$, such that for all the p_i's,*

$$p_i(j) \geq q(j) \qquad \forall j \in A\,,$$
$$p_i(j) \leq q(j) \qquad \forall j \in B.$$

Note that we ***do not*** *assume knowledge of what the sets A and B are, just that they exist.*

2.2.1 Alternative agreement conditions

To motivate Definition 1, our *structural condition*, we focus on the problem of uniformity testing. In the usual setting of uniformity testing, we are given sample access to a *single* unknown probability distribution p over $[n]$, and we wish to determine if p is equal to $\mathcal{U}_n$ or if $\|p - \mathcal{U}_n\|_1 \geq \epsilon$.

The most general relaxation of the single source assumption is to allow each sample to be drawn from a possibly different distribution. In particular, we wish to distinguish the completeness case, where each sample is i.i.d. from $\mathcal{U}_n$, from the soundness case, where sample i is drawn independently from p_i, and p_i and p_j are not necessarily the same for $i \neq j$, and $\|p_i - \mathcal{U}_n\|_1 \geq \epsilon$ for all i. By using the relation between the ℓ_1-norm and the total variation distance, this general setting can be written in the following way in the soundness case which we require the total variation distance between every p_i and the uniform distribution to be at least $\epsilon/2$:

$$\min_i \max_{A\subseteq[n]} |p_i(A) - \mathcal{U}_n(A)| \geq \epsilon/2. \tag{1}$$

However, we cannot hope to drive meaningful conclusions in this setting. Consider the case where each p_i is a singleton distribution on a random element $x \in [n]$. If we draw one sample from each distribution, the samples we obtain will be indistinguishable from the samples that are i.i.d. from a uniform distribution over $[n]$. A natural strengthening of (1) is to assume that in the soundness case, not only each distribution is different from $\mathcal{U}_n$ on some set A, as we had above, but all the p_i's are far from $\mathcal{U}_n$ on the *same* set A. This can be written as:

$$\max_{A\subseteq[n]} \min_i |p_i(A) - \mathcal{U}_n(A)| \geq \epsilon/2. \tag{2}$$

(Note that the min and max are switched from Equation (1)). In other words, there is some fixed set A such that p_i and $\mathcal{U}_n$ are assigning very different probability mass to the set A. However, this assumption is still too weak to support uniformity testing in sublinear time. The main reason is that we can come up with s distribution satisfying Equation (2), but the samples drawn from them look the same as uniform distribution. In general, for testing a symmetric property (i.e., a property that does not depend on the labeling of the elements), e.g., uniformity, we only consider the number of repetition in the sample set. The main sources of information is how many elements repeated t many times in the sample set. In the single distribution setting, these information is related to the moments of the underlying distribution, and it is known that distributions with the similar moments requires a lot of samples to tell them apart [29, 35, 37].

Consider the following example where we have $s < n$ distributions, and each distribution p_i is supported on $[1, i] \subset [n]$. For $i \in [s]$ The distribution p_i assigns the following probability to the domain element $x \in [n]$.

$$p_i(x) = \begin{cases} \frac{1}{n} & \text{if } x < i \\ 1 - \frac{i-1}{n} & \text{if } x = i \\ 0 & \text{if } x > i \end{cases}.$$

Let A be the set of elements that all the p_i's assign zero probability to them: $\{s+1, s+2, \ldots, n\}$. Clearly in our example Equation (2) holds for a parameter $\epsilon < 1$. As long as $s \leq (1-\epsilon)n$ since $\|p_i - \mathcal{U}_n\|_1/2 \geq |p_i(A) - \mathcal{U}_n(A)| \geq (n-s)/n \geq \epsilon$ for all i. Now, the probability that samples i and j, where $i < j$, are equal is

$$\frac{i-1}{n^2} + \left(1 - \frac{i-1}{n}\right)\frac{1}{n} = \frac{1}{n}$$

which is exactly the probability of a collision between two different samples in the completeness case. Furthermore, for any $k \leq s \leq (1-\epsilon)n$, we can compute the probability that any k samples $i_1 < \cdots < i_k$ match. Due to the support of p_{i_1}, we know that this quantity is precisely

$$\frac{i_1 - 1}{n^k} + \left(1 - \frac{i_1 - 1}{n}\right)\frac{1}{n^{k-1}} = \frac{1}{n^{k-1}}$$

which is exactly the probability that any k samples all match if all samples are drawn from the uniform distribution. Therefore with some generalized notion of the moments, the set of distributions in the above example match the first $O(n)$ moments of the uniform distribution. Due to the matching of these moments, we cannot hope to test uniformity (or any other symmetric property). Hence, a stronger structural condition than (2) is needed to allow testing in our setting. In this work, we proposed a natural strengthening of the assumption (2), given in Definition 1, which is enough to perform uniformity testing, along with other hypothesis testing problems. This is elaborated in Section 2.3.

2.3 Our Contributions

2.3.1 Uniformity testing with multiple sources

In our multiple source distributions setting for uniformity testing, we have s distributions, $p_1, \ldots, p_s$, and each distribution provides *exactly one* sample. Our goal is to distinguish the following cases with probability at least $2/3$[2]:

- **Completeness case:** $p_1, p_2, \cdots$ are all uniform on $[n]$.
- **Soundness case:** $p_1, p_2, \cdots$ are all ϵ-far from uniform on $[n]$ in ℓ_1-distance.

Furthermore, we impose that in the soundness case, the distributions $\{p_i\}_{i=1}^s$ satisfy the *structural condition* given in Definition 1 with q being $\mathcal{U}_n$, the uniform distribution. That is in the soundness case, all the distributions have mass at least $1/n$ on the elements in A and at most $1/n$ on the elements in B for some sets A and B that are **unknown** to us. Note that the structural condition trivially holds in the completeness case when all the p_i's are the same distribution. Therefore, we can think of our setting as a generalization of uniformity testing.

[2] Note that the constant $2/3$ is arbitrary here. One can boost the accuracy to $1 - \delta$ for an arbitrary small δ by increasing the number of samples (distributions) by a $O(\log \delta^{-1})$ factor.

We show that the standard collision-based algorithm used in the single distribution case of uniformity testing ([23, 6, 14]) is able to distinguish the completeness and the soundness case in our multiple sources setting. The statistic that we calculate is the number of pairwise collisions among the samples. We show that in the completeness case, there are "few" collisions among the samples whereas in the soundness case, we see "many" collisions. The main challenge is the analysis of this statistic in the soundness case, since the distributions $p_1, p_2, \cdots$ are not necessarily the same.

In the completeness case that all the p_i's are equal to some distribution p, the collision statistic is equal to a multiple of the ℓ_2-norm of p. We proceed similarly by introducing a more general notion of ℓ_2-norm in our setting. In addition, we argue that our statistic is sufficiently concentrated by calculating its variance. We generalize the tight variance analysis of [14], which shows that the collision based tester is optimal in the single source setting. Again if all the p_i's are equal to some distribution p, as is the case in the single source uniformity testing setting, the variance is related to the ℓ_3-norm of p. In our case where the p_i's are not necessarily the same, we introduce a generalized notion of ℓ_3-norm and relate it to our notion of ℓ_2-norm. This argument relies on Maclaurin's inequality. Altogether, our analysis shows that we can perform uniformity testing in our setting using $O(\sqrt{n}/\epsilon^2)$ samples, which is optimal since the standard single source uniformity testing is a special case of our setting, has a known sample complexity lower bound of $\Omega(\sqrt{n}/\epsilon^2)$ [28]. This result is presented in Section 3.

2.3.2 Identity testing with multiple sources

We now describe identity testing in the multiple source distributions setting. We first assume that we explicitly know some fixed distribution q over $[n]$. We suppose we have s distributions, $p_1, \ldots, p_s$. Our goal then is to distinguish the following cases with probability at least 2/3:

- **Completeness case:** $p_1, p_2, \cdots, p_s$ are identical to q
- **Soundness case:** $p_1, p_2, \cdots, p_s$ are all ϵ-far from q in ℓ_1-distance.

Furthermore, we impose that the distributions $\{p_i\}_{i=1}^s$ and q satisfy the *structural condition* given in Definition 1. For identity testing with multiple sources, we use a generalization of the poissonization method used in many distribution testing problem (see [9]): we assume that we receive Poi(1) samples, as opposed to one sample from each distribution p_i that we had in the uniformity case. Clearly, each distribution provides one sample in expectation, and with high constant probability, no distribution provides more than $O(\log s)$ samples.

In standard single distribution identity testing, a modified version of Pearson's χ^2-test statistic is picked to calculate the expected value of $\|q-p\|_2^2$, where q is the known distribution and all samples are from p [32, 10, 1, 16]. In our case, we generalize this approach and give a new statistic, again a modified version of Pearson's χ^2-test, which calculates a variant of the ℓ_2-distance between our known distribution q and the distributions that our samples come from.

In particular, if we take s samples, we show that the expected value of our statistic is $\left\|\sum_{j=1}^s \vec{\mathbf{e}}_j\right\|_2^2$, where $\vec{\mathbf{e}}_j$ is a vector in $\mathbb{R}^n$ where the x-th entry is $|p_j(x) - q(x)|$ for $x \in [n]$. Note that one can think of this quantity as a generalization of $\|q - p\|_2^2$. We then show that the sample complexity of distinguishing the soundness and completeness cases for our generalized identity testing depends on $\|q\|_2$, the ℓ_2-norm of the known distribution. The main technical issue is to analyze the variance of our statistic which is challenging since each sample can come from a possibly distinct distribution. Finally, we show how to reduce $\|q\|_2$ using a "flattening" scheme adapted from [16] that only enlarges the domain size by a

constant factor which results in the sample complexity of $O(\sqrt{n}/\epsilon^2)$ which is optimal since the standard single distribution uniformity testing is a special case of our generalized identity testing, and it requires $\Omega(\sqrt{n}/\epsilon^2)$ samples [28].

Remark on Goldreich's reduction from identity to uniformity testing. There is a reduction from identity testing to uniformity testing given in [19] in which Goldreich gives mappings F_1 and F_2 such that if p is ϵ-far from q, then $F_2(F_1(p))$ is a distribution that is $O(\epsilon)$-far from the uniform distribution over a larger domain of size $m = n/\gamma$ where γ is a parameter of the reduction. This reduction works partially in our case. Denote $F = F_2 \circ F_1$. Then we can check that $F(p_i)$ is $O(\epsilon)$-far from q for every p_i in the soundness case. Furthermore, if $x \in [n]$ is also in A, then the domain elements corresponding to x in $[m]$ are all at least $1/m$ and similarly, if $x \in [n]$ is in the set B, the domain elements corresponding to x in $[m]$ are all at most $1/m$. In particular, F maps the set $A \subseteq [n]$ to a set $A' \subseteq [m]$ that has the same properties as A and similarly, F maps the set $B \subseteq [n]$ to another subset $B' \subseteq [m]$.

The issue in applying this reduction to our setting is with F_1. In particular, $F_1(p_i)$ increases the domain size from $[n]$ to possibly $[n+1]$, and there is no guarantee if the domain elements in m corresponding to $n+1$ will be in A' or B'. In particular, it could be that for some p_i's, these domain elements will be in A' and for other p_i's, these domain elements can possibly be in B'. This could create potential "cancellations" that hide collisions when observing samples from $F(p_i)$. To fix this, we would have to make sure these domain elements don't have much probability mass, which leads to letting $\gamma = O(\epsilon)$. This ultimately leads to a sub optimal query complexity in terms of ϵ for identity testing. Therefore, we do not pursue this approach.

2.3.3 Closeness testing with multiple sources

We now describe our generalized version of closeness testing. We assume that we have access to two streams of samples. In the first stream, all samples are i.i.d. from some fixed distribution q over $[n]$ that is unknown to us. In the second stream, samples are drawn independently from distributions $p_1, \ldots, p_s$ where p_i and p_j are not necessarily the same distribution for $i \neq j$. Our goal then is to distinguish the following cases with probability at least $2/3$:

- **Completeness case:** $p_1, p_2, \cdots, p_s$ are identical to q
- **Soundness case:** $p_1, p_2, \cdots, p_s$ are all ϵ-far from q in ℓ_1-distance.

We also impose that the distributions $\{p_i\}_{i=1}^s$ and q satisfy the *structural condition* given in Definition 1. Note that the structural condition trivially holds in the completeness case.

Our approach to closeness testing with multiple sources is very similar to our approach for identity testing above. We make use of the poissonization method. In particular, we draw $\mathrm{Poi}(s)$ samples from distribution q. Also, we take $\mathrm{Poi}(1)$ samples from each of the distributions p_i, so in total we have $\mathrm{Poi}(s)$ samples from the distributions $\{p_i\}_{i=1}^s$. Furthermore, we use a (different) modified version of Pearson's χ^2-test proposed in [10, 16] and show that the expected value of our statistic is $\left\|\sum_{j=1}^s \vec{\mathbf{e}}_j\right\|_2^2$ where the vector $\vec{\mathbf{e}}_j$ is the same as in the identity testing case above. With a careful analysis of the statistic, in contrast with [16], we show that the sample complexity only depends on the ℓ_2^2-norm of the q. [3] Finally, we use a (randomized) "flattening" scheme from [16] which results in the sample complexity of $O(n^{2/3}/\epsilon^{4/3} + \sqrt{n}/\epsilon^2)$

[3] Similar analysis has appeared in [3] before this work.

which is optimal since there is a known lower bound of $\Omega(\max(n^{2/3}/\epsilon^{4/3}, \sqrt{n}/\epsilon^2))$ for the single distribution setting of closeness testing [16]. Using the same techniques, we also obtain a tester which uses asymptotically different number of samples from q compared to the number of sources (known as testing with unequal-sized samples). See Remark 18 for more details.

2.3.4 Failure of de Finetti's Theorem with sublinear number of samples

An infinite sequence $X_1, X_2, \ldots$ of random variables is called exchangeable if for all $m \geq 1$, the distribution of the sequence $X_1, \ldots, X_m$ is identical to the distribution of $X_{\sigma(1)}, \ldots, X_{\sigma(m)}$ for any permutation σ on m elements. de Finetti's theorem states that any *infinite* exchangeable sequence is a mixture of product distributions. In other words, there exists a probability measure μ such that conditioned on μ, $X_1, X_2, \ldots$ can be seen as i.i.d. samples from a distribution.

Similarly, a finite sequence $X_1, \ldots, X_m$ is called exchangeable if all the permutations of the sequence have the same distribution. If an exact version of de Finetti's theorem were to hold for finite sequences, our new setting where each sample can come from a different distribution reduces to the known setting where all the samples are i.i.d. (since an algorithm can turn the samples it sees into an exchangeable sequence by randomly permuting the samples). However, all the known finite versions of de Finetti's type theorems have an error term which roughly states that finite exchangeable sequences are only *approximately* close to mixtures of product distributions (see [13, 12, 24]).

In Section 6, we give an example of a finite sequence of random variables that falls in the soundness case of our setting of uniformity testing with multiple sources that is $\Omega(1)$-far from any mixture of product distributions. More precisely, our theorem, Theorem 2, tells us that it is not always possible to approximate a finite exchangeable sequence $X_1, \ldots, X_s$ arbitrarily well by a mixture of product distributions. This suggests that it is not possible to use de Finetti's theorem in our setting and therefore, more refined tools are needed rather than a hammer like de Finetti's theorem. More formally, our theorem is the following.

▶ **Theorem 2.** *Let $s = O(\sqrt{n})$ be the number of samples required by Algorithm 1 for $\epsilon = 1/3$. There exists an exchangeable sequence $X_1, \ldots, X_s$ such that X_i is drawn from distribution q_i which are all supported in $[n]$ and satisfy $\|q_i - \mathcal{U}_n\|_1 \geq 1/3$ for all i. Furthermore, $\{q_i\}_{i=1}^s$ all satisfy the structural condition given in Definition 1 with $q = \mathcal{U}_n$. Let P denote the distribution of the sequence $X_1, \ldots, X_s$. Then P is $\Omega(1)$-far in ℓ_1-distance from any mixture of product distributions.*

The proof of Theorem 2 uses ideas from Diaconis and Freedman in [13]. For other variants and finite extension of de Finetti's theorem, see [24].

3 Uniformity Testing with Multiple Sources

We now present our algorithm, Uniformity-Tester, for uniformity testing with multiple sources. We show the standard collision based statistic, introduced in [22, 7], is a sufficient statistic to distinguish whether all sources are uniform or all sources are ϵ-far from uniform in our multiple sources setting. The collision statistic is selected based on a simple observation: if we draw two samples from a distribution, the probability that these two samples are equal (also known as a *collision*) is lowest when the distribution is uniform. Thus, the number of pairwise collisions tends to be "small" if the samples are drawn from a uniform distribution. We show that this observation still holds in our setting. Our algorithm takes s samples (for

Algorithm 1 UNIFORMITY-TESTER.

Input : n, ϵ, one sample from each of $p_1, p_2, \ldots, p_s$
Output : accept or reject

$s \leftarrow \frac{c_1\sqrt{n}}{\epsilon^2}$
Take s samples $X_1, \cdots, X_s$.
For each $1 \le i < j \le s$, let σ_{ij} be the indicator variable for the event $X_i = X_j$.
$\tau \leftarrow \frac{1+\epsilon^2/16}{n}$
$Z \leftarrow \sum_{i<j} \sigma_{ij} / \binom{s}{2}$
if $Z \ge \tau$ **then**
    Output reject and abort.
Output accept.

a parameter s which we determine later) and calculates the number of pairwise collisions in the samples. Then, it compares the number collisions to a threshold, τ, which we specify later. If the number of collisions is less than τ, we infer the sources are uniform and output accept; otherwise, we infer the sources are far from uniform, and output reject. We present our approach in Algorithm 1 along with the main theorem, Theorem 3, which proves the correctness of our algorithm.

▶ **Theorem 3** (Correctness of UNIFORMITY-TESTER)**.** *There exists a constants c_1 independent of n such that the following statements hold with probability $2/3$:*

- ***Completeness case:*** UNIFORMITY-TESTER *outputs* accept *if each of the s distributions $p_1, \ldots, p_s$ are uniform.*
- ***Soundness Case:*** UNIFORMITY-TESTER *outputs* reject *if the p_i's are ϵ-far from the uniform distribution (i.e., $\|p_i - \mathcal{U}_n\|_1 \ge \epsilon$) and $\{p_i\}_{i=1}^s$ satisfy the structural condition of Definition 1 with $q = \mathcal{U}_n$.*

▶ Remark 4. The sample complexity of Algorithm 1 is optimal due to the lower bound of $\Omega(\sqrt{n}/\epsilon^2)$ for testing uniformity in the standard single source setting presented in [28].

Overview of the proof. To prove the correctness of UNIFORMITY-TESTER, we analyze the statistic Z which is the number of collisions in the sample set:

$$Z = \frac{1}{\binom{s}{2}} \sum_{1 \le i < j \le s} \sigma_{ij}$$

where σ_{ij}, for $i < j$, is the indicator that sample i is equal to sample j. The algorithm outputs accept or reject by comparing the statistic Z to a threshold τ. Our goal is to show Z is below the threshold in the completeness case and above the threshold in the soundness case. To do so, we first compute the expectation of Z and then show a sufficiently strong concentration around its expectation by bounding the variance of Z. By a careful selection of the number of samples and the threshold τ, we can prove with high probability that Z is on the desired side of the threshold, and consequently the correctness of the algorithm.

Proof of Theorem 3. We start by setting the parameters: Let the threshold τ be $(1 + \epsilon^2/16)/n$. Define α to be the solution to $\mathbf{E}[Z] = (1+\alpha)/n$. Let the number of samples, s, to be $c_1\sqrt{n}/\epsilon^2$ for a sufficiently large constant c_1.

Note that in the completeness case, all samples are coming from the uniform distribution. In this case, Z is analyzed in [22, 6, 14], so we know the expected value of the statistic is as follows:

$$\mathbf{E}[Z] = \|\mathcal{U}_n\|_2^2 = \frac{1}{n}\,.$$

Furthermore, the variance of Z is bounded from above as below (see Lemma 2.3 in [14]):

$$\mathbf{Var}[Z] \leq \Theta\left(\frac{s^2 \cdot \|\mathcal{U}_n\|_2^2 + m^3\left(\|\mathcal{U}_n\|_3^3 - \|\mathcal{U}_n\|_2^4\right)}{\binom{s}{2}^2}\right) \leq \Theta\left(\frac{1}{ns^2}\right)\,.$$

Now, by Chebyshev's inequality, we can bound the probability that Z become larger than the threshold as follows

$$\Pr\left[Z \geq \tau\right] \leq \Pr\left[|Z - \mathbf{E}[Z]| \geq \frac{\epsilon^2}{16\,n}\right] \leq \Theta\left(\frac{n}{\epsilon^4\, s^2}\right) \leq \frac{1}{3}$$

where the last inequality holds for a sufficiently large constant c_1 and having $s = c_1\sqrt{n}/\epsilon^2$ which proves the correctness of the completeness case.

The main challenge of this proof is to analyze the soundness case when the p_i's are potentially different. We first give a lower bound for the expected value of Z. We begin by providing an intuitive overview of our approach. In the soundness case, we can compute that the expected value of the indicator random variable for a collision between the i-th and the j-th sample is given by

$$\mathbf{E}[\sigma_{ij}] = \sum_{x\in[n]} p_i(x)p_j(x) \tag{3}$$

where p_i and p_j are the distributions that sample i and j are respectively drawn from. One can think of Equation (3) as a generalization of $\|p\|_2^2$ when two distributions are involved. To bound Equation (3) from below, we make use of the *structural condition*. Namely, we can define the error terms

$$\begin{aligned} e_i(x) &= p_i(x) - \frac{1}{n} \qquad \forall x \in A \\ e_i(x) &= \frac{1}{n} - p_i(x) \qquad \forall x \in B\,. \end{aligned} \tag{4}$$

We know that

$$\sum_{x\in A} e_i(x) = \sum_{x\in B} e_i(x)\,.$$

In fact, the above quantities are half the ℓ_1-distance between p_i and the uniform distribution. We define $e_j(x)$ similarly for p_j, and the above identity similarly holds for the e_j's as well. Using these equations, we show in Lemma 5 that

$$\sum_{x\in[n]} p_i(x)p_j(x) = \frac{1}{n} + \sum_{x\in[n]} e_i(x)e_j(x).$$

Recall that our goal is to show that the expected number of collisions in the soundness case is substantially larger than $\binom{s}{2}/n$. Thus, we desired to bound find a lower bound for the second term in the right hand side above. However, since p_i and p_j are not necessarily the same distribution, it could be the case that for a fixed pair i, j we have $\sum_{x\in[n]} e_i(x)e_j(x) = 0$ which is what we would expect if p_i and p_j were both uniform. Thus, instead of bounding $\sum_{x\in[n]} e_i(x)e_j(x)$ for each pair i and j, we show that the sum of these terms over *all the pairs* $i < j$ is $\Theta(\epsilon^2)$. More formally, we have the following lemma.

▶ **Lemma 5.** *Let $\{p_i\}_{i=1}^s$ be distributions over $[n]$ that are all ϵ-far from $\mathcal{U}_n$ in ℓ_1-distance, and satisfy the structural condition given in Definition 1 with $q = \mathcal{U}_n$. Let X_i be drawn independently from p_i for all $1 \le i \le s$. Let σ_{ij} be the indicator variable for the event $X_i = X_j$ and define $Z = \sum_{i<j} \sigma_{ij} / \binom{s}{2}$. Then the following estimate holds*

$$\mathbf{E}[Z] \ge \frac{1 + \epsilon^2/8}{n}\,.$$

Proof. Recall the error terms which we defined in Equation (4):

$$e_i(x) = p_i(x) - \frac{1}{n} \qquad \forall x \in A\,,$$
$$e_i(x) = \frac{1}{n} - p_i(x) \qquad \forall x \in B\,.$$

We start by giving a convenient representation of $\mathbf{E}[\sigma_{ij}]$ in terms of the error terms:

$$\mathbf{E}[\sigma_{ij}] = \frac{1}{n} + \sum_{x\in[n]} e_i(x)e_j(x)\,. \tag{5}$$

To prove the above equation, observe that since the sum of the probabilities in any discrete distribution is one, we have:

$$\sum_{x\in A} e_i(x) = \sum_{x\in B} e_i(x) \tag{6}$$

and similar for the e_j's. All of the $e_i(x)$'s and the $e_j(x)$'s are non-negative for any domain element x by definition and the *structural condition*. Thus, we can obtain:

$$\begin{aligned}
\mathbf{E}[\sigma_{ij}] &= \sum_{x\in[n]} p_i(x)p_j(x) \\
&= \sum_{x\in A}\left(e_i(x) + \frac{1}{n}\right)\left(e_j(x) + \frac{1}{n}\right) + \sum_{x\in B}\left(\frac{1}{n} - e_i(x)\right)\left(\frac{1}{n} - e_j(x)\right) \\
&= \frac{|A|}{n^2} + \frac{1}{n}\sum_{x\in A} e_i(x) + \frac{1}{n}\sum_{x\in A} e_j(x) + \sum_{x\in A} e_i(x)e_j(x) \\
&+ \frac{|B|}{n^2} - \frac{1}{n}\sum_{x\in B} e_i(x) - \frac{1}{n}\sum_{x\in B} e_j(x) + \sum_{x\in B} e_i(x)e_j(x)\,.
\end{aligned}$$

Using Equation (6), it is clear that the sum of two middle terms above are zero:

$$\frac{1}{n}\sum_{x\in A} e_i(x) - \frac{1}{n}\sum_{x\in B} e_i(x) = 0\,, \quad \text{and} \qquad \frac{1}{n}\sum_{x\in A} e_j(x) - \frac{1}{n}\sum_{x\in B} e_j(x) = 0\,,$$

which implies the desired identity we claimed in Equation (5):

$$\mathbf{E}[\sigma_{ij}] = \frac{|A| + |B|}{n^2} + \sum_{x\in[n]} e_i(x)e_j(x) = \frac{1}{n} + \sum_{x\in[n]} e_i(x)e_j(x)\,.$$

Using this identity for all $\binom{s}{2}$ pair of samples, yields to the following:

$$\mathbf{E}\left[\binom{s}{2} Z\right] = \mathbf{E}\left[\sum_{j<i} \sigma_{ij}\right] = \sum_{j<i}\sum_{x\in[n]} p_i(x)p_j(x) = \binom{s}{2}\frac{1}{n} + \sum_{j<i}\sum_{x\in[n]} e_i(x)e_j(x). \tag{7}$$

Now, we focus on the second term on the right hand side of the equation above. We can compute that

$$\sum_{j<i}\sum_{x\in B} e_i(x)e_j(x) = \frac{1}{2}\left(\underbrace{\sum_{x\in B}\left(\sum_{i=1}^{s} e_i(x)\right)^2}_{\text{first term}} - \underbrace{\sum_{i=1}^{s}\sum_{x\in B} e_i(x)^2}_{\text{second term}}\right). \tag{8}$$

To fine a lower bound $\mathbf{E}[Z]$, we find a lower bound for the first term and an upper bound for the second term in the right hand side above.

Lower bound for the first term. Note that if $x \in B$, by definition, $e_i(x)$ is at most $1/n$. On the other hand, $\sum_{x\in B} e_i(x)$ is half of the ℓ_1-distance between p_i and the uniform distribution. Define ϵ_i' to be $\|p_i - \mathcal{U}_n\|_1 = 2\sum_{x\in B} e_i(x)$. Clearly, ϵ_i' is at least ϵ. Then, we have the following lower bound for the size of B:

$$|B|\cdot\frac{1}{n} \geq \sum_{x\in B} e_i(x) = \frac{\epsilon_i'}{2}, \quad \Rightarrow \quad |B| \geq \frac{\epsilon_i' n}{2} \geq \frac{\epsilon n}{2}.$$

Therefore, it follows that for any i, we have

$$\sum_{x\in B} e_i(x)^2 \leq \frac{1}{n^2}\cdot\frac{\epsilon_i' n}{2} = \frac{\epsilon_i'}{2n}.$$

Now by the Cauchy-Schwarz inequality, and having $|B| \leq n$, we have:

$$\sum_{x\in B}\left(\sum_{i=1}^{s} e_i(x)\right)^2 \geq \frac{1}{|B|}\left(\sum_{i=1}^{s}\sum_{x\in B} e_i(x)\right)^2 \geq \frac{\left(\sum_{i=1}^{s}\epsilon_i'\right)^2}{4\,n}.$$

Upper bound for the second term. On the other hand, for the second term in Equation (8), we obtain:

$$\sum_{i=1}^{s}\sum_{x\in B} e_i(x)^2 \leq \sum_{i=1}^{s}\sum_{x\in B}\frac{e_i(x)}{n} \leq \frac{1}{2n}\sum_{i=1}^{s}\epsilon_i'$$

where the first inequality holds since the $e_i(x)$'s are at most $1/n$.

Putting it all together. Using the two bounds above, we achieve the following lower bound for Equation (8):

$$\sum_{j<i}\sum_{x\in B} e_i(x)e_j(x) \geq \frac{1}{2}\left(\frac{\left(\sum_{i=1}^{s}\epsilon_i'\right)^2}{4n} - \frac{\sum_{i=1}^{s}\epsilon_i'}{2n}\right).$$

Observe that since $s = c_1\sqrt{n}/\epsilon^2$, for a sufficiently large c_1, s is at least $\Theta(1/\epsilon)$. Therefore, we have:

$$\sum_{i=1}^{s}\epsilon_i' \geq s\epsilon \geq 4 \quad \Rightarrow \quad \frac{1}{2}\left(\sum_{i=1}^{s}\epsilon_i'\right)^2 - \sum_{i=1}^{s}\epsilon_i' \geq \frac{\left(\sum_{i=1}^{s}\epsilon_i'\right)^2}{4}.$$

Therefore, we obtain:

$$\sum_{j<i}\sum_{x\in B} e_i(x)e_j(x) \geq \frac{1}{2}\left(\frac{\left(\sum_{i=1}^{s}\epsilon_i'\right)^2}{4n} - \frac{\sum_{i=1}^{s}\epsilon_i'}{2n}\right) \geq \frac{\left(\sum_{i=1}^{s}\epsilon_i'\right)^2}{16n} \geq \frac{s^2\epsilon^2}{16n}.$$

Going back to Equation (7), we achieve:

$$\mathbf{E}\left[\binom{s}{2}Z\right] \geq \binom{s}{2}\frac{1}{n} + \frac{s^2\epsilon^2}{16n} \geq \binom{s}{2}\frac{(1+\epsilon^2/8)}{n},$$

which concludes the proof of the lemma. ◀

We now proceed with the proof of Theorem 3. In the next step, we show a tight bound for the variance of the our statistic Z. We generalize the tight variance analysis given in [14] for the standard collision based tester in the single source setting to our multiple source setting. We start by a useful identity for the variance: $\mathbf{Var}[Z] = \mathbf{E}[Z^2] - \mathbf{E}[Z]^2$. Note that when we expand $Z^2 = (\sum_{i<j}\sigma_{ij})^2$, we get terms of the form $\sigma_{ij}\sigma_{jk}$. In the single distribution case, this term can be related to the ℓ_3-norm of p. In our setting, we introduce a generalization of the ℓ_3-norm which is the following:

$$\mathbf{E}[\sigma_{ij}\sigma_{jk}] = \sum_{x\in[n]} p_i(x)p_j(x)p_k(x). \tag{9}$$

To upper bound Equation (9), we again make use of the *structural condition* and relate it to our version of the ℓ_2-norm, Equation (3), by using Maclaurin's inequality which roughly states that the ℓ_3-norm is at most the ℓ_2-norm. More formally, we have the following lemma and our proof is presented in Section 3.1.

▶ **Lemma 6.** *Let $\{p_i\}_{i=1}^s$ be distributions over $[n]$ that are all ϵ-far from $\mathcal{U}$ in ℓ_1-distance and satisfy the structural condition given in Definition 1 with $q = \mathcal{U}_n$. Let X_i be drawn independently from p_i for all $1 \leq i \leq s$. Let σ_{ij} be the indicator variable for the event $X_i = X_j$. Then the following estimate holds*

$$\mathbf{Var}\left(\sum_{i<j}\sigma_{ij}\right) \leq \frac{18\alpha s}{n^2}\binom{s}{2} + 3\left(\frac{\alpha}{n}\binom{s}{2}\right)^{3/2} + \sum_{i<j}\mathbf{E}[\sigma_{ij}]$$

where α is defined to be the solution to $\mathbf{E}[Z] = (1+\alpha)/n$, and it is at least $\epsilon^2/8$.

We can now prove the correctness of the algorithm by bounding the probability that Z is below the threshold τ. Recall that $\mathbf{E}[Z] = (1+\alpha)/n \geq (1+\epsilon^2/8)/n$ from Lemma 5. Therefore, $\alpha \geq \epsilon^2/8$. By Chebyshev's inequality, we have

$$\begin{aligned}
\Pr[Z < \tau] &= \Pr[\mathbf{E}[Z] - Z \geq \mathbf{E}[Z] - \tau] \leq \Pr[|\mathbf{E}[Z] - Z| \geq \mathbf{E}[Z] - \tau] \\
&\leq \Pr\left[|\mathbf{E}[Z] - Z| \geq \frac{\alpha - \epsilon^2/16}{n}\right] \leq \mathbf{Var}[Z]\cdot\left(\frac{n}{\alpha-\epsilon^2/16}\right)^2 \\
&\leq \mathbf{Var}\left[\sum_{i<j}\sigma_{ij}\right]\cdot\left(\frac{n}{\binom{s}{2}(\alpha-\epsilon^2/16)}\right)^2.
\end{aligned}$$

Now, Lemma 6 gives us

$$\mathbf{Var}\left(\sum_{i<j}\sigma_{ij}\right) \leq \underbrace{\frac{18\alpha s}{n^2}\binom{s}{2} + 3\left(\frac{\alpha}{n}\binom{s}{2}\right)^{3/2}}_{T_1} + \underbrace{\sum_{i<j}\mathbf{E}[\sigma_{ij}]}_{T_2}.$$

We use T_1 and T_2 to indicate the two terms in the upper bound above. In either of the cases $T_1 \leq T_2$ or $T_1 > T_2$, we show the error probability is bounded by $1/3$.

Case 1: $T_1 \leq T_2$. In this case, we bound the variance of the number of collisions by $2\,T_2$. We have:

$$\Pr[Z < \tau] \leq \mathbf{Var}\left[\sum_{i<j}\sigma_{ij}\right]\cdot\left(\frac{n}{\binom{s}{2}(\alpha-\epsilon^2/16)}\right)^2 \leq 2\,T_2\cdot\left(\frac{n}{\binom{s}{2}(\alpha-\epsilon^2/16)}\right)^2$$
$$\leq 2\sum_{i<j}\mathbf{E}[\sigma_{ij}]\cdot\left(\frac{n}{\binom{s}{2}(\alpha-\epsilon^2/16)}\right)^2 \leq \Theta\left(\frac{s^2(1+\alpha)}{n}\cdot\frac{n^2}{s^4(\alpha-\epsilon^2/16)^2}\right)$$
$$\leq \Theta\left(\frac{n}{s^2}\cdot\underbrace{\frac{1+\alpha}{(\alpha-\epsilon^2/16)^2}}_{f(\alpha)}\right).$$

Define $f(\alpha) := (1+\alpha)/(\alpha-\epsilon^2/16)^2$. We can compute that f is a decreasing function over the range $[\epsilon^2/8, \infty)$, so we can bound $f(\alpha)$ by $f(\epsilon^2/8) = \Theta(1/\epsilon^2)$ from above. Thus, we bound the probability of $Z < \tau$ as

$$\Pr[Z < \tau] \leq \Theta\left(\frac{n}{s^2\epsilon^2}\right) \leq \frac{1}{3},$$

where the last inequality holds for a sufficiently large constant c_1 and having $s = c_1\sqrt{n}/\epsilon^2$.

Case 2: $T_1 > T_2$. In this case, we bound the variance of Z by $2\,A$. Note that we know $\alpha \geq \epsilon^2/8$, so we have:

$$\Pr[Z < \tau] \leq \mathbf{Var}\left[\sum_{i<j}\sigma_{ij}\right]\cdot\left(\frac{n}{\binom{s}{2}(\alpha-\epsilon^2/16)}\right)^2 \leq 2\,T_2\cdot\left(\frac{n}{\binom{s}{2}(\alpha-\epsilon^2/16)}\right)^2$$
$$\leq 2\left(\frac{18\alpha s}{n^2}\binom{s}{2} + 3\left(\frac{\alpha}{n}\binom{s}{2}\right)^{3/2}\right)\cdot\left(\frac{n}{\binom{s}{2}(\alpha-\epsilon^2/16)}\right)^2$$
$$\leq \Theta\left(\left(\frac{\alpha s^3}{n^2} + \frac{\alpha^{3/2}s^3}{n^{3/2}}\right)\cdot\frac{n^2}{s^4\alpha^2}\right) \leq \Theta\left(\frac{1}{s\,\alpha} + \frac{\sqrt{n}}{s\sqrt{\alpha}}\right).$$

The number of samples, s is chosen to be

$$s = c_1\cdot\frac{\sqrt{n}}{\epsilon^2} \geq \Theta\left(\frac{1}{\epsilon^2} + \frac{\sqrt{n}}{\epsilon}\right) \geq \Theta\left(\frac{1}{\alpha} + \frac{\sqrt{n}}{\sqrt{\alpha}}\right),$$

and therefore, by picking a sufficiently large constant c_1, we can bound the probability of outputting the incorrect answer in the soundness case by $1/3$. ◀

3.1 Proof of Lemma 6

▶ **Lemma 6.** *Let $\{p_i\}_{i=1}^s$ be distributions over $[n]$ that are all ϵ-far from $\mathcal{U}$ in ℓ_1-distance and satisfy the structural condition given in Definition 1 with $q = \mathcal{U}_n$. Let X_i be drawn independently from p_i for all $1 \leq i \leq s$. Let σ_{ij} be the indicator variable for the event $X_i = X_j$. Then the following estimate holds*

$$\mathbf{Var}\left(\sum_{i<j}\sigma_{ij}\right) \leq \frac{18\alpha s}{n^2}\binom{s}{2} + 3\left(\frac{\alpha}{n}\binom{s}{2}\right)^{3/2} + \sum_{i<j}\mathbf{E}[\sigma_{ij}]$$

where α is defined to be the solution to $\mathbf{E}[Z] = (1+\alpha)/n$, and it is at least $\epsilon^2/8$.

Proof. For simplicity, let W denote $\sum_{i<j} \sigma_{ij}$. We bound the variance of W from above in the following steps.

$$\begin{aligned}
&\mathbf{Var}[W] = \mathbf{E}[W^2] - \mathbf{E}[W]^2 = \mathbf{E}\left[\left(\sum_{i<j} \sigma_{ij}\right)^2\right] - \left(\sum_{i<j} \mathbf{E}[\sigma_{ij}]\right)^2 \\
&= \mathbf{E}\left[\sum_{\substack{i<j,k<\ell \\ \text{all distinct}}} \sigma_{ij}\sigma_{k\ell} + 2\sum_{i<j<\ell} (\sigma_{ij}\sigma_{ik} + \sigma_{ij}\sigma_{jk} + \sigma_{ik}\sigma_{jk}) + \sum_{i<j} \sigma_{ij}^2\right] \\
&- \sum_{\substack{i<j,k<\ell \\ \text{all distinct}}} \mathbf{E}[\sigma_{ij}]\,\mathbf{E}[\sigma_{k\ell}] - 2\sum_{i<j<\ell} (\mathbf{E}[\sigma_{ij}]\,\mathbf{E}[\sigma_{ik}] + \mathbf{E}[\sigma_{ij}]\,\mathbf{E}[\sigma_{jk}] + \mathbf{E}[\sigma_{ik}]\,\mathbf{E}[\sigma_{jk}]) \\
&- \sum_{i<j} \mathbf{E}[\sigma_{ij}]^2 .
\end{aligned}$$

Note that if i, j, k, and ℓ are all distinct, then σ_{ij} is independent from $\sigma_{k\ell}$. Thus, we have:

$$\mathbf{E}[\sigma_{ij}\sigma_{k\ell}] = \mathbf{E}[\sigma_{ij}]\,\mathbf{E}[\sigma_{k\ell}] .$$

Moreover, we know that $\mathbf{E}[\sigma_{ij}] \geq 1/n$ for all $i < j$ from Lemma 5. Having $\sigma_{ij}^2 = \sigma_{ij}$, we continue bounding the variance as follows:

$$\begin{aligned}
\mathbf{Var}[W] &= 2\sum_{i<j<\ell} (\mathbf{E}[\sigma_{ij}\sigma_{ik}] + \mathbf{E}[\sigma_{ij}\sigma_{jk}] + \mathbf{E}[\sigma_{ik}\sigma_{jk}]) \\
&\quad + \mathbf{E}\left[\sum_{i<j} \sigma_{ij}\right] - \binom{s}{3}\frac{6}{n^2} - \sum_{i<j} \mathbf{E}[\sigma_{ij}]^2 .
\end{aligned}$$

For now, we focus on the first sum in the right hand side above. We bound this term via the error terms we defined in Equation (4). We note that

$$\begin{aligned}
&\mathbf{E}[\sigma_{ij}\sigma_{ik}] = \mathbf{E}[\sigma_{ij}\sigma_{jk}] = \mathbf{E}[\sigma_{ik}\sigma_{jk}] = \sum_{x\in[n]} p_i(x)p_j(x)p_k(x) \\
&= \sum_{x\in A}\left(\frac{1}{n} + e_i(x)\right)\left(\frac{1}{n} + e_j(x)\right)\left(\frac{1}{n} + e_k(x)\right) \\
&+ \sum_{x\in B}\left(\frac{1}{n} - e_i(x)\right)\left(\frac{1}{n} - e_j(x)\right)\left(\frac{1}{n} - e_k(x)\right) \\
&\leq \frac{1}{n^2}\left(\underbrace{\sum_{x\in A} e_i(x) - \sum_{x\in B} e_i(x)}_{=0} + \underbrace{\sum_{x\in A} e_j(x) - \sum_{x\in B} e_j(x)}_{=0} + \underbrace{\sum_{x\in A} e_k(x) - \sum_{x\in B} e_k(x)}_{=0}\right) \\
&+ \frac{1}{n}\left(\sum_{x\in[n]} e_i(x)e_j(x) + e_i(x)e_k(x) + e_j(x)e_k(x)\right) + \sum_{x\in[n]} e_i(x)e_j(x)e_k(x) + \sum_{x\in[n]} \frac{1}{n^3}
\end{aligned}$$

where the last inequality holds since all the $e_i(x)$'s are non-negative. Therefore, we can

continue bounding the variance as follows:

$$\mathbf{Var}[W] \leq \binom{s}{3}\frac{6}{n^2} + \frac{18\,s}{n}\sum_{i<j}\sum_{x\in[n]} e_i(x)e_j(x) + 6\sum_{i<j<k}\sum_{x\in[n]} e_i(x)e_j(x)e_k(x) + \mathbf{E}[W] - \binom{s}{3}\frac{6}{n^2}.$$

To bound the above terms, we use Maclaurin's inequality proved in [11].

▶ **Lemma 7** (Maclaurin's inequality). *Let $\{a_i\}_{i=1}^s$ be non-negative real numbers. Define*

$$S_k = \frac{1}{\binom{s}{k}}\sum_{1\leq i_1<\cdots<i_k\leq s} a_{i_1}a_{i_2}\cdots a_{i_k}.$$

Then,

$$S_1 \geq \sqrt{S_2} \geq \sqrt[3]{S_3} \geq \cdots \geq \sqrt[s]{S_s}.$$

For our purposes, we prove a strengthening of Maclaurin's inequality which is given in the following lemma:

▶ **Lemma 8** (Strengthened Maclaurin's Inequality).

$$\left(2\sum_{i<j<k}\sum_{x\in[n]} e_i(x)e_j(x)e_k(x)\right)^2 \leq \left(\sum_{i<j}\sum_{x\in[n]} e_i(x)e_j(x)\right)^3.$$

Proof. We prove this by induction on n. Consider the case $n = 1$. By Lemma 7, we have:

$$\left(\sum_{i<j} e_i(1)e_j(1)\right)^3 \geq \frac{\binom{s}{2}^3}{\binom{s}{3}^2}\left(\sum_{i<j<k} e_i(1)e_j(1)e_k(1)\right)^2.$$

Now for $s \geq 3$, we have: $\binom{s}{2}^3/\binom{s}{3}^2 > 4$ which proves our claim. We now proceed by induction. Suppose that the induction hypothesis is true for $n-1$. We know by the induction hypothesis that the following two inequalities hold:

$$\left(\underbrace{2\sum_{i<j<k}\sum_{x\in[n-1]} e_i(x)e_j(x)e_k(x)}_{F}\right)^2 \leq \left(\underbrace{\sum_{i<j}\sum_{x\in[n-1]} e_i(x)e_j(x)}_{F'}\right)^3$$

$$\left(\underbrace{2\sum_{i<j<k} e_i(n)e_j(n)e_k(n)}_{G}\right)^2 \leq \left(\underbrace{\sum_{i<j} e_i(n)e_j(n)}_{G'}\right)^3$$

where the first inequality is the induction hypothesis and the second inequality is just the base case of the induction which was proved earlier. Let F, F', G, and G' denote the terms as indicated above. The above inequalities after substituting new variables become: $F'^3 \geq F^2$ and $G'^3 \geq G^2$. Since all of these terms are positive, we have:

$$(F'^3G'^3)^{1/2} \geq FG$$

Then, by the arithmetic mean-geometric mean inequality, we have

$$3F'^2G' + 3F'G'^2 \geq 6(F'G')^{3/2} \geq 6FG \geq 2FG.$$

Using the fact that $F'^3 \geq F^2$ and $G'^3 \geq G^2$ again, yields

$$(F' + G')^3 \geq (F + G)^2\,,$$

which concludes the lemma. ◀

We now proceed to bound the variance of W. We know

$$\mathbf{Var}[W] \leq \frac{18\,s}{n} \sum_{i<j} \sum_{x\in[n]} e_i(x)e_j(x) + 6 \sum_{i<j<k} \sum_{x\in[n]} e_i(x)e_j(x)e_k(x) + \mathbf{E}[W]\,.$$

In the next step, we bound the two middle term based on n, s, and α. Using Lemma 8, we have

$$\sum_{i<j<k} \sum_{x\in[n]} e_i(x)e_j(x)e_k(x) \leq \frac{1}{2}\left(\sum_{i<j} \sum_{x\in[n]} e_i(x)e_j(x)\right)^{3/2}.$$

Recall that $\mathbf{E}[Z] = (1+\alpha)/n$. Thus, using Equation (5), we know

$$\binom{s}{2}\frac{1+\alpha}{n} = \mathbf{E}[W] = \sum_{i<j} \sum_{x\in[n]} p_i(x)p_j(x) = \sum_{i<j} \sum_{x\in[n]} \left(e_i(x)e_j(x) + \frac{1}{n}\right),$$

which immediately implies that

$$\sum_{i<j} \sum_{x\in[n]} e_i(x)e_j(x) = \binom{s}{2}\frac{\alpha}{n}.$$

Putting all of it together, we obtain

$$\mathbf{Var}[W] \leq \frac{18\,\alpha\,s}{n^2}\binom{s}{2} + 3\left(\binom{s}{2}\frac{\alpha}{n}\right)^{3/2} + \sum_{i<j} \mathbf{E}[\sigma_{ij}]\,,$$

as desired. ◀

4 Identity Testing with Multiple Sources

In this section, we present our algorithm for identity testing with multiple sources and its analysis. Recall that our goal is to distinguish the following two cases with probability at least 2/3 given knowledge of some fixed distribution q over $[n]$:

- **Completeness case:** $p_1, p_2, \cdots, p_s$ are identical to q
- **Soundness case:** $p_1, p_2, \cdots, p_s$ are all ϵ-far from q in ℓ_1-distance

where we receive samples from $\{p_i\}_{i=1}^s$. In the soundness case, we also assume that the p_i's satisfy the *structural condition* given in Definition 1: we assume there are disjoint sets A and B that partition $[n]$ such that all p_i's are larger than q on the indices in A, and all the p_i's are smaller than q on the indices in B. Note that *structural condition* trivially holds in the completeness case.

4.1 Algorithm for Identity Testing

We now present our algorithm, Identity-Tester, for identity testing with multiple sources. Suppose we receive Poi(1) samples from each of the distributions p_i. This is a generalization of the standard technique in distribution testing which significantly simplifies the analysis of our algorithm by making certain random variables independent, as we explain later. Furthermore, as Poi(s) is tightly concentrated around s, we can carry out this poissonization method at the expense of only constant factor increases in the sample complexity. Moreover, while we draw one sample in expectation per source, with probability 0.9, we will not receive more than $O(\log s)$ samples per distribution.

Our algorithm calculates a new χ-square type statistics inspired by the previous χ^2-type statistics [32, 1, 10, 16]). The statistic is designed so that its expected value is related to the "ℓ_2-norm" of the difference of the distributions, as explained in Section 2.3. Similarly to uniformity testing, our algorithm in this section also proceeds by taking samples and calculating our statistic. Then, it compares the value of this statistic to a threshold τ. If the value of the statistic is "large", the algorithm outputs reject and aborts, and outputs accept otherwise. Ultimately, we prove that the sample complexity of our generalized identity tester depends on the ℓ_2-norm of q, the known distribution. We give a flattening procedure in Section 4.2 which allows us to assume that the ℓ_2-norm of the known distribution is $O(1/\sqrt{n})$, resulting in the optimal sample complexity. We present our algorithm below along with the main theorem, Theorem 9, which proves the correctness of our algorithm.

Algorithm 2 Identity-Tester.

Input : n, ϵ, q, Poi(1) samples from each of $p_1, p_2, \cdots, p_s$
Output : accept or reject

$s \leftarrow \frac{c_1 n \|q\|_2}{\epsilon^2}$
Draw Poi(1) samples from each of the s distributions $\{p_j\}_{j=1}^s$.
$T_x \leftarrow$ # times we see element $x \in [n]$ among the samples.
$\tau \leftarrow \frac{5s^2\epsilon^2}{8n}$
$Z \leftarrow \sum_{x\in[n]} (T_x - sq(x))^2 - T_x$
if $Z \geq \tau$ **then**
    Output reject and abort.
Output accept

▶ **Theorem 9** (Correctness of Identity-Tester)**.** *There exist a constant c_1 independent of n such that the following statements hold with probability 2/3:*

- Identity-Tester *outputs accept if each of the s distributions $p_1, \cdots, p_s$ are equal to q.*
- Identity-Tester *outputs reject if the p_i's are ϵ-far from q ($\|p_i - q\|_1 \geq \epsilon$) and $\{p_i\}_{i=1}^s$ satisfy the structural condition given in Definition 1.*

▶ Remark 10. The sample complexity of Algorithm 2 is $\Theta(n\|q\|_2/\epsilon^2)$. Using the flattening procedure of Section 4.2, the sample complexity of Identity-Tester reduces to $\Theta(\sqrt{n}/\epsilon^2)$ which is optimal since the lower bound of $\Omega(\sqrt{n}/\epsilon^2)$ holds for identity testing in the standard single distribution setting [28].

▶ Remark 11. Note that identity testing is a generalization of uniformity testing in Section 3. However, we keep our approach for uniformity testing since we only use *exactly one* sample per distribution, rather than one sample in expectation.

Overview of the proof. To prove the correctness of IDENTITY-TESTER, we analyze the statistic

$$Z = \sum_{x\in[n]} (T_x - s\,q(x))^2 - T_x\,,$$

where T_x denotes the number of times we observe element x among our $s' \sim \mathrm{Poi}(s)$ samples. Note that by employing the poissonization method, T_x is a Poisson random variable with parameter $\lambda_x \coloneqq \sum_{j=1}^{s} p_j(x)$ and T_x and T_y are independent for $x \neq y$. The independence among T_x's greatly simplifies our calculations.

Our goal is to show Z is below the threshold τ in the completeness case, and above the threshold in the soundness case. To do so, we make use of the *structural condition* to first define a convenient representation of $\mathbf{E}[Z]$ in Lemma 12. We then show a strong concentration around its expectation by bounding the variance of Z in Lemma 13. Finally, we show that Z is always on the desired side of the threshold proving the correctness of our algorithm.

Proof of Theorem 9. Note that we set the (expected) number of samples to be $s = c_1 n\|q\|_2/\epsilon^2$ for some sufficiently large constant c_1, and the threshold τ to be equal to $5s^2\epsilon^2/(8n)$. We begin by stating a convenient representation of $\mathbf{E}[Z]$. To motivate our calculations, note that for a fixed x,

$$\begin{aligned}\mathbf{E}[(T_x - s\,q(x))^2 - T_x] &= \mathbf{E}[T_x^2] - \mathbf{E}[T_x] - 2sq(x)\mathbf{E}[T_x] + s^2q(x)^2\\ &= \lambda_x^2 - 2s\,q(x)\lambda_x + s^2q(x)^2 = (\lambda_x - s\,q(x))^2\end{aligned}$$

which follows from the fact that T_x is a Poisson random variable with parameter $\lambda_x = \sum_{j=1}^{s} p_j(x)$. Now using the *structural condition*, we can define error terms similar to our uniformity testing section. For each distribution p_j, we define

$$\begin{aligned} e_j(x) &= p_j(x) - q(x) \qquad \forall x \in A\\ e_j(x) &= q(x) - p_j(x) \qquad \forall x \in B.\end{aligned}$$

After plugging in $e_j(x)$ for all x into our expression for λ_x, we combine these terms into a more useful representation of $\mathbf{E}[Z]$. We precisely show this representation in Lemma 12 where we prove that $\mathbf{E}[Z]$ is given by $\|\vec{\mathbf{e}}_1 + \cdots + \vec{\mathbf{e}}_s\|_2^2$ where we interpret the vector $\vec{\mathbf{e}}_j \in \mathbb{R}^n$ as the vector with entries $e_j(x) = |q(x) - p_j(x)|$. Note that this is a natural generalization of the quantity $s^2\|q-p\|_2^2$ which is the quantity calculated by all χ^2-based testers in the single distribution setting of identity testing (where all the samples are i.i.d. from a fixed distribution p). More formally, we have the following lemma which we prove in Section 4.3.

▶ **Lemma 12.** *Let $\{p_i\}_{i=1}^s$ be distributions over $[n]$ that satisfy the* structural condition *given in Definition 1. Suppose we draw* $\mathrm{Poi}(1)$ *samples from each p_i and let T_x be the number of times we see element $x \in [n]$ among the samples. Let $Z = \sum_{x\in[n]}(T_x - sq(x))^2 - T_x$. Then,*

$$\mathbf{E}[Z] = \|\vec{\mathbf{e}}_1 + \cdots + \vec{\mathbf{e}}_s\|_2^2$$

where the x-th coordinate of $\vec{\mathbf{e}}_j \in \mathbb{R}^n$ is $|q(x) - p_j(x)|$.

We now give a tight upper bound for the variance of our statistic Z. Let Z_x denote the x-th term in Z, $(T_x - sq(x))^2 - T_x$. As we establish earlier, using the Poissonization method, T_x's are independent from each other. Thus, the Z_x's are independent as well. Therefore, one can expand the variance of Z as bellow:

$$\mathbf{Var}[Z] = \sum_{x\in[n]} \mathbf{Var}[Z_x] = \sum_{x\in[n]} \mathbf{E}[Z_x^2] - \mathbf{E}[Z_x]^2\,.$$

As we expand the term Z_x^2, to bound $\mathbf{E}\left[Z_x^2\right]$, higher norms of T_x, i.e., $\mathbf{E}\left[T_x^k\right]$ for $k \in [4]$, appear in our calculation. We can compute the closed-form of these quantities via the known norms of the Poisson distribution. Combining these terms, we again get an upper bound of $\mathbf{Var}[Z]$ in terms of the vectors $\vec{\mathbf{e}}_j$. Formally, we prove the following lemma in Section 4.4.

▶ **Lemma 13.** *Let $\{p_i\}_{i=1}^s$ be s distributions over $[n]$ that satisfy the structural condition given in Definition 1. Suppose we draw* Poi(1) *samples from each p_i, and let T_x be the number of times we see element $x \in [n]$ among the samples. Let $Z = \sum_{x\in[n]}(T_x - sq(x))^2 - T_x$. Then, we have:*

$$\mathbf{Var}[Z] \le 4s\|q\|_2 \left\|\sum_{j=1}^s \vec{\mathbf{e}}_j\right\|_4^2 + 2\left\|\sum_{j=1}^s \vec{\mathbf{p}}_j\right\|_2^2 + 4\left\|\sum_{j=1}^s \vec{\mathbf{e}}_j\right\|_3^3,$$

where $|q(x) - p_j(x)|$ is the x-th coordinates of $\vec{\mathbf{e}}_j \in \mathbb{R}^n$, and $\vec{p_j}$ is the vector representation of the distribution p_j.

We can now proceed to the proof of the theorem in the completeness case.

Proof of the completeness case. In this case, Lemma 12 gives us $\mathbf{E}[Z] = 0$, and Lemma 13 gives us $\mathbf{Var}[Z] \le 2s^2\|q\|_2^2$. Therefore by Chebyshev's inequality,

$$\Pr[Z \ge \tau] \le \Pr\left[|Z| = |Z - \mathbf{E}[Z]| \ge \frac{s^2\epsilon^2}{4n}\right] \le \frac{32s^2\|q\|_2^2 n^2}{s^4\epsilon^4} = \frac{32\|q\|_2^2 n^2}{s^2\epsilon^4}.$$

Recall that we let $s = c_1 n\|q\|_2/\epsilon^2$. The right hand side of the above inequality can be made arbitrarily small by picking a sufficiently large constant c_1, which proves the completeness case.

Proof of the soundness case. In this case, Lemma 12 gives us

$$\mathbf{E}[Z] = \|\vec{\mathbf{e}}_1 + \cdots + \vec{\mathbf{e}}_s\|_2^2 = \sum_{x\in[n]}\left(\sum_{j=1}^s e_j(x)\right)^2 \ge \frac{1}{n}\left(\sum_{j=1}^s\sum_{x\in[n]} e_j(x)\right)^2 \ge \frac{s^2\epsilon^2}{n}$$

where the first inequality is Cauchy-Schwarz, and the second inequality follows from the fact that

$$\sum_{x\in[n]} e_j(x) = \|q - p_j\|_1 \ge \epsilon$$

for each $j \in [s]$. Then by Chebyshev's inequality and Lemma 13,

$$\begin{aligned}\Pr\left[|Z - \mathbf{E}[Z]| \ge \frac{\mathbf{E}[Z]}{4}\right] &\le \frac{16\mathbf{Var}[Z]}{\mathbf{E}[Z]^2} \\ &\le \frac{4s\|q\|_2\left\|\sum_{j=1}^s \vec{\mathbf{e}}_j\right\|_4^2}{\left\|\sum_{j=1}^s \vec{\mathbf{e}}_j\right\|_2^4} + \frac{2\left\|\sum_{j=1}^s \vec{\mathbf{p}}_j\right\|_2^2}{\left\|\sum_{j=1}^s \vec{\mathbf{e}}_j\right\|_2^4} + \frac{4\left\|\sum_{j=1}^s \vec{\mathbf{e}}_j\right\|_3^3}{\left\|\sum_{j=1}^s \vec{\mathbf{e}}_j\right\|_2^4} \qquad (10)\end{aligned}$$

Now, we bound each of the three terms above separately. We start off by introducing a new distribution denoted by $\widetilde{\mathbf{p}}$ to be $\widetilde{\mathbf{p}} := \frac{1}{s}\sum_{j=1}^s \mathbf{p}_j$. In some of our calculation, this new representation simplifies our calculations.

- **First term:** Now, we focus on the first term in Equation (10). We note that all the $e_j(x)$ are positive, so we have:

$$\left\|\sum_{j=1}^{s}\vec{\mathbf{e}}_j\right\|_4^2 = \sqrt{\sum_{x\in[n]}\left(\sum_{j=1}^{s}e_j(x)\right)^4} \le \sum_{x\in[n]}\left(\sum_{j=1}^{s}e_j(x)\right)^2 \le \left\|\sum_{j=1}^{s}\vec{\mathbf{e}}_j\right\|_2^2. \tag{11}$$

  We again use the same Cauchy-Schwarz calculation as in $\mathbf{E}[Z]$ and get:

$$\left\|\sum_{j=1}^{s}\vec{\mathbf{e}}_j\right\|_2^2 = \sum_{x\in[n]}\left(\sum_{j=1}^{s}e_j(x)\right)^2 \ge \frac{1}{n}\left(\sum_{x\in[n]}\sum_{j=1}^{s}e_j(x)\right)^2 \ge \frac{s^2\epsilon^2}{n}. \tag{12}$$

  Therefore, we bound the first term from above:

$$\frac{4s\|q\|_2\left\|\sum_{j=1}^{s}\vec{\mathbf{e}}_j\right\|_4^2}{\left\|\sum_{j=1}^{s}\vec{\mathbf{e}}_j\right\|_2^4} \le \frac{4n\|q\|_2}{s\epsilon^2}.$$

- **Second term**: For the second term, we have:

$$\frac{2\left\|\sum_{j=1}^{s}\vec{\mathbf{p}}_j\right\|_2^2}{\left\|\sum_{j=1}^{s}\vec{\mathbf{e}}_j\right\|_2^4} = \frac{2s^2\|\widetilde{\mathbf{p}}\|_2^2}{\left\|\sum_{j=1}^{s}\vec{\mathbf{e}}_j\right\|_2^4}$$

  where $\widetilde{\mathbf{p}} = \frac{1}{s}\sum_{j=1}^{s}\mathbf{p}_j$. We now consider two cases. If it is the case that $\|\widetilde{\mathbf{p}}\|_2 \le 3\|q\|_2$, then we have:

$$\frac{2s^2\|\widetilde{\mathbf{p}}\|_2^2}{\left\|\sum_{j=1}^{s}\vec{\mathbf{e}}_j\right\|_2^4} \le \left(\frac{5n\|q\|_2}{s\epsilon^2}\right)^2.$$

  On the other hand, suppose that $\|\widetilde{\mathbf{p}}\|_2 > 3\|q\|_2$. Then, using our *structural condition*, we obtain:

$$\frac{2s^2\|\widetilde{\mathbf{p}}\|_2^2}{\left\|\sum_{j=1}^{s}\vec{\mathbf{e}}_j\right\|_2^4} = \frac{2\|\widetilde{\mathbf{p}}\|_2^2}{s^2\|\widetilde{\mathbf{p}}-q\|_2^4}$$

  and note that

$$\|\widetilde{\mathbf{p}}-q\|_2^2 \ge \|\widetilde{\mathbf{p}}\|_2^2 + \|q\|_2^2 - 2\|q\|_2\|\widetilde{\mathbf{p}}\|_2 \ge \|\widetilde{\mathbf{p}}\|_2^2/3.$$

  Hence,

$$\frac{2\|\widetilde{\mathbf{p}}\|_2^2}{s^2\|\widetilde{\mathbf{p}}-q\|_2^4} \le \frac{18}{s^2\|\widetilde{\mathbf{p}}\|_2^2} \le \left(\frac{5}{s\|q\|_2}\right)^2.$$

- **Third term**: Again, we use the Cauchy-Schwarz inequality to obtain:

$$\left\|\sum_{j=1}^{s}\vec{\mathbf{e}}_j\right\|_3^3 = \sum_{x\in[n]}\left(\sum_{j=1}^{s}e_j(x)\right)^3 \le \sqrt{\left(\sum_{x\in[n]}\left(\sum_{j=1}^{s}e_j(x)\right)^2\right)\cdot\left(\sum_{x\in[n]}\left(\sum_{j=1}^{s}e_j(x)\right)^4\right)}$$
$$\le \left\|\sum_{j=1}^{s}\vec{\mathbf{e}}_j\right\|_2\cdot\left\|\sum_{j=1}^{s}\vec{\mathbf{e}}_j\right\|_4^2$$

Now, we use Equation (11) and Equation (12), which we show earlier, to bound the third term:

$$\frac{4\left\|\sum_{j=1}^{s}\vec{\mathbf{e}}_j\right\|_3^3}{\left\|\sum_{j=1}^{s}\vec{\mathbf{e}}_j\right\|_2^4} \leq \frac{4\left\|\sum_{j=1}^{s}\vec{\mathbf{e}}_j\right\|_2 \cdot \left\|\sum_{j=1}^{s}\vec{\mathbf{e}}_j\right\|_4^2}{\left\|\sum_{j=1}^{s}\vec{\mathbf{e}}_j\right\|_2^4} \leq \frac{4}{\left\|\sum_{j=1}^{s}\vec{\mathbf{e}}_j\right\|_2} \leq \frac{4\sqrt{n}}{s\epsilon}.$$

Thus in all cases, we have

$$\Pr\left[|Z - \mathbf{E}[Z]| \geq \frac{\mathbf{E}[Z]}{4}\right] \leq \frac{4n\|q\|_2}{s\epsilon^2} + \left(\frac{5n\|q\|_2}{s\epsilon^2}\right)^2 + \left(\frac{5}{s\|q\|_2}\right)^2 + \frac{4\sqrt{n}}{s\epsilon}.$$

Note that $s = c_1 n\|q\|_2/\epsilon^2$ and $\|q\|_2 \geq 1/\sqrt{n}$. Therefore, by letting c_1 be a sufficiently large constant, we get that the above probability is smaller than $1/3$. Hence with probability at least $2/3$, we know $Z \geq 3s^2\epsilon^2/(4n)$ in the soundness case, so we reject with probability at least $2/3$, as desired. ◀

4.2 Flattening Procedure

In this section, we present the flattening procedure which allows us to assume that $\|q\|_2^2 = O(1/n)$ in Remark 10 without loss of generality where q is our known distribution. While this procedure is similar to the one used in [16], we state it here for the sake of completeness. For each $x \in [n]$, define

$$b_x \coloneqq \lfloor nq(x) \rfloor + 1.$$

We note that $b_x \geq 1$ for each $x \in [n]$. Given a sample x from a distribution p over $[n]$, we can get a sample from the "flattened" distribution p' over a new domain, $\mathcal{D}$, defined as

$$\mathcal{D} \coloneqq \{(x, y) \mid x \in [n], y \in [b_x]\},$$

by drawing an element from $y \in [b_x]$ uniformly at random and creating the tuple (x, y). This is the flattening procedure that we use for our version of identity testing. Note that the probability mass over $[n]$ placed by p gets "flattened" to be a probability distribution over the domain $\mathcal{D}$.

Furthermore, this procedure has a few desired properties which we state here: First, the size of this new domain is $O(n)$:

$$|\mathcal{D}| = \sum_{x\in[n]} b_x \leq 2n.$$

Second, the procedure preserves the ℓ_1-distance between two distributions: let p' and q' denote the flattened versions of p and q. Then, we have:

$$\|q' - p'\|_1 = \sum_{x\in[n]}\sum_{y\in[b_x]} \frac{|q(x) - p(x)|}{|b_x|} = \sum_{x\in[n]} |q(x) - p(x)| = \|q - p\|_1.$$

Third, by definition of the b_x's, we can show that q' has a low ℓ_2-norm:

$$\|q'\|_2^2 = \sum_{x\in[n]}\sum_{y\in[b_x]} \frac{q(x)^2}{b_x^2} = \sum_{x\in[n]} \frac{q(x)^2}{b_x} \leq \sum_{x\in[n]} \frac{q(x)}{n} \leq \frac{1}{n}.$$

The above inequality implies that the ℓ_2-norm of q' is within a constant factor of the smallest possible norm, which is $(|\mathcal{D}|)^{-1/2}$.

Therefore whenever we get a sample over $[n]$ in IDENTITY-TESTER, we can use this flattening procedure to draw a sample over $\mathcal{D}$. By using this flattening procedure to draw samples from a slightly larger domain, we can assume that the ℓ_2-norm of the known distribution q is $O(1/\sqrt{n})$. Note that since the size of the larger domain is still $O(n)$, the flattening procedure only affects the sample complexity up to constant factors. Therefore, by combining with Theorem 9, we can perform our generalized version of identity testing by using $s = O(\sqrt{n}/\epsilon^2)$ samples, which is optimal up to constant factors.

4.3 Proof of Lemma 12

▶ **Lemma 12.** *Let $\{p_i\}_{i=1}^s$ be distributions over $[n]$ that satisfy the* structural condition *given in Definition 1. Suppose we draw* Poi(1) *samples from each p_i and let T_x be the number of times we see element $x \in [n]$ among the samples. Let $Z = \sum_{x\in[n]}(T_x - sq(x))^2 - T_x$. Then,*

$$\mathbf{E}[Z] = \|\vec{\mathbf{e}}_1 + \cdots + \vec{\mathbf{e}}_s\|_2^2$$

where the x-th coordinate of $\vec{\mathbf{e}}_j \in \mathbb{R}^n$ is $|q(x) - p_j(x)|$.

Proof. Let

$$Z_x = (T_x - sq(x))^2 - T_x.$$

We have:

$$Z_x^2 = T_x^2 - 2sq(x)T_x + s^2q(x)^2 - T_x.$$

We can compute that:

$$\begin{aligned}\mathbf{E}[Z_x] &= \mathbf{E}[T_x^2] - \mathbf{E}[T_x] - 2sq(x)\mathbf{E}[T_x] + s^2q(x)^2 \\ &= \lambda_x^2 - 2sq(x)\lambda_x + s^2q(x)^2\end{aligned}$$

where we have used the fact that the variance of a Poisson random variable with parameter λ is also λ. We introduce the following notation $(-1)^{x\in B}$ which is defined as

$$(-1)^{x\in B} = \begin{cases} -1 & \text{if } x \in B, \\ 1 & \text{if } x \notin B. \end{cases}$$

Using the fact that $\lambda_x = \sum_{j=1}^s p_j(x)$, we can compute that

$$\sum_{x\in[n]} \lambda_x = \sum_{j=1}^{s}\sum_{x\in[n]} \left(q(x) + (-1)^{x\in B}e_j(x)\right).$$

In Appendix A, we calculate the $\sum_{x\in[n]} \lambda_x^2$. There we show that

$$\begin{aligned}\sum_{x\in[n]} \lambda_x^2 = s^2||q||_2^2 + 2s\sum_{j=1}^{s}\sum_{x\in[n]} (-1)^{x\in B}q(x)e_j(x) \\ + \sum_{j=1}^{s}\sum_{x\in[n]} e_j(x)^2 + \sum_{j\neq k}\sum_{x\in[n]} e_j(x)e_k(x).\end{aligned}$$

Using these two results, we have:

$$\begin{aligned}\sum_{x\in[n]}\mathbf{E}[Z_x] &= \sum_{x\in[n]}\lambda_x^2 - 2s\sum_{x\in[n]}q(x)\lambda_x + s^2\sum_{x\in[n]}q(x)^2\\ &= 2s\sum_{j=1}^{s}\sum_{x\in[n]}(-1)^{x\in B}q(x)e_j(x) + \sum_{j=1}^{s}\sum_{x\in[n]}e_j(x)^2 + \sum_{j\neq k}\sum_{x\in[n]}e_j(x)e_k(x)\\ &\quad - 2s\sum_{j=1}^{s}\sum_{x\in[n]}(q(x)^2 + (-1)^{x\in B}q(x)e_j(x)) + 2s^2\|q\|_2^2\\ &= \sum_{j=1}^{s}\sum_{x\in[n]}e_j(x)^2 + \sum_{j\neq k}\sum_{x\in[n]}e_j(x)e_k(x)\\ &= \|\vec{\mathbf{e}}_1 + \cdots + \vec{\mathbf{e}}_s\|_2^2.\end{aligned}$$

Therefore, our final result is as follows:

$$\mathbf{E}[Z] = \|\vec{\mathbf{e}}_1 + \cdots + \vec{\mathbf{e}}_s\|_2^2,$$

as desired. ◀

▶ Remark 14. Note that the quantity $\|\vec{\mathbf{e}}_1 + \cdots + \vec{\mathbf{e}}_s\|_2^2$ is a natural generalization of the quantity $s^2\|q-p\|_2^2$ which is the expectation of the random variable calculated by identity testers in the single distribution case where samples come from a fixed distribution p.

4.4 Proof of Lemma 13

▶ **Lemma 13.** *Let $\{p_i\}_{i=1}^s$ be s distributions over $[n]$ that satisfy the structural condition given in Definition 1. Suppose we draw* Poi(1) *samples from each p_i, and let T_x be the number of times we see element $x \in [n]$ among the samples. Let $Z = \sum_{x\in[n]}(T_x - sq(x))^2 - T_x$. Then, we have:*

$$\mathbf{Var}[Z] \le 4s\|q\|_2\left\|\sum_{j=1}^{s}\vec{\mathbf{e}}_j\right\|_4^2 + 2\left\|\sum_{j=1}^{s}\vec{\mathbf{p}}_j\right\|_2^2 + 4\left\|\sum_{j=1}^{s}\vec{\mathbf{e}}_j\right\|_3^3,$$

where $|q(x)-p_j(x)|$ is the x-th coordinates of $\vec{\mathbf{e}}_j \in \mathbb{R}^n$, and $\vec{p_j}$ is the vector representation of the distribution p_j.

Proof. Let

$$Z_x = (T_x - sq(x))^2 - T_x.$$

Due to the independence of T_x, we have

$$\mathbf{Var}[Z] = \sum_{x\in[n]}\mathbf{Var}[Z_x] = \sum_{x\in[n]}\mathbf{E}[Z_x^2] - \mathbf{E}[Z_x]^2$$

where

$$Z_x = (T_x - sq(x))^2 - T_x.$$

Noting that T_x is a Poisson with parameter λ_x, we can compute that

$$\mathbf{E}[Z_x^2] = \lambda_x^4 + 4\lambda_x^3(1-sq(x)) + 2\lambda_x^2(1-4sq(x)+3s^2q(x)^2) + 4s^2q(x)^2\lambda_x(1-sq(x)) + s^4q(x)^4.$$

In Appendix A we calculate the $\sum_{x\in[n]} \lambda_x^k$ for $k \in \{1,2,3,4\}$. Using these results and simplifying, we arrive at the following expression:

$$\begin{aligned}
&\sum_{x\in[n]} \mathbf{E}[Z_x^2] = 4s\sum_{j=1}^{s}\sum_{x\in[n]} q(x)e_j(x)^2 + 4s\sum_{j\neq k}\sum_{x\in[n]} q(x)e_j(x)e_k(x) + 2s^2\|q\|_2^2\\
&+ 4s\sum_{j=1}^{s}\sum_{x\in[n]} (-1)^{x\in B}q(x)e_j(x) + 2\sum_{j=1}^{s}\sum_{x\in[n]} e_j(x)^2 + 2\sum_{j\neq k}\sum_{x\in[n]} e_j(x)e_k(x)\\
&+ 4\sum_{j=1}^{s}\sum_{x\in[n]} (-1)^{x\in B}e_j(x)^3 + 12\sum_{j\neq k}\sum_{x\in[n]} (-1)^{x\in B}e_j(x)^2e_k(x)\\
&+ 4\sum_{j\neq k\neq \ell}\sum_{x\in[n]} (-1)^{x\in B}e_j(x)e_k(x)e_\ell(x) + \left(\sum_{j=1}^{s}\sum_{x\in[n]} e_j(x)^4 + 6\sum_{j\neq k\neq \ell} e_j(x)^2e_k(x)e_\ell(x)\right.\\
&\left.+ 4\sum_{j\neq k} e_j(x)^3e_k(x) + 3\sum_{j\neq k} e_j(x)^2e_k(x)^2 + \sum_{j\neq k\neq \ell\neq t} e_j(x)e_k(x)e_\ell(x)e_t(x)\right).
\end{aligned}$$

We now simplify the above expression. First note that

$$\begin{aligned}
&4s\sum_{j=1}^{s}\sum_{x\in[n]} q(x)e_j(x)^2 + 4s\sum_{j\neq k}\sum_{x\in[n]} q(x)e_j(x)e_k(x)\\
&= 4s\sum_{x\in[n]} q(x)\left(\sum_{j=1}^{s} e_j(x)^2 + \sum_{j\neq k} e_j(x)e_k(x)\right) = 4s\sum_{x\in[n]} q(x)\left(\sum_{j=1}^{s} e_j(x)\right)^2\\
&\leq 4s\|q\|_2\|\vec{\mathbf{e}}_1 + \cdots + \vec{\mathbf{e}}_s\|_4^2.
\end{aligned}$$

Furthermore,

$$\begin{aligned}
&2s^2\|q\|_2^2 + 4s\sum_{j=1}^{s}\sum_{x\in[n]} (-1)^{x\in B}q(x)e_j(x) + 2\sum_{j=1}^{s}\sum_{x\in[n]} e_j(x)^2 + 2\sum_{j\neq k}\sum_{x\in[n]} e_j(x)e_k(x)\\
&= 2\sum_{j\neq k}\sum_{x\in[n]} (q(x) + (-1)^{x\in B}e_j(x))(q(x) + (-1)^{x\in B}e_k(x))\\
&= 2\|\vec{\mathbf{p}}_1 + \cdots \vec{\mathbf{p}}_s\|_2^2
\end{aligned}$$

and

$$\begin{aligned}
&4\sum_{j=1}^{s}\sum_{x\in[n]} (-1)^{x\in B}e_j(x)^3 + 12\sum_{j\neq k}\sum_{x\in[n]} (-1)^{x\in B}e_j(x)^2e_k(x)\\
&+ 4\sum_{j\neq k\neq \ell}\sum_{x\in[n]} (-1)^{x\in B}e_j(x)e_k(x)e_\ell(x)\\
&\leq 4\left(\sum_{j=1}^{s}\sum_{x\in[n]} e_j(x)^3 + 3\sum_{j\neq k}\sum_{x\in[n]} e_j(x)^2e_k(x) + \sum_{j\neq k\neq \ell}\sum_{x\in[n]} e_j(x)e_k(x)e_\ell(x)\right)\\
&= 4\|\vec{\mathbf{e}}_1 + \cdots + \vec{\mathbf{e}}_s\|_3^3.
\end{aligned}$$

Finally, the last expression inside the parenthesis in the expression for $\sum_{x\in[n]} \mathbf{E}[Z_x^2]$ which is:

$$\sum_{j=1}^{s}\sum_{x\in[n]} e_j(x)^4 + 6\sum_{j\neq k\neq \ell} e_j(x)^2 e_k(x) e_\ell(x) + 4\sum_{j\neq k} e_j(x)^3 e_k(x) + 3\sum_{j\neq k} e_j(x)^2 e_k(x)^2 + \sum_{j\neq k\neq \ell\neq t} e_j(x)e_k(x)e_\ell(x)e_t(x)$$

is precisely $\sum_{x\in[n]} \left(\sum_{j=1}^{s} e_j(x)\right)^4$. Therefore, we obtain the following bound:

$$\sum_{x\in[n]} \mathbf{E}[Z_x^2] \leq 4s\|q\|_2 \left\|\sum_{j=1}^{s}\vec{\mathbf{e}}_j\right\|_4^2 + 2\left\|\sum_{j=1}^{s}\vec{\mathbf{p}}_j\right\|_2^2 + 4\left\|\sum_{j=1}^{s}\vec{\mathbf{e}}_j\right\|_3^3 + \sum_{x\in[n]}\left(\sum_{j=1}^{s} e_j(x)\right)^4.$$

Using the calculation for $\mathbf{E}[Z_x]$ that we performed for Lemma 12, we see that:

$$\sum_{x\in[n]} \mathbf{E}[Z_x]^2 = \sum_{x\in[n]}\left(\sum_{j=1}^{s} e_j(x)^2 + \sum_{j\neq k} e_j(x)e_k(x)\right)^2 = \sum_{x\in[n]}\left(\sum_{j=1}^{s} e_j(x)\right)^4$$

so altogether,

$$\mathbf{Var}[Z] \leq 4s\|q\|_2 \left\|\sum_{j=1}^{s}\vec{\mathbf{e}}_j\right\|_4^2 + 2\left\|\sum_{j=1}^{s}\vec{\mathbf{p}}_j\right\|_2^2 + 4\left\|\sum_{j=1}^{s}\vec{\mathbf{e}}_j\right\|_3^3,$$

as desired. ◀

▶ Remark 15. As stated in Section 4.3, the quantity $\left\|\sum_{j=1}^{s}\vec{\mathbf{e}}_j\right\|_4^2$ is a natural generalization of $s^2\|q-p\|_2^2$ which appears in the variance calculations of the statistic used by identity testers in the single distribution setting where samples come from a fixed distribution p. Similarly, the quantity $\left\|\sum_{j=1}^{s}\vec{\mathbf{p}}_j\right\|_2^2$ is a natural generalization of $s^2\|p\|_2^2$ which also appears in these calculations.

5 Closeness Testing with Multiple Sources

in this section, we present our algorithm for closeness testing with multiple sources and its analysis. Recall that our goal is to distinguish the following two cases with probability at least 2/3:

- **Completeness case:** $p_1, p_2, \cdots$ are identical to q
- **Soundness case:** $p_1, p_2, \cdots$ are all ϵ-far from q in ℓ_1-distance

where we have access to two streams. In the first stream, all samples are i.i.d. from some fixed distribution q over $[n]$ that is unknown to us while in the second stream, we receive samples from $\{p_i\}_{i=1}^{s}$. In the soundness case, we also assume that the p_i's satisfy the *structural condition* given in Definition 1: we assume there are disjoint sets A and B that partition $[n]$ such that all p_i's are larger than q on the indices in A, and all the p_i's are smaller than q on the indices in B. Note that the *structural condition* is trivially satisfied in the completeness case.

5.1 Algorithm for Closeness Testing

We now present our algorithm, Closeness-Tester, for closeness testing with multiple sources. Overall, our approach is very similar to our approach in Section 4. As in the identity testing section, we again make the assumption that we are able to receive Poi(1) samples from each of the distributions p_i. This is a generalization of the standard assumption in the single source closeness testing [10, 16] and it significantly simplifies the analysis of our algorithm by making certain random variables independent, similar to the identity testing section. Furthermore, as Poi(s) is tightly concentrated around s, we can carry out this poissonization method at the expense of only constant factor increases in the sample complexity.

As in the identity testing section, the statistic calculated by our algorithm is introduced in [10, 16]. We show that the expected value of our statistic is related to the "ℓ_2-norm" of the difference of the distributions, as explained in Section 2.3. If the value of the statistic is "large" compared to some threshold τ, the algorithm outputs reject and outputs accept otherwise. As in uniformity testing, the challenge is to show that the value of this statistic concentrates which we do so by analyzing its variance. Ultimately, we prove that the sample complexity of our generalized identity tester depends on the ℓ_2-norm of q, the distribution that we have i.i.d. sample access from. We then present a randomized flattening procedure in Section 5.2 which shows how to reduce the ℓ_2-norm of q to $O(1/\sqrt{k})$ where k is a parameter in our algorithm. This randomized flattening procedure is slightly different than the one used in Section 4.2 since we do not know q in advance. Therefore, we must use some samples from q to towards this procedure. We present our algorithm below along with the main theorem, Theorem 16, which proves the correctness of our algorithm.

Algorithm 3 Closeness-Tester.

Input : n, ϵ, sample access to q, Poi(1) samples from each of $p_1, p_2, \cdots, p_s$
Output : Accept or Reject

$k \leftarrow \frac{n^{2/3}}{\epsilon^{4/3}}$
Draw Poi(k) samples from q to perform the randomized flattening procedure given in Section 5.2.
$s \leftarrow \frac{c_1 n}{\epsilon^2 \sqrt{k}}$
Draw Poi(s) samples from q.
Draw Poi(1) samples from each of the s distributions $\{p_j\}_{j=1}^s$.
$Y_x \leftarrow$ # times we see element $x \in [n]$ among the Poi(s) samples from q
$T_x \leftarrow$ # times we see element $x \in [n]$ among the samples from $\{p_j\}_{j=1}^s$.
$\tau \leftarrow \frac{5s^2\epsilon^2}{8n}$
$Z \leftarrow \sum_{x\in[n]} (T_x - Y_x)^2 - T_x - Y_x$
if $Z \geq \tau$ **then**
    Output reject and abort.
Output accept

▶ **Theorem 16** (Correctness of Closeness-Tester)**.** *There exist a constant c_1 independent of n such that the following statements hold with probability 2/3:*

- Closeness-Tester *outputs accept if each of the s distributions $p_1, \cdots, p_s$ are q.*
- Closeness-Tester *outputs reject if each of the p_i's are ϵ-far from q ($\|p_i - q\|_1 \geq \epsilon$) and $\{p_i\}_{i=1}^s$ satisfy the structural condition given in Definition 1.*

▶ Remark 17. The sample complexity of Closeness-Tester is $\Theta(k + n/(\epsilon^2\sqrt{k}))$ using the flattening procedure of Section 5.2. Optimizing in k, the sample complexity of **Closeness-Tester** reduces to $\Theta(n^{2/3}/\epsilon^{4/3} + \sqrt{n}/\epsilon^2)$ which is optimal since the same lower bound holds for closeness testing in the standard single distribution setting [16].

▶ Remark 18. Note that given the above result, we may need fewer sources as long as we have more samples from the distribution q. In particular, suppose we use $k_1 = \Theta\left(\min\left(n^{2/3}/\epsilon^{4/3} + \sqrt{n}/\epsilon^2, n\right)\right)$ samples from q for flattening, which implies that the ℓ_2-norm of the "flattened" q is $O(1/\sqrt{k_1})$ with high probability. Then, we can use $s = \Theta(n/(\sqrt{k_1}\,\epsilon^2) + \sqrt{n}/\epsilon^2)$ sources, and $\Theta(s)$ samples from the "flattened" q and use **Closeness-Tester** to distinguish between the completeness and the soundness case. This is an sample-optimal trade off up to constant factors since the same lower bound holds in the standard single distribution setting [16].

Overview of the proof. To prove the correctness of Closeness-Tester, we analyze the statistic

$$Z = \sum_{x\in[n]} (T_x - Y_x)^2 - T_x - Y_x$$

where T_x denote the number of times we observe element x among the samples from $\{p_i\}_{i=1}^s$ and Y_x denotes the number of times we see element x among the $\mathrm{Poi}(s)$ samples from q. Note that by the poissonization method, T_x is a Poisson random variable with parameter $\lambda_x = \sum_{j=1}^s p_j(x)$, similar to the identity testing section. Furthermore, T_x and T_y are independent for $x \neq y$ and furthermore, Y_x is a Poisson random variable with parameter $s\,q(x)$. Our goal is to show Z is below the threshold τ in the completeness case, and above the threshold in the soundness case. To do so, we make use of the *structural condition* to first define a convenient representation of $\mathbf{E}[Z]$ in Lemma 19. We then show a strong concentration around its expectation by bounding the variance of Z in Lemma 20. These two lemmas are very similar to the ones proved in Section 4 due to the similarity of the statistic Z used here and in Section 4. Finally, we show that Z is always on the desired side of the threshold proving the correctness of our algorithm.

Proof of Theorem 16. Note that we set the number of samples to be $c_1 n/(\epsilon^2\sqrt{k})$ for a sufficiently large constant c_1, and the threshold τ to be equal to $5s^2\epsilon^2/(8n)$. We begin by stating a convenient representation of $\mathbf{E}[Z]$. To motivate our calculations, note that for a fixed x,

$$\begin{aligned}\mathbf{E}[(T_x - Y_x)^2 - T_x - Y_x] &= \mathbf{E}[T_x^2] - \mathbf{E}[T_x] + \mathbf{E}[Y_x^2] - \mathbf{E}[Y_x] - 2\mathbf{E}[Y_x]\mathbf{E}[T_x] \\ &= \lambda_x^2 - 2sq(x)\lambda_x + s^2q(x)^2\end{aligned}$$

where we have used the fact that T_x and Y_x are Poisson random variables with parameters $\lambda_x = \sum_{j=1}^s p_j(x)$ and $sq(x)$ respectively. Now using the *structural condition*, we can define error terms similar to Sections 3 and 4. For each distribution p_j, we define

$$\begin{aligned} e_j(x) &= p_j(x) - q(x) \qquad \forall x \in A \\ e_j(x) &= q(x) - p_j(x) \qquad \forall x \in B. \end{aligned}$$

After plugging in $e_j(x)$ for for all x into our expression for λ_x, we again get terms of the form $\sum_j e_j(x)^2$ and cross terms $\sum_{j\neq k} e_j(x)e_k(x)$, similar to the proof of Theorem 9. Our goal is to combine these terms into a more useful representation. We precisely show this in

Lemma 19 where we prove that $\mathbf{E}[Z]$ is given by $\|\vec{\mathbf{e}}_1 + \cdots + \vec{\mathbf{e}}_s\|_2^2$ where we interpret the vector $\vec{\mathbf{e}}_j \in \mathbb{R}^n$ as the vector with entries $e_j(x) = |q(x) - p_j(x)|$. Note that this is a natural generalization of the quantity $s^2\|q - p\|_2^2$. More formally, we have the following lemma which we prove in Section 5.3.

▶ **Lemma 19.** *Let $\{p_i\}_{i=1}^s$ be distributions over $[n]$ that satisfy the structural condition given in Definition 1. Suppose we draw* $\mathrm{Poi}(1)$ *samples from each p_i and* $\mathrm{Poi}(s)$ *samples from q and let T_x be the number of times we see element $x \in [n]$ among the samples among the p_i, and let Y_x be the number of times we see element $x \in [n]$ among the samples from q. Let $Z = \sum_{x\in[n]}(T_x - y_x)^2 - T_x - Y_x$. Then*

$$\mathbf{E}[Z] = \|\vec{\mathbf{e}}_1 + \cdots + \vec{\mathbf{e}}_s\|_2^2$$

where $\vec{\mathbf{e}}_j \in \mathbb{R}^n$ has coordinates $|q(x) - p_j(x)|$.

We now give a tight upper bound for the variance of our statistic Z. Defining $Z_x = (T_x - Y_x)^2 - T_x - Y_x$ and recalling the poissonization method, we see that

$$\mathbf{Var}[Z] = \sum_{x\in[n]} \mathbf{Var}[Z_x] = \sum_{x\in[n]} \mathbf{E}[Z_x^2] - \mathbf{E}[Z_x]^2.$$

Expanding Z_x^2, we get terms involving λ_x^k for $k \in \{1, 2, 3, 4\}$. Combining these terms, we again get an upper bound of $\mathbf{Var}[Z]$ in terms of the vectors $\vec{\mathbf{e}}_j$. Formally, we prove the following lemma in Section 5.4.

▶ **Lemma 20.** *Let $\{p_i\}_{i=1}^s$ be distributions over $[n]$ that satisfy the structural condition given in Definition 1. Suppose we draw* $\mathrm{Poi}(1)$ *samples from each p_i and* $\mathrm{Poi}(s)$ *samples from q and let T_x be the number of times we see element $x \in [n]$ among the samples among the p_i, and let Y_x be the number of times we see element $x \in [n]$ among the samples from q. Let $Z = \sum_{x\in[n]}(T_x - y_x)^2 - T_x - Y_x$. Then*

$$\mathrm{Var}(Z) \le 8s\|q\|_2 \left\|\sum_{j=1}^s \vec{\mathbf{e}}_j\right\|_4^2 + 8\left\|\sum_{j=1}^s \vec{\mathbf{p}}_j\right\|_2^2 + 4\left\|\sum_{j=1}^s \vec{\mathbf{e}}_j\right\|_3^3$$

where $\vec{\mathbf{e}}_j \in \mathbb{R}^n$ has coordinates $|q(x) - p_j(x)|$ and $\vec{\mathbf{p}}_j \in \mathbb{R}^n$ has coordinates $p_j(x)$.

We can now proceed to the proof of the theorem in the completeness case.

Proof of the Completeness Case. In this case, Lemma 19 gives us $\mathbf{E}[Z] = 0$ and Lemma 20 gives us $\mathbf{Var}[Z] \le 8s^2\|q\|_2^2$. Therefore by Chebyshev's inequality,

$$\Pr[|Z| \ge \tau] = \Pr\left[|Z| \ge \frac{s^2\epsilon^2}{4n}\right] \le \frac{128s^2\|q\|_2^2 n^2}{s^4\epsilon^4} = \frac{128\|q\|_2^2 n^2}{s^2\epsilon^4}.$$

The right hand side of the above inequality can be made arbitrarily small by letting $s = c_1 n\|q\|_2/\epsilon^2$ for a sufficiently large constant c_1. Due to our randomized flattening procedure of Section 5.2, we can assume that $\|q\| = O(1/\sqrt{k})$. Therefore, we can make the above probability bound arbitrarily small by letting $s = c_1 n/(\epsilon^2\sqrt{k})$ for a sufficiently large constant c_1.

Proof of the Soundness Case. In this case,

$$\mathbf{E}[Z] = \|\vec{\mathbf{e}}_1 + \cdots + \vec{\mathbf{e}}_s\|_2^2 = \sum_{x\in[n]} \left(\sum_{j=1}^s e_j(x)\right)^2 \ge \frac{1}{n}\left(\sum_{j=1}^s \sum_{x\in[n]} e_j(x)\right)^2 \ge \frac{s^2\epsilon^2}{n}$$

where the first inequality is Cauchy-Schwarz and the last inequality follows from our assumption about the error terms $e_j(x)$ in the soundness case. Then by Chebyshev's inequality,

$$\Pr\left[|Z - \mathbf{E}[Z]| \geq \frac{\mathbf{E}[Z]}{4}\right] \leq \frac{16\mathbf{Var}[Z]}{\mathbf{E}[Z]^2}$$
$$\leq \frac{8s\|q\|_2 \left\|\sum_{j=1}^{s} \vec{\mathbf{e}}_j\right\|_4^2}{\left\|\sum_{j=1}^{s} \vec{\mathbf{e}}_j\right\|_2^4} + \frac{8\left\|\sum_{j=1}^{s} \vec{\mathbf{p}}_j\right\|_2^2}{\left\|\sum_{j=1}^{s} \vec{\mathbf{e}}_j\right\|_2^4} + \frac{4\left\|\sum_{j=1}^{s} \vec{\mathbf{e}}_j\right\|_3^3}{\left\|\sum_{j=1}^{s} \vec{\mathbf{e}}_j\right\|_2^4}.$$

Note that the right hand side of the above inequality is identical to the right hand side of Inequality (10) that appears in the probability calculation in the proof of Theorem 9. Using the identical bounds given there, we arrive at the following inequality:

$$\Pr\left[|Z - \mathbf{E}[Z]| \geq \frac{\mathbf{E}[Z]}{4}\right] \leq C\left(\frac{n\|q\|_2}{s\epsilon^2} + \left(\frac{n\|q\|}{s\epsilon^2}\right)^2 + \left(\frac{1}{s\|q\|_2}\right)^2 + \frac{\sqrt{n}}{s\,\epsilon}\right)$$

for some constant C. Note that if we let $s = c_1 n\|q\|_2/\epsilon^2$ for a sufficiently large constant c_1 and use the fact that $\|q\|_2 \geq 1/\sqrt{n}$, we have that the above probability is smaller than $1/3$. Again using the randomized flattening procedure of Section 5.2, we see that we can let $s = c_1 n/(\epsilon^2\sqrt{k})$. Hence with probability at least $2/3$, we know $Z \geq 3s^2\epsilon^2/(4n)$ in the soundness case so we reject with probability at least $2/3$, as desired. ◀

5.2 Randomized Flattening Procedure

in this section, we present our randomized flattening procedure. Let k be some fixed parameter (which is an input to **Closeness-Tester**). We show that we can assume that $\|q\|_2^2 = O(1/k)$ for Section 5 without loss of generality where q is the distribution that we have i.i.d. sample access to. This procedure is similar to the one used in [16] for single distribution closeness testing.

First, suppose that we draw $\mathrm{Poi}(k)$ i.i.d. samples from q. Then for each $x \in [n]$, define b_x to be the number of instances of element $x \in [n]$ that we see among these samples plus 1. Note the resemblance between this definition and the one given in the flattening procedure for identity testing in Section 4.2. Now given a sample x from a distribution p over $[n]$, we can get a sample from the "flattened" distribution p' over

$$\mathcal{D} = \{(x, y) \mid x \in [n], y \in [b_x]\}$$

by drawing an element from $y \in [b_x]$ uniformly at random and creating the tuple (x, y). This is the flattening procedure that we use for our generalized version closeness testing. Note that the probability mass over $[n]$ placed by p gets "flattened" to be a probability distribution over $\mathcal{D}$, hence the name. The size of this new domain is $n + k$. We can calculate that this procedure preserves the ℓ_1-distance:

$$\|q' - p'\|_1 = \sum_{x\in[n]} \sum_{y\in[b_x]} \frac{|q(x) - p(x)|}{|b_x|} = \sum_{x\in[n]} |q(x) - p(x)| = \|q - p\|_1.$$

Furthermore,

$$\mathbf{E}[\|q'\|_2^2] = \mathbf{E}\left[\sum_{x\in[n]} \sum_{y\in[b_x]} \frac{q(x)^2}{b_x^2}\right] \leq \sum_{x\in[n]} q(x)^2 \mathbf{E}\left[1/b_x\right].$$

By the poissonization method, we know that b_x is distributed as $1+Z$ where Z is a $\mathrm{Poi}(kq(x))$ random variable. Therefore, similar to [16], we have:

$$\mathbf{E}\left[1/(Z+1)\right] = \mathbf{E}\left[\int_0^1 s^z \, ds\right] = \int_0^1 \mathbf{E}[s^z] \, ds = \int_0^1 e^{kq(x)(s-1)} \, ds \leq \frac{1}{kq(x)}$$

where we have used the probability generating function for a Poisson random variable. This gives us

$$\mathbf{E}[\|q'\|_2^2] \leq \sum_{x\in[n]} \frac{q(x)}{k} \leq \frac{1}{k}.$$

Hence by Markov's inequality, we can say that $\|q'\|_2^2 = O(1/k)$ holds with an arbitrarily large, constant probability. Now whenever we get a sample over $[n]$, we can use this flattening procedure to draw a sample over $\mathcal{D}$. Furthermore, by using this flattening procedure to draw samples from a slightly larger domain, we can assume that $\|q\|_2^2 = O(1/k)$ in Section 5 at the expense of losing a negligible factor in our error probability.

Note that the size of the larger domain is $O(n+k) = O(n)$ if we pick $k \leq n$. Therefore, combining with Theorem 16, we show that we can perform closeness testing with multiple sources by using

$$O\left(k + \frac{n\|q\|_2}{\epsilon^2}\right) = O\left(k + \frac{n}{\epsilon^2\sqrt{k}}\right) = O\left(\frac{n^{2/3}}{\epsilon^{4/3}} + \frac{\sqrt{n}}{\epsilon^2}\right)$$

samples after optimizing the value of k. This sample complexity is optimal since a matching lower bound holds for the single distribution closeness testing setting.

5.3 Proof of Lemma 19

▶ **Lemma 19.** *Let $\{p_i\}_{i=1}^s$ be distributions over $[n]$ that satisfy the structural condition given in Definition 1. Suppose we draw* $\mathrm{Poi}(1)$ *samples from each p_i and* $\mathrm{Poi}(s)$ *samples from q and let T_x be the number of times we see element $x \in [n]$ among the samples among the p_i, and let Y_x be the number of times we see element $x \in [n]$ among the samples from q. Let $Z = \sum_{x\in[n]}(T_x - y_x)^2 - T_x - Y_x$. Then*

$$\mathbf{E}[Z] = \|\vec{\mathbf{e}}_1 + \cdots + \vec{\mathbf{e}}_s\|_2^2$$

where $\vec{\mathbf{e}}_j \in \mathbb{R}^n$ has coordinates $|q(x) - p_j(x)|$.

Proof. Note that T_x is a Poisson random variable with parameter $\lambda_x = \sum_{j=1}^s p_j(x)$ and Y_x is a Poisson random variable with parameter $sq(x)$. Let

$$Z_x = (T_x - Y_x)^2 - T_x - Y_x.$$

We can compute that

$$\begin{aligned}\mathbf{E}[Z_x] &= \mathbf{E}[T_x^2] - \mathbf{E}[T_x] + \mathbf{E}[Y_x^2] - \mathbf{E}[Y_x] - 2\mathbf{E}[Y_x]\mathbf{E}[T_x] \\ &= \lambda_x^2 - 2sq(x)\lambda_x + s^2q(x)^2.\end{aligned}$$

This is the same as the expected value of the variable Z_x in Lemma 12. Therefore, the same computations hold and we arrive at

$$\mathbf{E}[Z] = \sum_{x\in[n]} \mathbf{E}[Z_x] = \|\vec{\mathbf{e}}_1 + \cdots + \vec{\mathbf{e}}_s\|_2^2,$$

as desired. ◀

5.4 Proof of Lemma 20

▶ **Lemma 20.** *Let $\{p_i\}_{i=1}^s$ be distributions over $[n]$ that satisfy the structural condition given in Definition 1. Suppose we draw* $\mathrm{Poi}(1)$ *samples from each p_i and* $\mathrm{Poi}(s)$ *samples from q and let T_x be the number of times we see element $x \in [n]$ among the samples among the p_i, and let Y_x be the number of times we see element $x \in [n]$ among the samples from q. Let $Z = \sum_{x\in[n]}(T_x - y_x)^2 - T_x - Y_x$. Then*

$$\mathrm{Var}(Z) \le 8s\|q\|_2 \left\|\sum_{j=1}^s \vec{\mathbf{e}}_j\right\|_4^2 + 8\left\|\sum_{j=1}^s \vec{\mathbf{p}}_j\right\|_2^2 + 4\left\|\sum_{j=1}^s \vec{\mathbf{e}}_j\right\|_3^3$$

where $\vec{\mathbf{e}}_j \in \mathbb{R}^n$ has coordinates $|q(x) - p_j(x)|$ and $\vec{\mathbf{p}}_j \in \mathbb{R}^n$ has coordinates $p_j(x)$.

Proof. Define

$$Z_x = (T_x - Y_x)^2 - T_x - Y_x.$$

Due to the independence of T_x and Y_x, we have

$$\mathrm{Var}(Z) = \sum_{x\in[n]} \mathrm{Var}(Z_x) = \sum_{x\in[n]} \mathbf{E}[Z_x^2] - \mathbf{E}[Z_x]^2.$$

Noting that T_x is Poisson with parameter $\lambda_x = \sum_{j=1}^s p_j(x)$ and Y_x is Poisson with parameter $sq(x)$, we can compute that

$$\begin{aligned}\mathbf{E}[Z_x^2] = {} & \lambda_x^4 + 4\lambda_x^3(1 - sq(x)) + 2\lambda_x^2(3s^2q(x)^2 - 2sq(x) + 1) \\ & + 4sq(x)\lambda_x(1 - sq(x) - s^2q(x)^2) + s^4q(x)^4 + 4s^3q(x)^3 + 2s^2q(x)^2.\end{aligned}$$

Using the formula for λ_x^k for $k \in \{1, 2, 3, 4\}$ given in Appendix A and simplifying, we arrive at the following expression:

$$\begin{aligned}\sum_{x\in[n]} \mathbf{E}[Z_x^2] = {} & 8s\sum_{j=1}^s \sum_{x\in[n]} q(x)e_j(x)^2 + 8s\sum_{j\neq k}\sum_{x\in[n]} q(x)e_j(x)e_k(x) + 8s^2\|q\|_2^2 \\ & + 8s\sum_{j=1}^s\sum_{x\in[n]}(-1)^{x\in B}q(x)e_j(x) + 2\sum_{j=1}^s\sum_{x\in[n]} e_j(x)^2 + 2\sum_{j\neq k}\sum_{x\in[n]} e_j(x)e_k(x) \\ & + 4\sum_{j=1}^s\sum_{x\in[n]}(-1)^{x\in B}e_j(x)^3 + 12\sum_{j\neq k}\sum_{x\in[n]}(-1)^{x\in B}e_j(x)^2e_k(x) \\ & + 4\sum_{j\neq k\neq \ell}\sum_{x\in[n]}(-1)^{x\in B}e_j(x)e_k(x)e_\ell(x) + \Bigg(\sum_{j=1}^s\sum_{x\in[n]} e_j(x)^4 + 6\sum_{j\neq k\neq \ell} e_j(x)^2e_k(x)e_\ell(x) \\ & + 4\sum_{j\neq k} e_j(x)^3e_k(x) + 3\sum_{j\neq k} e_j(x)^2e_k(x)^2 + \sum_{j\neq k\neq \ell\neq t} e_j(x)e_k(x)e_\ell(x)e_t(x)\Bigg).\end{aligned}$$

Similar to Lemma 13, we have

$$8s\sum_{j=1}^s\sum_{x\in[n]} q(x)e_j(x)^2 + 8s\sum_{j\neq k}\sum_{x\in[n]} q(x)e_j(x)e_k(x) \le 8s\|q\|_2\|\vec{\mathbf{e}}_1 + \cdots + \vec{\mathbf{e}}_s\|_4^2,$$

and

$$8s^2\|q\|_2^2 + 8s\sum_{j=1}^{s}\sum_{x\in[n]}(-1)^{x\in B}q(x)e_j(x) + 2\sum_{j=1}^{s}\sum_{x\in[n]}e_j(x)^2 + 2\sum_{j\neq k}\sum_{x\in[n]}e_j(x)e_k(x)$$
$$\leq 8\|\vec{\mathbf{p}}_1 + \cdots \vec{\mathbf{p}}_s\|_2^2,$$

and finally,

$$4\sum_{j=1}^{s}\sum_{x\in[n]}(-1)^{x\in B}e_j(x)^3 + 12\sum_{j\neq k}\sum_{x\in[n]}(-1)^{x\in B}e_j(x)^2e_k(x)$$
$$+ 4\sum_{j\neq k\neq \ell}\sum_{x\in[n]}(-1)^{x\in B}e_j(x)e_k(x)e_\ell(x) = 4\|\vec{\mathbf{e}}_1 + \cdots + \vec{\mathbf{e}}_s\|_3^3.$$

As in Lemma 13, the last expression inside the parenthesis of $\mathbf{E}[Z_x^2]$ is precisely

$$\sum_{x\in[n]}\left(\sum_{j=1}^{s}e_j(x)\right)^4.$$

Therefore,

$$\sum_{x\in[n]}\mathbf{E}[Z_x^2] \leq 8s\|q\|_2\left\|\sum_{j=1}^{s}\vec{\mathbf{e}}_j\right\|_4^2 + 8\left\|\sum_{j=1}^{s}\vec{\mathbf{p}}_j\right\|_2^2 + 4\left\|\sum_{j=1}^{s}\vec{\mathbf{e}}_j\right\|_3^3 + \sum_{x\in[n]}\left(\sum_{j=1}^{s}e_j(x)\right)^4.$$

The same calculation as in Lemma 13 gives us

$$\sum_{x\in[n]}\mathbf{E}[Z_x]^2 = \sum_{x\in[n]}\left(\sum_{j=1}^{s}e_j(x)^2 + \sum_{j\neq k}e_j(x)e_k(x)\right)^2 = \sum_{x\in[n]}\left(\sum_{j=1}^{s}e_j(x)\right)^4$$

so altogether,

$$\mathrm{Var}(Z) \leq 8s\|q\|_2\left\|\sum_{j=1}^{s}\vec{\mathbf{e}}_j\right\|_4^2 + 8\left\|\sum_{j=1}^{s}\vec{\mathbf{p}}_j\right\|_2^2 + 4\left\|\sum_{j=1}^{s}\vec{\mathbf{e}}_j\right\|_3^3. \tag{13}$$

◀

6 Failure of de Finetti's Theorem with Sublinear Number of Samples

in this section, we prove Theorem 2. Recall the statement of Theorem 2.

▶ **Theorem 2.** *Let $s = O(\sqrt{n})$ be the number of samples required by Algorithm 1 for $\epsilon = 1/3$. There exists an exchangeable sequence $X_1, \ldots, X_s$ such that X_i is drawn from distribution q_i which are all supported in $[n]$ and satisfy $\|q_i - \mathcal{U}_n\|_1 \geq 1/3$ for all i. Furthermore, $\{q_i\}_{i=1}^s$ all satisfy the structural condition given in Definition 1 with $q = \mathcal{U}_n$. Let P denote the distribution of the sequence $X_1, \ldots, X_s$. Then P is $\Omega(1)$-far in ℓ_1-distance from any mixture of product distributions.*

Note that an algorithm can turn samples $Y_1, \cdots, Y_s$ into an exchangeable sequence $X_1, \cdots, X_s$ by permuting randomly. In this section, we given an example of samples $Y_1, \cdots, Y_s$ such that after permuting them to turn them into an exchangeable sequence

$X_1, \cdots, X_s$, the exchangeable sequence is "far" from a mixture of product distributions. The main idea behind Theorem 2 is based on Proposition 31 in [13]. Essentially, Diaconis and Freedman show in [13] that a Polya's urn process generates an exchangeable sequence that is far from the mixture of any product distributions. This example does not quite work in our case since we would like the *structural condition* to hold. Therefore, we adapt the Polya's urn idea by partitioning our domain into a "large" set and a "small" set. We then apply a Polya's urn type process on the small set so that no collisions can happen on it. This results in an event $\mathcal{E}_1$ which we use to lower bound the distance from the distribution of our exchangeable sequence to any mixture of product distributions. However, we need an additional event $\mathcal{E}_2$ to deal with the large set. Combining these two events allows us to prove Theorem 2. This overview is formalized below.

Proof of Theorem 2. Let $\epsilon = 1/3$ and let s be the number of samples required by our uniformity tester in Algorithm 1 for this value of ϵ. In particular, $s^2 = Cn$ for some constant $C > 10$ and define δ as $\delta = 1/C$. We now construct distributions $\{q_i\}_{i=1}^s$ as follows:

$$q_i(x) = \begin{cases} 1 - \delta/20 - s/n, & x = 1 \\ 1/n, & x \in \{2, \cdots, s+1\} \setminus \{i+1\} \\ \delta/20, & x = i+1. \end{cases}$$

Note that all distributions are supported only on $\{1, \cdots, s+1\}$. We can check that for large enough n, we have $\|q_i - \mathcal{U}_n\|_1 \geq 1/3$ for all i. We then draw sample Y_i independently from q_i. Note that $A = \{1, \cdots, s+1\}$ and $B = [n] \setminus \{1, \cdots, s+1\}$ for the *structural condition* in Definition 1. In other words, all the distributions are larger than uniform on A and smaller than uniform on B so $\{q_i\}_{i=1}^s$ satisfy the conditions of the theorem.

Now let $\{X_i\}_{i=1}^s$ be an exchangeable sequence derived from $\{Y_i\}_{i=1}^s$ (for example, by permuting them randomly). Let p be the distribution of $(X_1, \cdots, X_s)$. Consider the following two events:

$\mathcal{E}_1 =$ Event that $X_i = X_j \in \{2, \cdots, s+1\}$ for some $i \neq j$

$\mathcal{E}_2 =$ Event that at least $s(1 - \delta^2/2)$ of the X_i's are equal to 1.

We first compute $p(\mathcal{E}_1)$. Note that for any $i \neq j$, we have

$$\mathbb{P}(X_i = X_j \in \{2, \cdots, s+1\}) \leq \frac{\delta}{10n} + \frac{s}{n^2}.$$

Therefore by a union bound, the probability that there exists some $i \neq j$ such that $\mathcal{E}_1$ holds is at most

$$\frac{C\delta}{10} + \frac{s^3}{n^2} \leq \frac{1}{9}$$

for sufficiently large n. We now compute $p(\mathcal{E}_2)$. We know that $\mathbb{P}(X_i = 1) \leq 1 - \delta/20$ for all i. Therefore, the number of 1's among the X_i's is sum of s Bernoulli random variables with parameters at most $1 - \delta/20$ and hence, the expected number of 1's is at most $s(1 - \delta/20)$. Then by a Chernoff bound, we have $\mathbb{P}(\mathcal{E}_2) \leq 1/100$ if we take n (and therefore s) sufficiently large.

Now for a distribution p_k over $[n]$, we let p_k^s be the distribution of s independent picks from p_k. Let $M = \sum_k w_k p_k^s$ be any mixture of product distributions p_k^s. We wish to show that $\|p - M\|_1 = \Omega(1)$ for any M. Note that all of the $\{q_i\}_{i=1}^s$ are only supported on $\{1, \cdots, s+1\}$. Therefore, we can assume without loss of generality that the p_k is also supported only on $\{1, \cdots, s+1\}$ for all k. Now consider a single p_k^s. We consider two cases.

Case 1: $p_k(\{2, \cdots, s+1\}) \geq \delta^2$. In this case, let Z be the number of collisions among elements in $\{2, \cdots, s+1\}$ if we draw s independent samples from p_k. We have

$$\mathbf{E}[Z] = \binom{s}{2} \sum_{i \in \{2,\cdots,s\}} p_k(i)^2.$$

Define $\sum_{i\in\{2,\cdots,s\}} p_k(i)^2 = \|\widetilde{p_k}\|_2^2$. Using standard calculations as in the single distribution uniformity testing case, see [23, 6], we can compute

$$\mathbf{Var}[Z] \leq 4\left(\binom{s}{2}\|\widetilde{p_k}\|_2^2\right)^{3/2}.$$

Hence by Chebyshev's inequality,

$$\mathbb{P}(Z=0) \leq \mathbb{P}(|Z - \mathbf{E}[Z]| \geq \mathbf{E}[Z]) \leq \frac{4\binom{s}{2}^{3/2}\|\widetilde{p_k}\|_2^3}{\binom{s}{2}^2\|\widetilde{p_k}\|_2^4} \leq \frac{C'}{s\|\widetilde{p_k}\|_2}$$

for some absolute constant C'. Now by Cauchy Schwarz,

$$\|\widetilde{p_k}\|_2 \geq \frac{\delta^2}{\sqrt{s}}$$

so we have $\mathbb{P}(Z=0) \leq 1/100$ for sufficiently large n. Therefore, $p_k^s(\mathcal{E}_1) \geq 99/100$.

Case 2: $p_k(\{2, \cdots, s+1\}) < \delta^2$. In this case, we have $p_k(1) \geq 1 - \delta^2$. Therefore, if we draw s samples from p_k, the number of 1's that we will see is at least $s(1-\delta^2)$ in expectation. By Chernoff, the probability that we see less than $s(1-\delta^2/2)$ number of 1's is at at most $1/100$ for sufficiently large n. Hence, $p_k^s(\mathcal{E}_2) \geq 99/100$.
Now note that

$$\|p - M\|_1 = 2\|p - M\|_{TV} \geq |P(\mathcal{E}_1) - M(\mathcal{E}_1)| + |p(\mathcal{E}_2) - M(\mathcal{E}_2)|.$$

We have

$$|p(\mathcal{E}_1) - M(\mathcal{E}_1)| \geq \sum_{k|p_k \in \text{Case 1}} w_k p_k^s(\mathcal{E}_1) - \frac{1}{9} \geq \frac{99}{100} \sum_{k|p_k\in\text{Case 1}} w_k - \frac{1}{9}.$$

Similarly,

$$|p(\mathcal{E}_2) - M(\mathcal{E}_2)| \geq \sum_{k|p_k \in \text{Case 2}} w_k p_k^s(\mathcal{E}_2) - \frac{1}{100} \geq \frac{99}{100} \sum_{k|p_k\in\text{Case 2}} w_k - \frac{1}{100}.$$

Hence we have that

$$\|p - M\|_1 \geq \frac{99}{100}\left(\sum_k w_k\right) - \left(\frac{1}{9} + \frac{1}{100}\right) \geq \Omega(1).$$

Since M was arbitrary, we are done. Hence, p is $\Omega(1)$-far from any mixture of product distributions. ◀

References

1 Jayadev Acharya, Constantinos Daskalakis, and Gautam Kamath. Optimal Testing for Properties of Distributions. In *NIPS*, pages 3591–3599, 2015.

2 Maryam Aliakbarpour, Eric Blais, and Ronitt Rubinfeld. Learning and Testing Junta Distributions. In *COLT*, pages 19–46, 2016.

3 Maryam Aliakbarpour, Ilias Diakonikolas, Daniel Kane, and Ronitt Rubinfeld. Private Testing of Distributions via Sample Permutations. *To appear in NeurIPS*, 2019. URL: `http://papers.nips.cc/paper/9270-private-testing-of-distributions-via-sample-permutations.pdf`.

4 Tugkan Batu and Clément L. Canonne. Generalized Uniformity Testing. *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 880–889, 2017.

5 Tugkan Batu, Eldar Fischer, Lance Fortnow, Ravi Kumar, Ronitt Rubinfeld, and Patrick White. Testing random variables for independence and identity. In *FOCS*, pages 442–451, 2001.

6 Tugkan Batu, Lance Fortnow, Ronitt Rubinfeld, Warren D. Smith, and Patrick White. Testing That Distributions Are Close. In *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, FOCS '00, pages 259–, Washington, DC, USA, 2000. IEEE Computer Society. URL: `http://dl.acm.org/citation.cfm?id=795666.796548`.

7 Tugkan Batu, Lance Fortnow, Ronitt Rubinfeld, Warren D. Smith, and Patrick White. Testing Closeness of Discrete Distributions. *JACM*, 60(1):4:1–4:25, 2013.

8 Eric Blais, Clément L. Canonne, and Tom Gur. Distribution Testing Lower Bounds via Reductions from Communication Complexity. *ACM Trans. Comput. Theory*, 11(2):6:1–6:37, February 2019. `doi:10.1145/3305270`.

9 Clément L Canonne. A survey on distribution testing: Your data is big. but is it blue? *ECCC*, 22:63, 2015.

10 Siu-on Chan, Ilias Diakonikolas, Paul Valiant, and Gregory Valiant. Optimal Algorithms for Testing Closeness of Discrete Distributions. In *SODA*, pages 1193–1203, 2014.

11 Zdravko Cvetkovski. *Inequalities theorems, techniques and selected problems*. Springer, 2012.

12 Persi Diaconis. Finite forms of de Finetti's theorem on exchangeability. *Synthese*, 36(2):271–281, October 1977. `doi:10.1007/BF00486116`.

13 Persi Diaconis and David Freedman. Finite Exchangeable Sequences. *The Annals of Probability*, 8(4):745–764, 1980. URL: `http://www.jstor.org/stable/2242823`.

14 Ilias Diakonikolas, Themis Gouleakis, J. Peebles, and Eric Price. Collision-based Testers are Optimal for Uniformity and Closeness. *ECCC*, 23:178, 2016.

15 Ilias Diakonikolas, Themis Gouleakis, John Peebles, and Eric Price. Sample-Optimal Identity Testing with High Probability. In *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, pages 41:1–41:14, 2018.

16 Ilias Diakonikolas and Daniel M. Kane. A New Approach for Testing Properties of Discrete Distributions. In *FOCS*, pages 685–694, 2016.

17 Ilias Diakonikolas, Daniel M. Kane, and Vladimir Nikishkin. Testing Identity of Structured Distributions. In *SODA*, pages 1841–1854, 2015.

18 Oded Goldreich. The uniform distribution is complete with respect to testing identity to a fixed distribution. *Electronic Colloquium on Computational Complexity (ECCC)*, 23:15, 2016.

19 Oded Goldreich. The uniform distribution is complete with respect to testing identity to a fixed distribution. *ECCC*, 23, 2016. URL: `http://www.wisdom.weizmann.ac.il/~oded/R2/dr.pdf`.

20 Oded Goldreich. *Introduction to Property Testing*. Cambridge University Press, 2017. URL: `http://www.wisdom.weizmann.ac.il/~oded/pt-intro.html`.

21 Oded Goldreich, Shafi Goldwasser, and Dana Ron. Property testing and its connection to learning and approximation. *JACM*, 45:653–750, 1998.

22 Oded Goldreich and Dana Ron. On testing expansion in bounded-degree graphs. In *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation*, pages 68–75. Springer, 2011.

23 Oded Goldreich and Dana Ron. *On Testing Expansion in Bounded-Degree Graphs*, pages 68–75. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. doi:10.1007/978-3-642-22670-0_9.

24 Olav Kallenberg. *Probabilistic Symmetries and Invariance Principles*. Springer New York, 2005.

25 Erich L. Lehmann and Joseph P. Romano. *Testing statistical hypotheses*. Springer Texts in Statistics. Springer, 2005.

26 Reut Levi, Dana Ron, and Ronitt Rubinfeld. Testing Properties of Collections of Distributions. *Theory of Computing*, 9(8):295–347, 2013.

27 Jerzy Neyman and Egon S. Pearson. On the Problem of the Most Efficient Tests of Statistical Hypotheses. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 231(694-706):289–337, 1933. doi:10.1098/rsta.1933.0009.

28 Liam Paninski. A coincidence-based test for uniformity given very sparsely-sampled discrete data. *IEEE TOIT*, 54:4750–4755, 2008.

29 Sofya Raskhodnikova, Dana Ron, Amir Shpilka, and Adam Smith. Strong Lower Bounds for Approximating Distribution Support Size and the Distinct Elements Problem. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*, pages 559–569, October 2007. doi:10.1109/FOCS.2007.47.

30 Ronitt Rubinfeld. Taming big probability distributions. *XRDS*, 19(1):24–28, 2012.

31 Kevin Tian, Weihao Kong, and Gregory Valiant. Learning Populations of Parameters. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5778–5787. Curran Associates, Inc., 2017. URL: http://papers.nips.cc/paper/7160-learning-populations-of-parameters.pdf.

32 Gregory Valiant and Paul Valiant. An Automatic Inequality Prover and Instance Optimal Identity Testing. *SICOMP*, 46(1):429–455, 2017.

33 Gregory Valiant and Paul Valiant. Estimating the Unseen: Improved Estimators for Entropy and Other Properties. *JACM*, 64(6):37:1–37:41, 2017.

34 Paul Valiant. Testing Symmetric Properties of Distributions. In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, STOC '08, pages 383–392. ACM, 2008.

35 Paul Valiant. Testing symmetric properties of distributions. In *STOC*, pages 383–392, 2008.

36 Ramya Korlakai Vinayak, Weihao Kong, Gregory Valiant, and Sham Kakade. Maximum Likelihood Estimation for Learning Populations of Parameters. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, pages 6448–6457. PMLR, 2019.

37 Yihong Wu and Pengkun Yang. Chebyshev polynomials, moment matching, and optimal estimation of the unseen. *arXiv preprint*, 2016. arXiv:1504.01227v2.

A Moment Calculations of Sections 4 and 5

In this section, we calculate the moments of random variables that appear in Section 4 and Section 5. Suppose we have distributions q and $p_1, \cdots, p_s$ such that there exist subsets A and $B = [n] \setminus A$ with the property that $p_j(x) \geq q(x)$ for all $x \in A$ and all j and $p_j(x) \leq q(x)$ for all $x \in B$ and all j. Now, let T_x be a Poisson random variable with parameter

$$\lambda_x = \sum_{j=1}^{s} p_j(x).$$

We compute $\sum_{x\in[n]} \lambda_x^k$ for $k \in \{1,2,3,4\}$. We note that

$$\sum_{x\in[n]} \lambda_x = \sum_{j=1}^{s} \sum_{x\in[n]} (q(x) + (-1)^{x\in B} e_j(x)) \tag{14}$$

We now compute $\sum_{x\in[n]} \lambda_x^2$. We use the notation $(-1)^{x\in B}$ as follows:

$$(-1)^{x\in B} = \begin{cases} -1 & \text{if } x \in B \\ 1 & \text{if } x \notin B \end{cases}.$$

Note that:

$$\sum_{x\in[n]} \lambda_x^2 = \sum_{x\in[n]} \left(\sum_{j=1}^{s} p_j(x)\right)^2 = \sum_{x\in[n]} \left(\sum_{j=1}^{s} p_j(x)^2 + \sum_{j\neq k} p_j(x)p_k(x)\right).$$

For fixed j and k, we can compute that:

$$p_j(x)^2 = q(x)^2 + 2(-1)^{x\in B} q(x)e_j(x) + e_j(x)^2\,,$$

and we have:

$$p_j(x)p_k(x) = q(x)^2 + (-1)^{x\in B}(q(x)e_k(x) + q(x)e_j(x)) + e_j(x)e_k(x)\,.$$

Putting everything together, we obtain:

$$\begin{aligned}\sum_{x\in[n]} \lambda_x^2 = s^2||q||_2^2 + 2s \sum_{j=1}^{s} \sum_{x\in[n]} (-1)^{x\in B} q(x)e_j(x) \\ + \sum_{j=1}^{s} \sum_{x\in[n]} e_j(x)^2 + \sum_{j\neq k} \sum_{x\in[n]} e_j(x)e_k(x).\end{aligned} \tag{15}$$

We now compute $\sum_{x\in[n]} \lambda_x^3$. For a fixed x, we have:

$$\lambda_x^3 = \left(\sum_{j=1}^{s} p_j(x)\right)^3 = \sum_{j=1}^{s} p_j(x)^3 + 3\sum_{j\neq k} p_j(x)^2 p_k(x) + \sum_{j\neq k\neq \ell} p_j(x)p_k(x)p_\ell(x).$$

Now for a fixed j,

$$\begin{aligned}&p_j(x)^3 = (q(x) + (-1)^{x\in B} e_j(x))^3 \\ &= q(x)^3 + 3(-1)^{x\in B} q(x)^2 e_j(x) + 3q(x)e_j(x)^2 + (-1)^{x\in B} e_j(x)^3,\end{aligned}$$

while for fixed $j \neq k$,

$$\begin{aligned}p_j(x)^2 p_k(x) &= (q(x) + (-1)^{x\in B} e_j(x))^2 (q(x) + (-1)^{x\in B} e_k(x)) \\ &= q(x)^3 + (-1)^{x\in B} q(x)^2 (2e_j(x) + e_k(x)) + (-1)^{x\in B} e_j(x)^2 e_k(x) \\ &\quad + q(x)(e_j(x)^2 + 2e_j(x)e_k(x) + e_k^2(x)),\end{aligned}$$

and finally for $j \neq k \neq \ell$, we have:

$$\begin{aligned}&p_j(x)p_k(x)p_\ell(x) = (q(x) + (-1)^{x\in B} e_j(x))(q(x) + (-1)^{x\in B} e_k(x))(q(x) + (-1)^{x\in B} e_\ell(x)) \\ &= q(x)^3 + (-1)^{x\in B} q(x)^2 (e_j(x) + e_k(x) + e_\ell(x)) + (-1)^{x\in B} e_j(x)e_k(x)e_\ell(x) \\ &+ q(x)(e_j(x)e_k(x) + e_k(x)e_\ell(x) + e_j(x)e_\ell(x)).\end{aligned}$$

Putting everything together, we have

$$\begin{aligned}\sum_{x\in[n]}\lambda_x^3 = s^3\|q\|_3^3 + 3s^2\sum_{j=1}^{s}\sum_{x\in[n]}(-1)^{x\in B}q(x)^2e_j(x) + 3\sum_{j=1}^{s}\sum_{x\in[n]}q(x)e_j(x)^2 \\ + 3s\sum_{j\neq k}\sum_{x\in[n]}q(x)e_j(x)e_k(x) + 3\sum_{j\neq k}\sum_{x\in[n]}(-1)^{x\in B}e_j(x)^2e_k(x) \\ + \sum_{j=1}^{s}\sum_{x\in[n]}(-1)^{x\in B}e_j(x)^3 + \sum_{j\neq k\neq \ell}\sum_{x\in[n]}(-1)^{x\in B}e_j(x)e_k(x)e_\ell(x). \qquad (16)\end{aligned}$$

We now compute $\sum_{x\in[n]}\lambda_x^4$. We have

$$\begin{aligned}\lambda_x = \left(\sum_{j=1}^{s}p_j(x)\right)^4 = \sum_{j=1}^{s}p_j(x)^4 + 3\sum_{j\neq k}p_j(x)^2p_k(x)^2 + 4\sum_{j\neq k}p_j(x)^3p_k(x) \\ + 6\sum_{j\neq k\neq \ell}p_j(x)^2p_k(x)p_\ell(x) + \sum_{j\neq k\neq \ell\neq t}p_j(x)p_k(x)p_\ell(x)p_t(x).\end{aligned}$$

We first analyze $p_j(x)^4$ for a fixed j. We have:

$$\begin{aligned}p_j(x)^4 = (q(x)+(-1)^{x\in B}e_j(x))^4 = q(x)^4 + 4(-1)^{x\in B}q(x)^3e_j(x) + 6q(x)^2e_j(x)^2 \\ + 4(-1)^{x\in B}q(x)e_j(x)^3 + e_j(x)^4.\end{aligned}$$

Then for fixed $j\neq k$, we have:

$$\begin{aligned}p_j(x)^2p_k(x)^2 &= (q(x)+(-1)^{x\in B}e_j(x))^2(q(x)+(-1)^{x\in B}e_k(x))^2 \\ &= q(x)^4 + (-1)^{x\in B}q(x)^3(2e_j(x)+2e_k(x)) \\ &+ q(x^2(e_j(x)^2 + 4e_j(x)e_k(x) + e_k(x)^2) \\ &+ (-1)^{x\in B}q(x)(2e_j(x)^2e_k(x) + 2e_j(x)e_k(x)^2) + e_j(x)^2e_k(x)^2\,,\end{aligned}$$

and, we can get:

$$\begin{aligned}p_j(x)^3p_k(x) &= (q(x)+(-1)^{x\in B}e_j(x))^3(q(x)+(-1)^{x\in B}e_k(x)) \\ &= q(x)^4 + (-1)^{x\in B}q(x)^3(e_j(x)+e_k(x)) + q(x)^2(3e_j(x)^2 + 3e_j(x)e_k(x)) \\ &+ (-1)^{x\in B}q(x)(e_j(x)^3 + 3e_j(x)^2e_k(x)) + e_j(x)^3e_k(x).\end{aligned}$$

Furthermore, for fixed $j\neq k\neq \ell$, we have:

$$\begin{aligned}&p_j(x)^2p_k(x)p_\ell(x) \\ &= (q(x)+(-1)^{x\in B}e_j(x))^2(q(x)+(-1)^{x\in B}e_k(x))(q(x)+(-1)^{x\in B}e_\ell(x)) \\ &= q(x)^4 + (-1)^{x\in B}q(x)^3(2e_j(x)+e_\ell(x)+e_k(x)) \\ &+ q(x)^2(e_j(x)^2 + 2e_j(x)e_k(x) + 2e_j(x)e_\ell(x) + e_k(x)e_\ell(x)) \\ &+ (-1)^{x\in B}q(x)(e_j(x)^2e_\ell(x) + e_j(x)^2e_k(x) + 2e_j(x)e_k(x)e_\ell(x)) \\ &+ e_j(x)^2e_k(x)e_\ell(x).\end{aligned}$$

Finally, for fixed $j \neq k \neq \ell \neq t$, we have:

$$
\begin{aligned}
&p_j(x)p_k(x)p_\ell(x)p_t(x)\\
&= (q(x) + (-1)^{x\in B}e_j(x))(q(x) + (-1)^{x\in B}e_k(x))\\
&\cdot (q(x) + (-1)^{x\in B}e_\ell(x))(q(x) + (-1)^{x\in B}e_t(x))\\
&= q(x)^4 + (-1)^{x\in B}q(x)(e_j(x) + e_k(x) + e_\ell(x) + e_t(x))\\
&+ q(x)^2(e_j(x)e_k(x) + e_j(x)e_\ell(x) + e_j(x)e_t(x) + e_k(x)e_\ell(x) + e_k(x)e_t(x) + e_\ell(x)e_t(x))\\
&+ (-1)^{x\in B}q(x)(e_j(x)e_k(x)e_\ell(x) + e_j(x)e_k(x)e_t(x) + e_j(x)e_t(x)e_\ell(x) + e_k(x)e_\ell(x)e_t(x))\\
&+ e_j(x)e_k(x)e_\ell(x)e_t(x).
\end{aligned}
$$

Altogether, we have:

$$
\begin{aligned}
&\sum_{x\in[n]} \lambda_x^4\\
&= s^4\|q\|_4^4 + \sum_{x\in[n]}\sum_{j=1}^{s} e_j(x)^4 + 6s^2\sum_{j=1}^{s}\sum_{x\in[n]} q(x)^2e_j(x)^2 + 4s\sum_{j=1}^{s}\sum_{x\in[n]}(-1)^{x\in B}q(x)e_j(x)^3\\
&+ 4s^3\sum_{j=1}^{s}\sum_{x\in[n]}(-1)^{x\in B}q(x)^3e_j(x) + 6s^2\sum_{j\neq k}\sum_{x\in[n]} q(x)^2e_j(x)e_k(x)\\
&+ 12s\sum_{j\neq k}\sum_{x\in[n]}(-1)^{x\in B}q(x)e_j(x)^2e_k(x) + 4s\sum_{j\neq k\neq \ell}\sum_{x\in[n]}(-1)^{x\in B}q(x)e_j(x)e_k(x)e_\ell(x)\\
&+ 6\sum_{j\neq k\neq \ell} e_j(x)^2e_k(x)e_\ell(x) + 4\sum_{j\neq k}\sum_{x\in[n]} e_j(x)^3e_k(x) + 3\sum_{j\neq k}\sum_{x\in[n]} e_j(x)^2e_k(x)^2\\
&+ \sum_{j\neq k\neq \ell\neq t} e_j(x)e_k(x)e_\ell(x)e_t(x). \qquad (17)
\end{aligned}
$$

Beyond Natural Proofs: Hardness Magnification and Locality

Lijie Chen
Massachusetts Institute of Technology, Cambridge, MA, USA
lijieche@mit.edu

Shuichi Hirahara
National Institute of Informatics, Tokyo, Japan
s_hirahara@nii.ac.jp

Igor C. Oliveira
University of Warwick, United Kingdom
igor.oliveira@warwick.ac.uk

Ján Pich
University of Oxford, United Kingdom
jan.pich@cs.ox.ac.uk

Ninad Rajgopal
University of Oxford, United Kingdom
ninad.rajgopal@cs.ox.ac.uk

Rahul Santhanam
University of Oxford, United Kingdom
rahul.santhanam@cs.ox.ac.uk

Abstract

Hardness magnification reduces major complexity separations (such as $\mathsf{EXP} \not\subseteq \mathsf{NC}^1$) to proving lower bounds for some natural problem Q against weak circuit models. Several recent works [42, 36, 13, 39, 12, 38, 10] have established results of this form. In the most intriguing cases, the required lower bound is known for problems that appear to be significantly easier than Q, while Q itself is susceptible to lower bounds but these are not yet sufficient for magnification.

In this work, we provide more examples of this phenomenon, and investigate the prospects of proving new lower bounds using this approach. In particular, we consider the following essential questions associated with the hardness magnification program:

- *Does hardness magnification avoid the natural proofs barrier of Razborov and Rudich* [46]*?*
- *Can we adapt known lower bound techniques to establish the desired lower bound for* Q*?*

We establish that some instantiations of hardness magnification overcome the natural proofs barrier in the following sense: slightly superlinear-size circuit lower bounds for certain versions of the minimum circuit size problem MCSP imply the non-existence of natural proofs. As a corollary of our result, we show that certain magnification theorems not only imply strong worst-case circuit lower bounds but also rule out the existence of efficient learning algorithms.

Hardness magnification might sidestep natural proofs, but we identify a source of difficulty when trying to adapt existing lower bound techniques to prove strong lower bounds via magnification. This is captured by a *locality barrier*: existing magnification theorems *unconditionally* show that the problems Q considered above admit highly efficient circuits extended with small fan-in oracle gates, while lower bound techniques against weak circuit models quite often easily extend to circuits containing such oracles. This explains why direct adaptations of certain lower bounds are unlikely to yield strong complexity separations via hardness magnification.

2012 ACM Subject Classification Theory of computation → Computational complexity and cryptography

Keywords and phrases Hardness Magnification, Natural Proofs, Minimum Circuit Size Problem, Circuit Lower Bounds

11th Innovations in Theoretical Computer Science Conference (ITCS 2020).
Editor: Thomas Vidick; Article No. 70; pp. 70:1–70:48

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.70

Acknowledgements Part of this work was completed while some of the authors were visiting the Simons Institute for the Theory of Computing. We are grateful to the Simons Institute for their support. This work was supported in part by the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2014)/ERC Grant Agreement no. 615075. Ján Pich was supported in part by Grant 19-05497S of GA ČR. Lijie Chen is supported by NSF CCF-1741615 and a Google Faculty Research Award. Igor C. Oliveira was supported in part by a Royal Society University Research Fellowship.[1]

1 Introduction

Proving circuit size lower bounds for explicit Boolean functions is a central problem in Complexity Theory. Unfortunately, it is also notoriously hard, and arguments ruling out a wide range of approaches have been discovered. The most prominent of them is the *natural proofs barrier* of Razborov and Rudich [46].

A candidate approach for overcoming this barrier was investigated recently by Oliveira and Santhanam [42]. *Hardness Magnification* identifies situations where strong circuit lower bounds for explicit Boolean functions (e.g. $\mathsf{NP} \not\subseteq \mathsf{P/poly}$) follow from much weaker (e.g. slightly superlinear) lower bounds for specific natural problems. As discussed in [42], in some cases the lower bounds required for magnification are already known for explicit problems, but not yet for the problem for which the magnification theorem holds. This approach to lower bounds has attracted the interest of several researchers, and a number of recent works have proved magnification results [36, 13, 39, 12, 38, 10] (see also [50, 3, 35, 37] for related previous work). We provide a concise review of existing results in Appendix A.1.

In this work, we are interested in understanding the prospects of proving new lower bounds using hardness magnification, including potential barriers.

1.1 Hardness Magnification Frontiers

While hardness magnification is a broad phenomenon, its most promising instantiations seem to occur in the setting of circuit classes such as NC^1. The potential of hardness magnification stems from establishing the following scenario.

> **HM Frontier:** There is a natural problem Q and a computational model $\mathcal{C}$ such that:
>
> 1. (*Magnification*) $Q \notin \mathcal{C}$ implies $\mathsf{NP} \not\subseteq \mathsf{NC}^1$ or a similar breakthrough.
> 2. (*Evidence of Hardness*) $Q \notin \mathcal{C}$ under a standard conjecture.
> 3. (*Lower Bound against* $\mathcal{C}$) $L \notin \mathcal{C}$, where L is a simple function like PARITY.
> 4. (*Lower Bound for* Q) $Q \notin \mathcal{C}^-$, where $\mathcal{C}^-$ is slightly weaker than $\mathcal{C}$.

A frontier of this form provides hope that the required lower bound in Item 1 is true (thanks to Item 2), and that it might be within the reach of known techniques (thanks to Items 3 and 4, which provide evidence that we can analyse the circuit model and the problem). HM frontiers have been already achieved in earlier works with a striking example appearing in [39] (see also [10]). Despite the number of works in this area, we note that the HM frontier is achieved only by some magnification theorems (Item 3 is often unknown; e.g. in the case of results in [3, 13]).

[1] Most of this work was completed while Igor C. Oliveira was affiliated with the University of Oxford.

In order to make our subsequent discussion more concrete, we provide five examples of HM frontiers. Some of these results are new or require an extension of previous work, and the relevant statements will be explained in more detail in Section 3. The list of frontiers is not meant to be exhaustive, but we have tried to cover different computational models.

(A) HM Frontier for $\mathsf{MKtP}[n^c, 2n^c]$ and AC^0-XOR:

- **A1.** If $\mathsf{MKtP}[n^c, 2n^c] \notin \mathsf{AC}^0\text{-XOR}[N^{1.01}]$ for large $c > 1$ then $\mathsf{EXP} \not\subseteq \mathsf{NC}^1$ (Section 3.1).
- **A2.** $\mathsf{MKtP}[n^c, 2n^c] \notin \mathsf{P/poly}$ for large enough c under exponentially secure PRFs [46].
- **A3.** $\mathsf{Majority} \notin \mathsf{AC}^0\text{-XOR}[2^{N^{o(1)}}]$ (immediate from [44, 49]).
- **A4.** $\mathsf{MKtP}[n^c, 2n^c] \notin \mathsf{AC}^0$ for any sufficiently large constant c (Section 3.1).

A. $\mathsf{MKtP}[s, t]$ refers to the promise problem of determining if an N-bit input has Levin Kolmogorov complexity at most s versus at least t (cf. [39]). Here $N = 2^n$. The AC^0-XOR model is the extension of AC^0 where gates at the bottom layer of the circuit can compute arbitrary parity functions. $\mathsf{AC}^0\text{-XOR}[s]$ denotes AC^0-XOR circuits of size s where the size is measured as the number of gates. This circuit class has received some attention in recent years (cf. [15]), and a few basic questions about AC^0 circuits with parity gates (such as constructing PRGs of seed length $o(n)$ and learnability using random examples) remain open for AC^0-XOR as well.

(B) HM Frontier for $\mathbf{MCSP}[2^{n^{1/3}}, 2^{n^{2/3}}]$ and Formula-XOR:

- **B1.** $\mathsf{MCSP}[2^{n^{1/3}}, 2^{n^{2/3}}] \notin \mathsf{Formula\text{-}XOR}[N^{1.01}]$ implies $\mathsf{NQP} \not\subseteq \mathsf{NC}^1$ (Section 3.2).
- **B2.** $\mathsf{MCSP}[2^{n^{1/3}}, 2^{n^{2/3}}] \notin \mathsf{P/poly}$ under standard cryptographic assumptions [46].
- **B3.** $\mathsf{InnerProduct} \notin \mathsf{Formula\text{-}XOR}[N^{1.99}]$ (immediate consequence of [52]).
- **B4.** $\mathsf{MCSP}[2^{n^{1/3}}, 2^{n^{2/3}}] \notin \mathsf{Formula}[N^{1.99}]$ ([23]; see also [39]).

B. Here, NQP is nondeterministic quasi-polynomial time, InnerProduct is the Boolean function defined as $\mathsf{InnerProduct}(x, y) = \sum_i x_i \cdot y_i \pmod 2$, where $x, y \in \{0,1\}^N$, Formula-XOR$[s]$ refers to the class of Boolean formulas over the De Morgan basis with at most s leaves, where each leaf is an XOR of arbitrary arity over the inputs[2], and $\mathsf{MCSP}[s, t]$ denotes a promise problem over $N = 2^n$ input bits with YES inputs being truth-tables of Boolean functions on n inputs which are computable by circuits of size s, and NO instances being truth-tables of Boolean functions which are hard for circuits of size t.

(C) HM Frontier for $\mathbf{MCSP}[2^{n^{1/2}}/10n, 2^{n^{1/2}}]$ and Almost-Formulas:

- **C1.** $\mathsf{MCSP}[\frac{2^{n^{1/2}}}{10n}, 2^{n^{1/2}}] \notin N^{0.01}\text{-}\mathsf{Almost\text{-}Formula}[N^{1.01}]$ implies $\mathsf{NP} \not\subseteq \mathsf{NC}^1$ (Section 3.3).
- **C2.** $\mathsf{MCSP}[\frac{2^{n^{1/2}}}{10n}, 2^{n^{1/2}}] \notin \mathsf{P/poly}$ under standard cryptographic assumptions [46].
- **C3.** $\mathsf{PARITY} \notin N^{0.01}\text{-}\mathsf{Almost\text{-}Formula}[N^{1.01}]$ (Section 3.3).
- **C4.** $\mathsf{MCSP}[2^{n^{1/2}}/10n, 2^{n^{1/2}}] \notin \mathsf{Formula}[N^{1.99}]$ ([23]; see also [39]).

[2] Note that $\mathsf{Formula\text{-}XOR}[N^{1.01}] \subseteq \mathsf{Formula}[N^{3.01}]$. A better understanding of the former class is therefore necessary before we can understand the power and limitations of super-cubic formulas, which is a major open question in circuit complexity.

C. An almost-formula is a circuit with a bounded number of gates of fan-out larger than 1. More precisely, a γ-Almost-Formula$[s]$ is a circuit containing at most s AND, OR, NOT gates of fan-in at most 2, and among such gates, at most γ of them have fan-out larger than 1. Consequently, this class naturally interpolates between formulas and circuits. This magnification frontier can be seen as progress towards establishing magnification theorems for worst-case variants of MCSP in the regime of sub-quadratic formulas (see the discussion in [39]).

(D) HM Frontier for MCSP$[2^{\sqrt{n}}]$ and one-sided error randomized formulas:

D1. MCSP$[2^{\sqrt{n}}] \notin$ GapAND$_{O(N)}$-Formula$[N^{2.01}] \Rightarrow$ NQP $\notin$ NC^1 (Section 3.2).
D2. MCSP$[2^{\sqrt{n}}] \notin$ P/poly under standard cryptographic assumptions [46].
D3.1. Andreev$_N \notin$ GapAND$_{O(N)}$-Formula$[N^{2.99}]$ (implicit in [21]).
D3.2. MCSP$[2^n/n^4] \notin$ GapAND$_{O(N)}$-Formula$[N^{2.99}]$ (implicit in [16]).
D4. MCSP$[2^{\sqrt{n}}] \notin$ GapAND$_{O(N)}$-Formula$[N^{1.99}]$ ([11], building on [23, 39]).

D. GapAND$_N$ is the promise function on N bits such that it outputs 1 when all input bits are 1, and outputs 0 when at most 1/10 of the input bits are 1. GapAND$_{O(N)}$-Formula$[s]$ denotes circuits with GapAND$_{O(N)}$ gate at the top with formulas of size s being inputs of the top gate. Therefore, GapAND$_{O(N)}$-Formula can be seen as *randomized formulas with one-sided error*.[3] The most interesting aspect of this magnification frontier is that the gap between the known hardness result and the magnification threshold is *nearly-tight* ($N^{2-\varepsilon}$ versus $N^{2+\varepsilon}$).[4]

(E) HM Frontier for $(n-k)$-Clique and AC^0:

E1. If $(n-k)$-Clique $\notin$ AC$^0[m^{1.01}]$ for some $k = (\log n)^C$, then NP $\not\subseteq$ NC^1 (Section 3.4).
E2. (Non-uniform) ETH $\Rightarrow$ $(n-k)$-Clique $\notin$ P/poly for some $k = (\log n)^C$ (Section 3.4).
E3. Parity $\notin$ AC^0 [1, 19].
E4. $(n-k)$-Clique $\notin$ mP/poly for some $k = (\log n)^C$ ([4]; see Section 3.4).

E. The ℓ-Clique problem is defined on graphs on n vertices in the adjacency matrix representation of size $m = \Theta(n^2)$. (The statements above refers to the regime of very large clique detection.) The class mP/poly refers to monotone circuits of polynomial size. In this frontier we are modifying Item 4 from HM frontier so that instead of slightly weaker $\mathcal{C}^-$ we consider an incomparable $\mathcal{C}^-$. This frontier is however particularly interesting, as items E1 and E4 connect hardness magnification to a basic question about the power of *non-monotone*

[3] Suppose there is a GapAND$_{O(N)}$-Formula circuit computing a function $f\colon \{0,1\}^N \to \{0,1\}$. Consider a uniform distribution of all sub-formulas below the top GapAND$_{O(N)}$ gate. Then for any input x, if $f(x) = 1$ then a sample formula from that distribution always outputs 1 on x, otherwise it outputs 0 with probability at least 0.9 on x. On the other hand, it is possible to derandomize a distribution of formulas computing f with one-sided error using a top GapAND$_{O(N)}$ gate.

[4] This tight threshold is first observed in [11], we include it here to show that the barrier discussed in this paper also applies to this particular setting.

circuits when computing *monotone* functions (see [14, 20] and references therein): Is every monotone function in AC^0 computable by a monotone (unbounded depth) boolean circuit of polynomial size? If this is the case, $\mathsf{NP} \not\subseteq \mathsf{NC}^1$ would follow.

Note that these hardness magnification frontiers offer different approaches to proving lower bounds against NC^1.

Essential Questions. Do magnification theorems bring us closer to strong circuit lower bounds? In order to understand the limits and prospects of hardness magnification, the following questions are relevant.

Q1. *Naturalization.* Is hardness magnification a *non-naturalizing* approach to circuit lower bounds? If we accept standard cryptographic assumptions, non-naturalizability is a necessary property of any successful approach to strong circuit lower bounds.[5]

Q2. *Extending known lower bounds.* Can we adapt an existing lower bound proof from Items 3 and 4 in some HM frontier to show the lower bound required from Item 1 in that HM frontier? Is it possible to establish the required lower bounds via a reduction from L to Q?

Q3. *Improving existing magnification theorems.* Can we close the gap between Items 1 and 4 in HM frontier by establishing a magnification theorem that meets *known* lower bounds, such as the ones appearing in Item 4?

In the next sections, we present results that shed light into all these questions.

1.2 Hardness Magnification and Natural Proofs

The very existence of the natural proofs barrier provides a direction for proving strong circuit lower bounds: one can proceed by *refuting the existence of natural properties*.[6] In other words, a way to avoid natural proofs is to prove that there are no natural proofs. It is also easy to see that $\mathsf{P/poly}$-natural properties useful against $\mathsf{P/poly}$ can be turned into natural properties with much higher constructivity, e.g. into linear-size natural properties useful against circuits of polynomial-size. If read contrapositively, this gives a form of hardness magnification.

The initial hardness magnification theorem of Oliveira and Santhanam [42] proceeds in a similar fashion. It proposes to approach $\mathsf{NP} \not\subseteq \mathsf{P/poly}$ by deriving slightly superlinear circuit lower bounds for specific problems such as an *approximate version* of MCSP, which asks to distinguish truth-tables of Boolean functions computable by small circuits from truth-tables of Boolean functions which are hard to approximate by small circuits. Interestingly, this approach does not seem to naturalize, as it appears to yield strong lower bounds only for certain problems, and not for most of them. (The same heuristic argument appears in [3].) However, this is only an informal argument, and we would like to get stronger evidence that the natural proofs barrier does not apply here.

We show that hardness magnification for approximate MCSP can be used to conclude the *non-existence* of natural proofs against polynomial-size circuits. More precisely, we prove that if approximate MCSP requires slightly superlinear-size circuits, then there are no $\mathsf{P/poly}$-natural properties against $\mathsf{P/poly}$. This strongly suggests that the natural proofs

[5] We assume familiarity of the reader with the natural proofs framework of [46].

[6] A similar perspective has been employed in proof complexity in attempts to approach strong proof complexity lower bounds by extending the natural proofs barrier (see [34, 45]).

barrier isn't relevant to the magnification approach. Indeed, there remains the possibility that the weak circuit lower bound for MCSP in the hypothesis of the result can be shown using naturalizing techniques (as there aren't any strong enough plausible cryptographic conjectures known that rule this out), and yet by using magnification to "break" naturalness, we could get strong circuit lower bounds and even conclude the non-existence of natural proofs!

The core of our proof is the following new hardness magnification theorem: if approximate MCSP requires slightly superlinear-size circuits, then not only $\mathsf{NP} \not\subseteq \mathsf{P/poly}$ but *it is impossible even to learn efficiently*. We can then refute the existence of natural proofs by applying the known translation of natural properties to learning algorithms [8]. Similar implications hold with a *worst-case gap version* of MCSP (in the sense of HM Frontiers B and C but with different parameters) instead of approximate MCSP, following an idea from [22].

Interestingly, all the implications above are actually *equivalences*. In particular, the existence of natural properties is equivalent to the existence of highly efficient circuits for computing approximate MCSP and worst-case gap MCSP with certain parameters (cf. Theorem 1). This extends a known characterization of natural properties: Carmosino et al. [8] showed that P/poly natural proofs against P/poly are equivalent to learning P/poly by subexponential-size circuits, which was in turn shown to be equivalent by Oliveira and Santhanam [41] to the non-existence of non-uniform pseudorandom function families of sub-exponential security. The connection of hardness magnification to learning and pseudorandom function generators might be of independent interest, since it extends the consequences of magnification into two central areas in Complexity Theory.

▶ **Theorem 1** (Equivalences for Hardness Magnification). *The following statements are equivalent:*[7]

(a) ***Hardness of approximate* MCSP *against almost-linear size circuits.*** *There exist $c \geq 1$, $0 < \gamma < 1$, and $\varepsilon > 0$ such that $\mathit{MCSP}[(n^c, 0), (2^{n^\gamma}, n^{-c})] \notin \mathit{Circuit}[N^{1+\varepsilon}]$.*

(b) ***Hardness of worst-case* MCSP *against almost-linear size circuits.*** *There exists $c \geq 1$ and $\varepsilon > 0$ such that $\mathit{MCSP}[n^c, 2^n/n^c] \notin \mathit{Circuit}[N^{1+\varepsilon}]$.*

(c) ***Hardness of sub-exponential size learning using non-adaptive queries.*** *There exist $\ell \geq 1$ and $0 < \gamma < 1$ such that $\mathit{Circuit}[n^\ell]$ cannot be learned up to error $O(1/n^\ell)$ under the uniform distribution by circuits of size $2^{O(n^\gamma)}$ using non-adaptive membership queries.*

(d) ***Non-existence of natural properties against polynomial size circuits.*** *For some $d \geq 1$ there is no $\mathit{Circuit}[\mathsf{poly}(N)]$-natural property useful against $\mathit{Circuit}[n^d]$.*

(e) ***Existence of non-uniform* PRFs *secure against sub-exponential size circuits.*** *For every constant $a \geq 0$, there exists $d \geq 1$, a sequence $\mathfrak{F} = \{\mathcal{F}_n\}_{n\geq 1}$ of families $\mathcal{F}_n$ of n-bit boolean functions $f_n \in \mathit{Circuit}[n^d]$, and a sequence of probability distributions $\mathfrak{D} = \{\mathcal{D}_n\}_{n\geq 1}$ supported over $\mathcal{F}_n$ such that, for infinitely many values of n, $(\mathcal{F}_n, \mathcal{D}_n)$ is pseudo-random function family that $(1/N^{\omega(1)})$-fools (oracle) circuits of size $2^{a\cdot n}$.*

The proof of this result appears in Section 4.1. We highlight below the most interesting implications of Theorem 1. (Note that some of them have appeared in other works in similar or related forms.)

- $(a) \to (d)$: The initial hardness magnification result from [42, Theorem 1] (stated for circuits) implies the *non-existence* of natural proofs useful against polynomial-size circuits, indicating that the natural proofs barrier might not be relevant to the magnification approach.

[7] See Preliminaries (Section 2) for definitions.

- (a), $(b) \leftrightarrow (d)$: *Any* P/poly natural property useful against P/poly can be transformed into an almost-linear size natural property that is simply the approximate $\mathsf{MCSP}[(n^c, 0), (2^{n^\gamma}, n^{-c})]$ or worst-case gap $\mathsf{MCSP}[n^c, 2^n/n^c]$. (Note the different regime of circuit size parameters for these problems.)
- (a), $(b) \leftrightarrow (c)$: A weak-seeming hardness assumption for worst-case gap and approximate versions of MCSP implies a strong non-learnability result: polynomial-size circuits cannot be learned over the uniform distribution even non-uniformly in sub-exponential time.
- (a), $(b) \leftrightarrow (e)$: Hardness magnification for MCSP also yields cryptographic hardness in a certain regime.

We note that the use of non-adaptive membership queries in Theorem 4.1 Item (c) is not essential. It follows from [8] that, in the context of learnability of polynomial size circuits under the uniform distribution in sub-exponential time, adaptive queries are not significantly more powerful than non-adaptive queries.[8]

Towards a more robust theory. While Theorem 1 formally connects hardness magnification and natural properties, it would be very interesting to understand to which extent different hardness magnification theorems are provably non-naturalizable. This would provide a more complete answer to Question Q1 asked above. For instance, Theorem 1 leaves open whether hardness magnification for worst-case versions of MCSP such as $\mathsf{MCSP}[n^c, 2^{n^\varepsilon}]$ refutes natural proofs as well. Note that one way of approaching this question would be to study reductions from $\mathsf{MCSP}[n^c, 2^{n^\gamma}]$ to its approximate version $\mathsf{MCSP}[(n^{c'}, 0), (2^{n^{\gamma'}}, n^{-c'})]$.[9] In Section 4.2, we observe that this question is related to the problem of basing *hardness of learning* on *worst-case assumptions* such as $\mathsf{P} \neq \mathsf{NP}$ (cf. [5]). We refer to the discussion in Section 4.2 for more details.

1.3 The Locality Barrier

The results from the preceding section show that hardness magnification can go beyond natural proofs. Is there another barrier that makes it difficult to establish lower bounds via magnification? In this section, we present a general argument to explain why the lower bound techniques behind A3-E3, A4-D4 in the magnification frontiers from Section 1.1 cannot be adapted (without significantly new ideas) to establish the required lower bounds in Items A1-E1, respectively. We refer to it as the *locality barrier*. While we will focus on these particular examples to make the discussion concrete, we believe that this barrier applies more broadly.

In order to explain the locality barrier, let's consider the argument behind the proof of B1 presented in Section 3.2. Recall that this result shows that if $\mathsf{MCSP}[2^{n^{1/3}}, 2^{n^{2/3}}] \notin \mathsf{Formula\text{-}XOR}[N^{1.01}]$ then $\mathsf{NQP} \not\subseteq \mathsf{NC}^1$. This and other known hardness magnification theorems are established in the contrapositive. The core of the argument is to prove that there are highly efficient Formula-XOR circuits that reduce an input to $\mathsf{MCSP}[2^{n^{1/3}}, 2^{n^{2/3}}]$ of length $N = 2^n$ to deciding whether certain strings of length N' (much smaller than N) belong to

[8] In a bit more detail, one can easily extract a natural property from a learner that uses adaptive queries. In turn, closer inspection of the technique of [8] shows that a non-adaptive learner can be obtained from a natural property.

[9] More precisely, the existence of a reduction from $\mathsf{MCSP}[n^c, 2^{n^\gamma}]$ to $\mathsf{MCSP}[(n^{c'}, 0), (2^{n^{\gamma'}}, n^{-c'})]$ shows that lower bounds for the former problem yield lower bounds for the latter. Since any such lower bound must be non-naturalizable by Theorem 1, we obtain the same consequence for $\mathsf{MCSP}[n^c, 2^{n^\gamma}]$. (Note that in the context of hardness magnification it is also important to have highly efficient reductions.)

a certain language L'. Then, under the assumption that $\mathsf{NQP} \subseteq \mathsf{NC}^1$, one argues that L' has polynomial size formulas. Finally, since $N' \ll N$, we can employ such formulas and still conclude that $\mathsf{MCSP}[2^{n^{1/3}}, 2^{n^{2/3}}]$ is in $\mathsf{Formula\text{-}XOR}[N^{1.01}]$, which completes the proof.

Note that the argument above provides a *conditional* construction of highly efficient formulas for the original problem. Crucially, however, we can derive an *unconditional* circuit upper bound from this argument: If we stop right before we replace the calls to L' by an algorithm for L' (this is what makes the reduction conditional), it unconditionally follows that $\mathsf{MCSP}[2^{n^{1/3}}, 2^{n^{2/3}}]$ can be computed by highly efficient $\mathsf{Formula\text{-}XOR}$ circuits containing oracle gates of small fan-in, for some oracle. Similarly, one can argue that the problems in Items A1-E1 can be computed in the respective models by highly efficient Boolean devices containing oracles of small fan-in.

We stress that, as opposed to a magnification theorem, where one cares about the complexity of the oracle gates, in our discussion of the locality barrier we only need the fact that there is *some way* of setting these oracles gates so that the resulting circuit or formula solves the original problem. (A definition of this model appears in Section 2.5.) A more exhaustive interpretation of magnification theorems as construction of circuits with small fan-in oracles can be found in Appendix A.2.

On the other hand, we argue that the lower bound arguments from Items A3-E3 of the hardness magnification frontiers quite easily handle (in the respective models) the presence of oracles of small fan-in, *regardless of the function* computed by these oracles. Using a more involved argument we can localize also lower bounds from items A4-D4. Consequently, these methods do not seem to be refined enough to prove the lower bounds required by A1-D1 without excluding oracle circuits that are unconditionally known to exist for the corresponding problems.

Following the example above, we state our results for the Magnification Frontier B.

▶ **Theorem 2** (Locality Barrier for HM Frontier B). *The following results hold.*

- (B1$^{\mathcal{O}}$) (Oracle Circuits from Magnification) : *For any $\varepsilon > 0$, $\mathsf{MCSP}[2^{n^{1/3}}, 2^{n^{2/3}}] \in \mathsf{Formula\text{-}}\mathcal{O}\mathsf{\text{-}XOR}[N^{1.01}]$ for some oracle $\mathcal{O}$, where every oracle gate has fan-in at most N^{ε} and appears in the layer right above the* XOR *leaves.*
- (B3$^{\mathcal{O}}$) (Extension of Lower Bound Techniques Above Magnification Threshold) : *For any $\delta > 0$,* InnerProduct *over N input bits cannot be computed by $N^{2-3\delta}$-size $\mathsf{Formula\text{-}}\mathcal{O}\mathsf{\text{-}XOR}$ circuits with at most $N^{2-3\delta}$ oracle gates of fan-in N^{δ} in the layer right above the* XOR *leaves, for any oracle $\mathcal{O}$.*
- (B4$^{\mathcal{O}}$) (Extension of Lower Bound Techniques Below Magnification Threshold) : *There is a universal constant c such that for all constants $\varepsilon > 0$ and $\alpha > 2$, $\mathsf{MCSP}[n^c, 2^{\varepsilon/\alpha \cdot n}]$ cannot be computed by oracle formulas F with $\mathsf{SIZE}_3(F) \leq N^{2-\varepsilon}$ and adaptivity $o(\log N / \log\log N)$.*[10]

Here, $\mathsf{Size}_t(F)$ denotes the size of the formula, if we replace every oracle $\mathcal{O}$ with fan-in β in F by a formula of size β^t which reads all its inputs exactly β^{t-1} times (see Section 5.2.2 for the motivation of this definition).

The first two items of Theorem 2 are proved in Section 5.1.2. The third item is proved in Section 5.2.2. While Theorem 2 does not specify that, we actually localize all proofs of the lower bounds from B3 and B4 we are aware of. Interestingly, the localization of B4 allows us to refute the Anti-Checker Hypothesis from [39] (and a family of potential hardness magnification theorems), cf. Section 5.2.2. We refer to Section 5 for analogous statements describing the locality barrier in frontiers A, C, D, and E.

[10] That is, on any path from root to a leaf, there are at most $o(\log N / \log\log N)$ oracles.

Locality of Computations and Lower Bound Techniques. The fact that many lower bound techniques extend to computational devices with oracles of small fan-in was observed already by Yao in 1989 on a paper on local computations [56]. According to Yao, a local function is one that can be efficiently computed using only localized processing elements. In our terminology, this corresponds to circuits with oracles of small fan-in. Among other results, [56] argues that Razborov's monotone circuit size lower bound for k-Clique [43] and Karchmer and Wigderson's monotone formula size lower bound for ST-CONN [31] extend to boolean devices with monotone oracles of bounded fan-in. Compared to Yao's work, our motivation and perspective are different. While Yao is particularly interested in lower bounds that can be extended in this sense (see e.g. Sections 2 and 6 in [56]), here we view such extensions as a *limitation* of the corresponding arguments, meaning that they are not refined enough to address the locality barrier.[11]

We note, however, that not every lower bound technique extends to circuits with small fan-in oracles.[12] For instance, by the work of Allender and Koucký [3] (also a more recent work by Chen and Tell [13]), the parity function Parity_n over n input bits can be computed by a TC^0 circuit of size $O(n)$ (number of wires) containing $\leq n^{1-\varepsilon}$ oracle gates of fan-in $\leq n^{\varepsilon}$, provided that its depth $d = O(1/\varepsilon)$. On the other hand, it is known that $\mathsf{Parity}_n \notin \mathsf{TC}^0_d[n^{1+c^{-d}}]$ for a constant $c > 0$ [25] (again, the complexity measure is the number of wires). Since the latter lower bound is super-linear for every choice of d, it follows by the result of [3, 13] that it cannot be extended to circuits containing a certain number of oracles of fan-in n^{ε}, for a large enough depth d that depends on ε. Incidentally, the hardness magnification theorems of [3, 13] do not achieve a magnification frontier.

In Section 3.2 we identify one specific lower bound related to HM frontier D which is both above the magnification threshold and provably non-localizable, cf. Theorem 49. In principle, there might be ways to overcome the locality barrier and match the lower bound with the magnification threshold. We refer to Section 1.4 below for additional discussion.

On Lower Bounds Through Reductions. The discussion above has focused on the possibility of *directly* adapting existing lower bounds from Item 3 in HM frontier to establish the desired lower bound in Item 1. There is however an *indirect* approach that one might hope to use: *reductions*. For instance, in the context of the HM Frontier B discussed above, can we have a reduction from InnerProduct to $\mathsf{MCSP}[2^{n^{1/3}}, 2^{n^{2/3}}]$ that would allow us to show that $\mathsf{MCSP}[2^{n^{1/3}}, 2^{n^{2/3}}] \notin \mathsf{Formula\text{-}XOR}[N^{1.01}]$? The first thing to notice is that, for this approach to make sense, the reduction needs to have a specific form so that composing the reduction with a candidate Formula-XOR circuit for $\mathsf{MCSP}[2^{n^{1/3}}, 2^{n^{2/3}}]$ violates the hardness of InnerProduct. Is there any hope to design a reduction of this form?

The locality barrier presents a *definitive answer* in this case. Indeed, it is immediate from the first two items of Theorem 2 that such a reduction *does not exist*. For the same reason, it is not possible to use reductions to establish the required lower bounds in some other magnification frontiers, cf. Section 5.1.6.

[11] On a more technical level, we are interested in the regime of barely super-linear size circuits and formulas, and our results do not impose a monotonicity constraint on the oracle.

[12] Of course any such discussion depends on parameters such as number of oracles and their fan-ins, so whether a technique avoids or not the locality barrier is relative to a particular magnification theorem.

1.4 Concluding Remarks and Open Problems

Hardness magnification shows that obtaining a refined understanding of weak computational models is an approach to major complexity lower bounds, such as separating EXP from NC^1. As discussed in Sections 1.1 and 1.2 above, its different instantiations are connected to a few basic questions in Complexity Theory, including the power of non-monotone operations, learnability of circuit classes, and pseudorandomness.

One of our main conceptual contributions in this work is to identify a challenge when implementing this strategy for lower bounds. Quoting the influential article [46] that introduced the natural proofs barrier,

> *"We do not conclude that researchers should give up on proving serious lower bounds. Quite the contrary, by classifying a large number of techniques that are unable to do the job we hope to focus research in a more fruitful direction."*
>
> Razborov and Rudich [46, Section 6]

We share a similar opinion with respect to hardness magnification and the obstruction identified in Section 1.3. While locality provides a unified explanation for the difficulty of adapting combinatorial lower bound techniques to exploit most (if not all) known magnification frontiers, it might be possible to discover new HM frontiers whose associated lower bound techniques in Item 3 are sensitive to the presence of small fan-in oracles. For instance, in the case of *uniform* complexity lower bounds, this has been achieved in [38] via an indirect diagonalization that explores the theory of pseudorandomness.[13] Alternatively, it might be possible to establish magnification theorems using a technique that does not produce circuits with small fan-in oracles. Even if one is pessimistic about these possibilities, we believe that an important contribution of the theory of hardness magnification is to break the divide between "weak" and "strong" circuit classes advocated by the natural proofs barrier, and that it deserves further investigation.

We finish with a couple of technical questions related to our contributions. First, we would like to understand if it is possible to strengthen items (a) and (b) in Theorem 1 to a wider range of parameters. For example, is hardness magnification for worst-case $\mathsf{MCSP}[n^c, 2^{n^\gamma}]$ with $\gamma < 1$ non-naturalizable? The core of this question seems to be the problem of reducing worst-case MCSP from item (a) to approximate MCSP from item (b).

Another important direction is to show that hardness magnification avoids natural proofs also in the context of *non*-meta-computational problems. Interestingly, many magnification theorems from [39] established for MCSP and variants were subsequently shown to hold for any sparse language in NP [10]. Could it be the case that hardness magnification overcomes natural proofs in a much broader sense?

Finally, it would be useful to investigate the locality of additional lower bound techniques. Can we, for example, come up with non-localizable lower bounds similar to Theorem 49 which would be above the magnification threshold and work for a problem more closely related to the one from the corresponding HM frontier?

[13] In other words, the magnification theorem discussed in [38] admits a formulation for uniform randomized algorithms, and its proof provides an algorithm with oracle gates of small fan-in in the spirit of the oracle circuits discussed here. Nevertheless, the unconditional lower bound established in the same paper does not extend to algorithms with such oracle gates.

2 Preliminaries

2.1 Notation

Given a Boolean function $f : \{0,1\}^n \to \{0,1\}$, $\mathsf{tt}(f)$ denotes the 2^n-bit string representing the truth table of f. On the other hand, for any string $y \in \{0,1\}^{2^n}$, define f_y as the function on n inputs such that $\mathsf{tt}(f_y) = y$.

$\mathsf{Circuit}[s]$ denotes fan-in two Boolean circuits of size at most s where we count the number of gates. $\mathsf{Formula}[s]$ denotes formulas over the basis U_2 (fan-in two ANDs and ORs) of size at most s (counting the number of leaves) with input leaves labelled by literals or constants.

For a circuit class $\mathcal{C}$, $\mathcal{C}[s]$ denotes circuits from $\mathcal{C}$ of size at most s.

A function $f : \{0,1\}^n \to \{0,1\}$ is γ-approximated by a circuit C, if $\Pr_x[C(x) = f(x)] \geq \gamma$.

2.2 Complexity of Learning

▶ **Definition 3** (Learning). *A circuit class $\mathcal{C}$ is learnable over the uniform disribution by circuits in $\mathcal{D}$ up to error ε with confidence δ if there are randomized oracle $\mathcal{D}$-circuits L^f such that for every boolean function $f : \{0,1\}^n \mapsto \{0,1\}$ computable by a circuit from $\mathcal{C}$, when given oracle access to f, input 1^n and the internal randomness $w \in \{0,1\}^*$, L^f outputs the description of a circuit satisfying*

$$\Pr_w\{L^f(1^n, w) \ (1-\varepsilon)\text{-approximates } f\} \geq \delta.$$

L^f uses non-adaptive membership queries if the set of queries which L^f makes to the oracle does not depend on the answers to previous queries. If $\delta = 1$, we omit mentioning the confidence parameter.

2.3 Natural Properties, MCSP, and its Variants

Let $\mathcal{F}_n$ be the set of all functions on n variables. $\mathfrak{R} = \{\mathcal{R}_n \subseteq \mathcal{F}_n\}_{n\in\mathbb{N}}$ is a combinatorial property of Boolean functions.

▶ **Definition 4** (Natural property [46]). *Let $\mathfrak{R} = \{\mathcal{R}_n\}$ be a combinatorial property, $\mathcal{C}$ be a circuit class and Γ be a complexity class. Then, $\mathfrak{R}$ is a Γ-natural property useful against $\mathcal{C}[s(n)]$, if there exists an $n_0 \in \mathbb{N}$ such that the following hold:*

- ***Constructivity** : For any function $f_n \in \mathcal{F}_n$, the predicate $f_n \overset{?}{\in} \mathcal{R}_n$ is computable in Γ in the size of the truth table of f_n.*
- ***Largeness** : For every $n \geq n_0$, $\Pr_{f_n\sim\mathcal{F}_n}\{f_n \in \mathcal{R}_n\} \geq \frac{1}{2^{O(n)}}$.*
- ***Usefulness** : For every $n \geq n_0$, $\mathcal{R}_n \cap \mathcal{C}[s(n)] = \emptyset$.*

The following result which follows from [8] connects the existence of natural properties useful against a class $\mathcal{C}$ to designing learning algorithms for $\mathcal{C}$.

▶ **Theorem 5** (From Theorem 5.1 of [8] and Lemma 14 of [27]). *Let R be a P/poly-natural property useful against $\mathsf{Circuit}[n^d]$ for some $d \geq 1$. Then, for each $\gamma \in (0,1)$, there are randomized, oracle circuits $\{D_n\}_{n\geq 1} \in \mathsf{Circuit}[2^{O(n^\gamma)}]$ that learn $\mathsf{Circuit}[n^k]$ up to error $\frac{1}{n^k}$ using non-adaptive oracle queries to f_n, where $k = \frac{d\gamma}{a}$ and a is a universal constant that does not depend on d and γ.*

▶ **Definition 6** (Gap MCSP). *Let $s, t : \mathbb{N} \to \mathbb{N}$, where $s(n) \leq t(n)$ and $0 \leq \varepsilon, \sigma < 1/2$. Define MCSP$[(s, \sigma), (t, \varepsilon)]$ on inputs of length $N = 2^n$, as the following promise problem :*

- *YES instances: $y \in \{0,1\}^N$ such that there exists a circuit of size $s(n)$ that $(1-\sigma)$-approximates f_y.*
- *NO instances: $y \in \{0,1\}^N$ such that no circuit of size $t(n)$ $(1-\varepsilon)$-approximates f_y.*

We refer to MCSP$[(s,0),(t,0)]$ as MCSP$[s,t]$. Informally speaking, if $\varepsilon > 0$, we say that MCSP$[(s,0),(t,\varepsilon)]$ is an *approximate* version of MCSP. Otherwise, it is a *worst-case* version of MCSP.

▶ Remark 7. In Definition 6, if $s(n) = t(n)$, we also require that $\sigma < \varepsilon$ for the yes and no instances to be disjoint.

▶ **Definition 8** (Succinct MCSP). *For functions $s, t : \mathbb{N} \mapsto \mathbb{N}$, Succinct-MCSP$[s(n), t(n)]$ is the following problem. Given an input $\langle 1^n, 1^s, (x_1, b_1), \ldots, (x_t, b_t)\rangle$ where $x_i \in \{0,1\}^n, b_i \in \{0,1\}$, decide if there is a circuit C of size s such that $\bigwedge_{i=1,\ldots,t} C(x_i) = b_i$.*

2.4 Pseudorandom Generators

▶ **Definition 9** (Pseudorandom function families). *For any circuit class $\mathcal{C}$, size functions $s(n), t(n) \geq n$, family $\mathcal{G}_n$ of n-bit Boolean functions and distribution $\mathcal{D}_n$ over $\mathcal{G}_n$, we say that a pair $(\mathcal{G}_n, \mathcal{D}_n)$ is a $(t(n), \varepsilon(n))$-pseudorandom function family (PRF) in $\mathcal{C}[s(n)]$, if each function in $\mathcal{G}_n$ is in $\mathcal{C}[s(n)]$ and for every randomized circuit $A^O \in$ Circuit$^O[t(n)]$, where O denotes oracle access to a fixed Boolean function over n inputs, we have*

$$\left| \Pr_{g \sim \mathcal{D}_n, w} \{A^g(w) = 1\} - \Pr_{f \sim \mathcal{F}_n, w} \{A^f(w) = 1\} \right| \leq \varepsilon(n)$$

[41] state an equivalence between the non-existence of PRFs in a circuit class $\mathcal{C}$ and learning algorithms for $\mathcal{C}$. In particular, we care about the following direction which they prove using a small-support version of Von-Neumann's Min-max Theorem.

▶ **Theorem 10** (No PRFs in $\mathcal{C}$ implies Learning Algorithm for $\mathcal{C}$ [41]). *Let $t(n) \leq 2^{O(n)}$. Suppose that for every $k \geq 1$ and large enough n, there exists no $(\mathsf{poly}(t(n)), 1/10)$-pseudorandom function families in $\mathcal{C}[n^k]$. Then, for every $\varepsilon > 0, k \geq 1$ and large enough n, there is a randomized oracle circuit in Circuit$^O[2^{n^\varepsilon}]$ that learns every function $f_n \in \mathcal{C}[n^k]$ up to error $1/n^k$ with confidence $1 - 1/n$, where O denotes membership query access to f_n.*

2.5 Local Circuit Classes

Our definition of local computation is somewhat similar to some definitions appearing in [56].

▶ **Definition 11** (Local circuit classes). *Let $\mathcal{C}$ be a circuit class (such as $\mathsf{AC}^0[s]$, $\mathsf{TC}^0_d[s]$, Circuit$[s]$, etc). For functions $q, \ell, a \colon \mathbb{N} \to \mathbb{N}$, we say that a language L is in $[q, \ell, a]$–$\mathcal{C}$ if there exists a sequence $\{E_n\}$ of oracle circuits for which the following holds:*

- **(i)** *Each oracle circuit E_n is a circuit from $\mathcal{C}$.*
- **(ii)** *There are at most $q(n)$ oracle gates in E_n, each of fan-in at most $\ell(n)$, and any path from an input gate to an output gate encounters at most $a(n)$ oracle gates.*
- **(iii)** *There exists a language $\mathcal{O} \subseteq \{0,1\}^*$ such that the sequence $\{E_n^{\mathcal{O}}\}$ (E_n with its oracle gates set to $\mathcal{O}$) computes L.*

In the definition above, q stands for *quantity*, ℓ for *locality*, and a for *adaptivity* of the corresponding oracle gates.

2.6 Random Restrictions

Let $\rho : [N] \to \{0, 1, *\}$ be a *restriction*, and $\boldsymbol{\rho}$ be a *random restriction*, i.e., a distribution of restrictions. We say that $\boldsymbol{\rho}$ is p-regular if $\Pr[\boldsymbol{\rho}(i) = *] = p$ and $\Pr[\boldsymbol{\rho}(i) = 0] = \Pr[\boldsymbol{\rho}(i) = 1] = (1 - p)/2$ for every $i \in [N]$. We also say $\boldsymbol{\rho}$ is k-wise independent if any k coordinates of $\boldsymbol{\rho}$ are independent. For a function $f : \{0,1\}^N \to \{0,1\}$, we use $f\restriction_\rho$ to denote the function $\{0,1\}^{|\rho^{-1}(*)|} \to \{0,1\}$ obtained by restricting f according to ρ in the natural way.

We need the following lemma stating that one can sample from a k-wise independent random restriction with a short seed, and moreover all restrictions have a small circuit description.

▶ **Lemma 12** ([24, 55]). *There exists a q-regular k-wise independent random restriction $\boldsymbol{\rho}$ distributed over $\rho : [N] \to \{0, 1, *\}$ samplable with $O(k \cdot \log(N) \log(1/q))$ bits. Furthermore, each output coordinate of the random restriction can be computed in time polynomial in the number of random bits.*

2.7 Technical Results

▶ **Lemma 13** (Hoeffding's inequality). *Let $X_1, \ldots, X_n$ be independent random variables such that $0 \le X_i \le 1$ for every $i \in [n]$. Let $X = \sum_{i=1}^n X_i$. Then, for any $\varepsilon > 0$, we have*

$$\Pr\{|X - \boldsymbol{E}X| \ge \varepsilon n\} \le 2\exp(-2\varepsilon^2 n)$$

3 Magnification Frontiers

3.1 EXP $\not\subseteq$ NC^1 and AC^0-XOR Lower Bounds for MKtP

In this Section, we present the proofs of the new results stated in HM Frontier A. Recall that $\mathsf{Kt}(x)$ is defined as the minimum over $|M| + \log t$ such that a program M outputs x in t steps. For thresholds $\theta, \theta' \colon \mathbb{N} \to \mathbb{N}$, we denote by $\mathsf{MKtP}[\theta(N), \theta'(N)]$ the promise problem whose YES instances consist of the strings $x \in \{0,1\}^N$ such that $\mathsf{Kt}(x) \le \theta(N)$ and NO instances consist of the strings such that $\mathsf{Kt}(x) > \theta'(N)$.

We start with the hardness magnification theorem of HM Frontier A1.

▶ **Theorem 14.** *There exists a constant c such that, for every large enough constant $d > 1$,*

$$\mathit{MKtP}[(\log N)^d, (\log N)^d + c \log N] \notin \mathit{AC}^0\text{-}\mathsf{XOR}[N^{1.01}] \quad \textit{implies} \quad \mathsf{EXP} \not\subseteq \mathit{NC}^1.$$

Proof. We prove the contrapositive. Assume that $\mathsf{EXP} \subseteq \mathsf{NC}^1$. First, recall that any N-bit-input polynomial-size NC^1 circuit can be converted into a depth-d' AC^0 circuit of size $2^{N^{O(1/d')}}$ for every positive integer constant d' (see, e.g., [2, Lemma 8.1]).

Oliveira, Pich, and Santhanam [39] showed that there exists a problem $L \in \mathsf{EXP}$ such that $\mathsf{MKtP}[\theta(N), \theta(N) + c \log N] \in \mathsf{AND}_{O(N)}\text{-}L_{O(\theta(N))}\text{-}\mathsf{XOR}$ for $\theta(N) \ge \log N$. (Here the subscript denotes the fan-in of a gate.) That is, the promise problem $\mathsf{MKtP}[\theta(N), \theta(N) + c \log N]$ can be computed by the following form of an L-oracle circuit: The output gate is an AND gate of fan-in $O(N)$, at the middle layer are L-oracle gates of fan-in $O(\theta(N))$, and at the bottom layer are XOR gates. Under the assumption that $\mathsf{EXP} \subseteq \mathsf{NC}^1$, we can replace L-oracle circuits with depth-d' AC^0 circuits of size $2^{(\log N)^{O(d/d')}}$, which is smaller than $N^{0.01}$ by choosing a constant d' large enough. In particular, we obtain a depth-$(d' + O(1))$ almost linear size AC^0 circuit with bottom XOR gates that computes $\mathsf{MKtP}[\theta(N), \theta(N) + c \log N]$. ◀

The rest of this section is devoted to proving the following AC^0 lower bound for MKtP, which establishes HM Frontier A4.

▶ **Theorem 15.** *For any $d = d(N)$, for some $\theta(N) = d \cdot \widetilde{O}(\log N)^3$ and any $\theta'(N) = N/\omega(\log N)^d$, it holds that $\mathsf{MKtP}[\theta(N), \theta'(N)] \notin \mathsf{AC}^0_d$.*

Note that Theorem 15 is only meaningful if $d = o(\log N / \log\log N)$, because otherwise the promise problem is not well-defined.

The idea of the proof is as follows: Trevisan and Xue [54] showed that there exists a pseudorandom restriction ρ of seed length $\mathsf{polylog}(N)$ that shrinks every polynomial-size depth-2 circuit into shallow decision trees. Moreover, the expected fraction of unrestricted variables $\rho^{-1}(*)$ is at least $p = \Omega(1/\log N)$. In particular, by composing d independent pseudorandom restrictions $\rho_1, \cdots, \rho_d$, every depth-d circuit can be turned into a constant function, while still leaving at least p^d-fraction of inputs unrestricted. The seed length required to sample d independent pseudorandom restrictions is at most $d \times \mathsf{polylog}(N)$, and thus $\mathsf{Kt}(0^N \circ \rho) \leq \mathsf{polylog}(N)$. We stress that the exponent of the seed length does not depend on d. Since the circuit hit with the pseudorandom restriction becomes a constant function, it cannot distinguish $0^N \circ \rho$ with $U_N \circ \rho$, i.e., the distribution where the unrestricted variables of ρ are replaced with the uniform distribution U_N. Assuming that there remain sufficiently many unrestricted inputs (e.g., $N/O(\log N)^d \gg \mathsf{polylog}(N)$), the latter distribution has a large Kt complexity, which is a contradiction to the fact that the circuit computes a gap version of MKtP.

We note that Cheraghchi, Kabanets, Lu, and Myrisiotis [16] used the pseudorandom restriction method in order to obtain an exponential-size AC^0 lower bound. A crucial difference in this work is that instead of optimizing the size of AC^0 circuits, we aim at minimizing the threshold θ of $\mathsf{MKtP}[\theta]$.

Following [54], in order to generate a random restriction $\rho \in \{0,1,*\}^N$ that leaves a variable unrestricted with probability 2^{-q}, we regard a binary string $w \in \{0,1\}^{(q+1)N}$ as a random restriction ρ_w. Specifically:

▶ **Definition 16.** *For a string $w \in \{0,1\}^{(q+1)N}$, we define a restriction $\rho_w \in \{0,1,*\}^N$ as follows: Write w as $(w_1, b_1) \cdots (w_N, b_N)$, where $w_i \in \{0,1\}^q$ and $b_i \in \{0,1\}$. For each $i \in [N]$, if $w_i = 1^q$ then set $\rho_w(i) := *$; otherwise, set $\rho_w(i) := b_i$.*

Note that this is defined so that $\Pr_w[\rho_w(i) = *] = 2^{-q}$ for every $i \in [N]$, when w is distributed uniformly at random.

Trevisan and Xue [54] showed that Håstad's switching lemma can be derandomized by using a distribution that fools CNFs. To state this formally, we need the following definitions. Define a t-width CNF as one which has at most t literals in each clause. We say that a distribution $\mathcal{D}$ over $\{0,1\}^n$ ε-fools a set of functions $\mathcal{S}_n$ over n variables if for every $f \in \mathcal{S}_n$, $|\Pr_{x \sim D}\{f(x) = 1\} - \Pr_{x \sim U_n}\{f(x) = 1\}| \leq \varepsilon$. Finally, define $\mathsf{DT}(f)$ as the depth of the smallest decision tree computing f.

▶ **Lemma 17** (Derandomized Switching Lemma [54, Lemma 7])**.** *Let φ be a t-width M-clause CNF formula over N inputs. Let $p = 2^{-q}$ for some $q \in \mathbb{N}$. Assume that a distribution $\mathcal{D}$ over $\{0,1\}^{(q+1)N}$ ε_0-fools $M \cdot 2^{t(q+1)}$-clause CNFs. Then,*

$$\Pr_{w \sim \mathcal{D}}[\mathsf{DT}(\varphi|_{\rho_w}) > s] \leq 2^{s+t+1}(5pt)^s + \varepsilon_0 \cdot 2^{(s+1)(2t+\log M)}.$$

▶ **Theorem 18** (Based on [54] and [53, Theorem 56])**.** *Let $s, M, d, N \in \mathbb{N}$ be positive integers. Let $p = 2^{-q}$ for some $q \in \mathbb{N}$ so that $1/128s \leq p < 1/64s$. Assume that there is a pseudorandom generator $G\colon \{0,1\}^r \to \{0,1\}^{(q+1)N}$ that ε_0-fools CNFs of size $M \cdot 2^s \cdot 2^{s(q+1)}$. Then, there exists a distribution $\mathcal{R}$ of random restrictions that satisfies the following:*

1. *For every circuit C of size M and depth d over N inputs,*

$$\Pr_{\rho\sim\mathcal{R}}[\mathsf{DT}(C\restriction_\rho) > s] \leq M \cdot \left(2^{-s+1} + \varepsilon_0 \cdot 2^{(s+1)(3s+\log M)}\right).$$

2. *For any parameter $\delta < 1$, with probability at least $1-N(\delta+d\varepsilon_0)$, the number of unrestricted variables in $[N]$ is at least $\lfloor N \cdot p^{d-1}/64\log(1/\delta)\rfloor$.*
3. *$\mathcal{R}$ can be generated by a seed of length dr in polynomial time.*

Proof. We apply the derandomized switching lemma (Lemma 17) d times. In the first iteration, we set $p := 1/64$ (and $q := 6$) and generate $\rho_{G(z)[1,\cdots,(6+1)N]}$. (Here we use the first $(6+1)N$ bits of $G(z)$ to generate $\rho_{G(z)}$.) This turns a circuit C of size M into a circuit whose bottom fan-in is at most s. For every other iteration i (where $i = 2, \cdots, d$), we set $p := 2^{-q}$ and turn a circuit C of depth $d-i+2$ into a circuit of depth $d-i+1$. Our final pseudorandom restriction $\rho \sim \mathcal{R}$ is defined by the composition of the d independent pseudorandom restrictions $\rho_{G(z_1)[1,\cdots,(6+1)N]}, \rho_{G(z_2)}, \cdots, \rho_{G(z_d)}$.

Our proof is essentially the same with [53], except that (1) we apply the switching lemma d times (instead of $d-1$) in order to turn depth-d circuits into shallow decision trees, and (2) in [54, 53], for the application of constructing a pseudorandom generator for AC^0, fixed bits of pseudorandom restrictions must be generated by using truly random bits, whereas in our case we generate all the bits by using G.

In more detail, for each $i \in [d]$, let M_i be the number of the gates at level i in C (i.e., the gates whose distance from the input gates is i). At the first iteration, we set $p := 1/64 = 2^{-6}$ and $q := 6$. We then generate $\rho^1 := \rho_{G(z_1)[1,\cdots,(6+1)N]}$ by choosing a seed $z_1 \sim \{0,1\}^r$ uniformly at random. We regard C as a depth-$(d+1)$ circuit of bottom fan-in 1, and apply Lemma 17 to each gate at level 1 (in the original circuit C). The probability that there exists a gate at level 1 in $C\restriction_{\rho^1}$ that cannot be computed by a decision tree of depth s is bounded above by

$$M_1 \cdot \left(2^{s+1+1}(5/64)^s + \varepsilon_0 \cdot 2^{(s+1)(2+\log M)}\right).$$

In the complement event, each gate at level 1 can be written as DNFs and CNFs of width s, and hence can be merged into some gate at level 2. Thus a circuit $C\restriction_{\rho^1}$ can be turned into a circuit of depth d and bottom fan-in s. Moreover, the number of gates at level 1 is bounded by $M \cdot 2^s$, which is an invariant preserved during the iterations.

For every other iteration i ($i = 2, \cdots, d$), we generate $\rho^i := \rho_{G(z_i)}$ by choosing a seed $z_i \sim \{0,1\}^r$ uniformly at random. Using the invariant that the number of gates at level $i-1$ is at most $M \cdot 2^s$, the probability that some gate at level i in $C\restriction_{\rho^1\cdots\rho^i}$ cannot be computed by a decision tree of depth s is bounded above by

$$M_i \cdot \left(2^{s+s+1}(5ps)^s + \varepsilon_0 \cdot 2^{(s+1)(2s+\log(M2^s))}\right).$$

In the complement event, every gate at level i can be written as width-s CNFs or DNFs of size 2^s, and hence can be merged into some gate at level $i+1$ (for $i < d$). At the last iteration (i.e., $i = d$), the circuit $C\restriction_{\rho^1\cdots\rho^d}$ can be written as a decision tree of depth s. We define the pseudorandom restriction ρ as $\rho^d \circ \cdots \circ \rho^1$. Item 3 is obvious from this construction.

Overall, the probability that $\mathsf{DT}(C\restriction_\rho) > s$ is at most $M \cdot (2^{-s+1} + \varepsilon_0 \cdot 2^{(s+1)(3s+\log M)})$. This completes the proof of Item 1.

To see Item 2, we divide N input bits into k disjoint blocks $T_1, \cdots, T_k$ of size at least t (and hence $k = \lfloor N/t \rfloor$), where t is a parameter chosen later. We claim that each block must contain at least one unrestricted variable in $\rho \sim \mathcal{R}$ with high probability (and hence $|\rho^{-1}(*)| \geq \lfloor N/t \rfloor$).

Fix any block $T = T_i$ for some $i \in [k]$. As in [53], one can easily observe that the condition that every variable in T is restricted can be checked by a CNF of size at most $|T|$ ($\leq N$). By a simple hybrid argument, the concatenation of d independent pseudorandom distributions $G(z_1), \cdots, G(z_d)$ $d\varepsilon_0$-fools CNFs (cf. [53, Corollary 55]). Therefore, the probability that every variable in T is restricted by $\rho \sim \mathcal{R}$ is bounded by $(1 - p^{d-1}/64)^t + d\varepsilon_0$, where the first term is an upper bound for the probability that every variable in T is restricted by a truly random restriction. Choosing $t = 64 \log(1/\delta)/p^{d-1}$ and using a union bound, the probability that some block T_i is completely fixed can be bounded above by $\lfloor N/t \rfloor \cdot (\delta + d\varepsilon_0)$, which completes the proof of Item 2. ◀

▶ **Corollary 19.** *For every circuit C of size M ($\geq N$) and depth d over N inputs, there exists a restriction ρ such that*

1. *$C\restriction_\rho$ is a decision tree of depth at most $s := 2 \log 8M$,*
2. *$|\rho^{-1}(*)| \geq N/O(\log M)^d$, and*
3. *$\mathsf{Kt}(\rho) \leq d \cdot \widetilde{O}((\log M)^3)$.*

Proof. Tal [53, Theorem 52] showed that there exists a polynomial-time pseudorandom generator G of seed length $r := \widetilde{O}(\log M_0 \cdot \log(M_0/\varepsilon_0))$ that ε_0-fools CNFs of size M_0. We set $M_0 := M \cdot 2^s \cdot 2^{s(q+1)}$, $s := 2 \log 8M$, and $\varepsilon_0 := 2^{-9s^2}$. Then the seed length r of G is at most $r = \widetilde{O}(\log M_0 \cdot \log(M_0/\varepsilon_0)) = \widetilde{O}(\log M \cdot (\log M)^2)$. Applying Theorem 18, the probability that $\mathsf{DT}(C\restriction_\rho) > s$ is bounded by $\frac{1}{2}$. Choosing $\delta = 1/8N$, we also have that the probability that $|\rho^{-1}(*)| < \lfloor N \cdot p^{d-1}/64 \log(1/\delta) \rfloor$ is at most $\frac{1}{4}$. Thus there exists some restriction ρ in the support of $\mathcal{R}$ such that $\mathsf{DT}(C\restriction_\rho) \leq s$ and $|\rho^{-1}(*)| \geq \Omega(N \cdot p^{d-1}/\log N) \geq N/O(\log M)^d$. ◀

Using the assumption that a circuit computes MKtP, we show that shallow decision trees must be a constant function.

▶ **Lemma 20.** *Let C be a circuit and ρ be a restriction such that $C\restriction_\rho$ is a decision tree of depth s. If $\mathsf{MKtP}[O(s \log N) + \mathsf{Kt}(\rho)] \subseteq C^{-1}(1)$, then $C\restriction_\rho \equiv 1$.*

Proof. We prove the contrapositive. Assume that $C\restriction_\rho \not\equiv 1$, which means that there is a path $\pi \colon [N] \to \{0, 1, *\}$ of a decision tree $C\restriction_\rho$ that assigns at most s variables so that $C\restriction_{\rho\pi} \equiv 0$. Note that $\mathsf{Kt}(\pi) \leq O(s \log N)$, because one can specify each restricted variable of π by using $O(\log N)$ bits. Thus we have $\mathsf{Kt}(0^N \circ \pi \circ \rho) \leq O(s \log N) + \mathsf{Kt}(\rho)$. On the other hand, $C(0^N \circ \pi \circ \rho) = C\restriction_{\rho\pi}(0^N) = 0$. Therefore, we obtain $\mathsf{MKtP}[O(s \log N) + \mathsf{Kt}(\rho)] \not\subseteq C^{-1}(1)$. ◀

Now we are ready to prove the main result of this section.

Proof of Theorem 15. Assume, by way of contradiction, that there is a circuit C of size $M := N^{O(1)}$ and depth d that computes $\mathsf{MKtP}[d \cdot \widetilde{O}(\log N)^3, N/\omega(\log N)^d]$. Using Corollary 19, we take a restriction ρ such that $C\restriction_\rho$ is a decision tree of depth $s = O(\log N)$. By Lemma 20, we have $C\restriction_\rho \equiv 1$, under the assumption that $O(s \log N) + \mathsf{Kt}(\rho) \leq \theta(N)$, which is satisfied by choosing $\theta(N)$ large enough. Now, by counting the number of inputs accepted by $C\restriction_\rho$, we obtain

$$2^{N/O(\log N)^d} \leq 2^{|\rho^{-1}(*)|} = |(C\restriction_\rho)^{-1}(1)| \leq 2^{\theta'(N)+1},$$

where, in the last inequality, we used the fact that the number of strings whose Kt complexity is at most $\theta'(N)$ is at most $2^{\theta'(N)+1}$. However, the inequality contradicts the choice of $\theta'(N)$. ◀

3.2 NQP $\not\subseteq$ NC^1 and Formula-XOR or GapAND-Formula for MCSP

This section is devoted to proving HM Frontier B1 and HM Frontier D1. In fact, we provide two different proofs of HM Frontier B1, one based on [42], another one based on [10].

In both proofs, the hardness magnification is achieved by constructing an oracle circuit for MCSP. The most interesting part of the first proof is that it gives a *conditional* construction assuming QP $\subseteq$ P/poly. While the oracle circuit construction can be made *unconditional* (as in the second proof), it illustrates a potentially more applicable approach: proving the hardness magnification theorem while assuming the target circuit lower bound is false (i.e., NQP $\subseteq$ NC^1).

3.2.1 Reduction Based Approach from [42]

In the initial magnification theorem [42, Theorem 1], approximate MCSP was shown to admit hardness magnification phenomena. Here we present a similar hardness magnification theorem for a worst-case version of MCSP.

A natural way of reducing worst-case MCSP to approximate MCSP is to apply error-correcting codes. Error-correcting codes map a hard Boolean function to a Boolean function which is hard on average. A problem with this approach is that error-correcting codes do not guarantee that an easy Boolean function will be mapped to an easy Boolean function. Our main idea is to enforce the latter property with an extra assumption QP $\subseteq$ P/poly. Here, QP denotes TIME$[n^{\log^{O(1)} n}]$. Similarly, NQP will stand for NTIME$[n^{\log^{O(1)} n}]$.

We will use the following explicit error-correcting code.

▶ **Theorem 21** (Explicit linear error-correcting codes [30, 48]). *There exists a sequence $\{E_N\}_{N\in\mathbb{N}}$ of error-correcting codes $E_N\colon \{0,1\}^N \to \{0,1\}^{M(N)}$ with the following properties:*

- *$E_N(x)$ can be computed by a uniform deterministic algorithm running in time* poly(N).
- *$M(N) = b \cdot N$ for a fixed $b \geq 1$.*
- *There exists a constant $\delta > 0$ such that any codeword $E_N(x) \in \{0,1\}^{M(N)}$ that is corrupted on at most a δ-fraction of coordinates can be uniquely decoded to x by a uniform deterministic algorithm D running in time* poly$(M(N))$.
- *Each output bit is computed by a parity function: for each input length $N \geq 1$ and for each coordinate $i \in [M(N)]$, there exists a set $S_{N,i} \subseteq [N]$ such that for every $x \in \{0,1\}^N$,*

$$E_N(x)_i = \bigoplus_{j\in S_{N,i}} x_j.$$

Under the assumption that QP $\subseteq$ P/poly, we present an efficient reduction from worst-case MCSP to approximate MCSP: Given the truth table of a function f, we simply map it to $E_N(\mathsf{tt}(f))$. The following lemma establishes the correctness of this reduction.

▶ **Lemma 22** (Reducing worst-case MCSP to approximate MCSP). *Assume QP $\subseteq$ P/poly. Then the error-correcting code E_N from Theorem 21 satisfies the following:*

1. *$f_n \in$ Circuit$[2^{n^{1/3}}] \Rightarrow E_N(tt(f_n)) \in$ Circuit$[2^{\sqrt{m}}]$,*[14]
2. *$f_n \notin$ Circuit$[2^{n^{2/3}}] \Rightarrow E_N(tt(f_n))$ is hard to $(1-\delta)$-approximate by $2^{\sqrt{m}}$-size circuits,*

where $m = \Theta(n)$.

[14] Here we identify $E_N(\mathsf{tt}(f_n))$ with the function whose truth table is $E_N(\mathsf{tt}(f_n))$.

Proof. For the first implication we consider the map

$$C, i \mapsto E_N(\mathsf{tt}(C))_i$$

where C is a circuit with n inputs and size $2^{n^{1/3}}$, $i \in \{0,1\}^m$, and $m = \log |E_N|$. The map takes an input of length $2^{O(n^{1/3})}$, and is computable in time $2^{O(n)}$; hence the map is in $\mathsf{QP} \subseteq \mathsf{P/poly}$. Thus, there exists a circuit F of size $2^{O(n^{1/3})}$ that, taking the description of a circuit C of size $2^{n^{1/3}}$ and $i \in \{0,1\}^m$ as input, outputs the ith bit of $E_N(\mathsf{tt}(C))$. Therefore if f_n is computed by a circuit C of size $2^{n^{1/3}}$, the function $i \mapsto E_N(\mathsf{tt}(f_n))_i$ is computable by a circuit $F(C,\text{-})$ of size $2^{O(n^{1/3})} < 2^{\sqrt{m}}$.

The second implication is obtained in a similar way by considering the map

$$C, i \mapsto D_N(\mathsf{tt}(C))_i$$

where C is a circuit with $m = \log |E_N|$ inputs and size $2^{\sqrt{m}}$, $i \in \{0,1\}^n$ and D_N is an efficient decoder of E_N. The new map is computable in time $2^{O(m)}$ and again is in $\mathsf{QP} \subseteq \mathsf{P/poly}$. Therefore if $E_N(\mathsf{tt}(f_n))$ is $(1-\delta)$-approximated by a circuit C of size $2^{\sqrt{m}}$, f_n is computable by a circuit of size $2^{O(\sqrt{m})} < 2^{n^{2/3}}$. ◀

Since the error-correcting code of Theorem 21 can be computed by using one layer of XOR gates, we obtain the following corollary.

▶ **Corollary 23.** *If $\mathsf{QP} \subseteq \mathsf{P/poly}$, then $\mathsf{MCSP}[2^{n^{1/3}}, 2^{n^{2/3}}]$ is reducible to $\mathsf{MCSP}[(2^{\sqrt{n}}, 0), (2^{\sqrt{n}}, \delta)]$ by using a many-one reduction computed by a linear-size circuit of* XOR *gates.*

We are ready to prove the main result of this section:

▶ **Theorem 24** (Magnification for worst-case MCSP via error-correcting codes)**.** *Assume that $\mathsf{MCSP}[2^{n^{1/3}}, 2^{n^{2/3}}] \notin \mathsf{Formula}\text{-}\mathsf{XOR}[N^{1+\varepsilon}]$ for some constant $\varepsilon > 0$. Then either $\mathsf{QP} \not\subseteq \mathsf{P/poly}$ or $\mathsf{NP} \not\subseteq \mathsf{NC}^1$. In particular, $\mathsf{NQP} \not\subseteq \mathsf{NC}^1$.*

Proof. We prove the contrapositive. Assume that $\mathsf{QP} \subseteq \mathsf{P/poly}$ and $\mathsf{NP} \subseteq \mathsf{NC}^1$. [42, Lemma 16] shows that $\mathsf{NP} \subseteq \mathsf{NC}^1$ implies $\mathsf{MCSP}[(2^{\sqrt{n}}, 0), (2^{\sqrt{n}}, \delta)] \in \mathsf{Formula}[N^{1+\varepsilon}]$ for any constant $\varepsilon > 0$. By combining this with Corollary 23, we obtain that $\mathsf{MCSP}[2^{n^{1/3}}, 2^{n^{2/3}}] \in \mathsf{Formula}\text{-}\mathsf{XOR}[O(N^{1+\varepsilon})]$. ◀

3.2.2 Kernelization Based Approach from [10]

Now we give another proof of HM Frontier B1 by adapting techniques from [10]. In fact, the following proof implies (under a straightforward adjustment of parameters) both HM Frontier B1 and HM Frontier D1.

▶ **Theorem 25** (Magnification for worst-case MCSP via kernelization for GapAND-Formula-XOR)**.** *Assume that $\mathsf{MCSP}[2^{n^{1/3}}] \notin \mathsf{GapAND}_{O(N)}\text{-}\mathsf{Formula}\text{-}\mathsf{XOR}[N^{\varepsilon}]$ for some constant $\varepsilon > 0$. Then $\mathsf{NQP} \not\subseteq \mathsf{NC}^1$.*

Proof Sketch. The following proof is just an adaption of Theorem 3.4 of [10].

Let $N = 2^n$ and $s = 2^{n^{1/3}} = 2^{(\log N)^{1/3}}$. Let $S = \mathsf{MCSP}[2^{n^{1/3}}]^{-1}(1)$ (that is, all yes instances of $\mathsf{MCSP}[2^{n^{1/3}}]$ on inputs of length N), and $m = |S|$. We have that $m \leq s^{O(s)}$. Let E_N be the error correcting code from Theorem 21. Recall that E_N maps from $\{0,1\}^N$ to $\{0,1\}^{b \cdot N}$ for a constant b.

Let $T = c_1 \cdot \log m$ for a large enough constant c_1. Suppose we pick T random indexes $I = (i_1, i_2, \ldots, i_T)$ from $[b \cdot N]$ independently and uniformly at random. Given $x \in \{0,1\}^N$, let $H_I(x) := (E_N(x)_{i_1}, E_N(x)_{i_2}, \ldots, E_N(x)_{i_T})$.

By a Chernoff bound and a union bound, we can see that with high probability over random choices of I, all inputs from S are mapped into distinct strings in $\{0,1\}^T$ by H_I. We fix such a good collection of indexes I_{good}.

Now, consider the following language

$$L_{\mathsf{check}} : [b \cdot N]^T \times \{0,1\}^T \times [b \cdot N] \times \{0,1\} \to \{0,1\},$$

which takes as inputs I (hash function coordinates), w (hash value), i (index), and z (check-bit). $L_{\mathsf{check}}(I, w, i, z)$ guesses an input $y \in \{0,1\}^N$, and accepts if $H_I(y) = w$, $\mathsf{MCSP}[2^{n^{1/3}}](y) = 1$, and $E_N(y)_i = z$. It is easy to see that L_{check} is in NQP.

Given $x \in \{0,1\}^N$, we claim that $\mathsf{MCSP}[2^{n^{1/3}}](x) = 1$ iff $L_{\mathsf{check}}(I_{\mathsf{good}}, H_{I_{\mathsf{good}}}(x), i, E_N(x)_i) = 1$ for all $i \in [b \cdot N]$.

1. When $\mathsf{MCSP}[2^{n^{1/3}}](x) = 1$, on the particular guess $y = x$, $L_{\mathsf{check}}(I_{\mathsf{good}}, H_{I_{\mathsf{good}}}(x), i, E_N(x)_i)$ accepts for all $i \in [b \cdot N]$.
2. When $\mathsf{MCSP}[2^{n^{1/3}}](x) = 0$, we set $z = H_{I_{\mathsf{good}}}(x)$. By our choice of I_{good}, there is at most one x' satisfying both $\mathsf{MCSP}[2^{n^{1/3}}](x') = 1$ and $H_{I_{\mathsf{good}}}(x') = z$. If there is no such x', then all $L_{\mathsf{check}}(I_{\mathsf{good}}, H_{I_{\mathsf{good}}}(x), i, x_i)$ reject. Otherwise, we have $x \neq x'$. Let i be an index such that $E_N(x)_i \neq E_N(x')_i$. Then $L_{\mathsf{check}}(I_{\mathsf{good}}, H_{I_{\mathsf{good}}}(x), i, E_N(x)_i)$ rejects.

Moreover, in the second case, $L_{\mathsf{check}}(I_{\mathsf{good}}, H_{I_{\mathsf{good}}}(x), i, E_N(x)_i)$ indeed rejects at least for a constant fraction of $i \in [b \cdot N]$, since $E_N(x)$ is an error correcting code,

Now suppose $\mathsf{NQP} \subseteq \mathsf{NC}^1$ for the sake of contradiction. Since $H_{I_{\mathsf{good}}}(x)$ can be computed by $T = N^{o(1)}$ many XOR gates (I_{good} is hardwired into the circuit), we can construct $b \cdot N$ $\mathsf{Formula\text{-}XOR}[N^{o(1)}]$ circuits $C_1, C_2, \ldots, C_{b \cdot N}$, such that if $\mathsf{MCSP}[2^{n^{1/3}}](x) = 1$ then $C_i(x) = 1$ for all x, and otherwise $C_i(x) = 0$ for a constant fraction of i's.

By a simple error reduction via random sampling, we construct $m = O(N)$ $\mathsf{Formula\text{-}XOR}[N^{o(1)}]$ circuits $D_1, D_2, \ldots, D_m$, such that if $\mathsf{MCSP}[2^{n^{1/3}}](x) = 1$ then $D_i(x) = 1$ for all x, and otherwise $D_i(x) = 0$ for at least a 0.9 fraction of inputs. Hence, we have $\mathsf{MCSP}[2^{n^{1/3}}] \in \mathsf{GapAND}_{O(N)}\text{-}\mathsf{Formula\text{-}XOR}[N^{o(1)}]$, a contradiction to the assumption. ◀

▶ Remark 26. We note that $\mathsf{GapAND}_{O(N)}\text{-}\mathsf{Formula\text{-}XOR}[N^{\varepsilon}]$ circuits are a special case of both $\mathsf{Formula\text{-}XOR}[N^{1+\varepsilon}]$ circuits and $\mathsf{GapAND}_{O(N)}\text{-}\mathsf{Formula}[N^{2+\varepsilon}]$ circuits. Therefore, the above proof implies both HM Frontier B1 and HM Frontier D1.

3.3 NP $\not\subseteq$ NC^1 and Almost-Formula Lower Bounds for MCSP

Recall that near-quadratic *formula* lower bounds are known for $\mathsf{MCSP}[2^{n^{o(1)}}, 2^{n^{o(1)}}]$. On the other hand, a hardness magnification obtained by a super efficient construction of anticheckers established in [39] states that $\mathsf{NP} \subseteq \mathsf{P/poly}$ implies almost linear-size circuits for a worst-case version of parameterized $\mathsf{MCSP}[2^{n^{o(1)}}, 2^{n^{o(1)}}]$. Consequently, if we could make the hardness magnification work for formulas, $\mathsf{NP} \not\subseteq \mathsf{NC}^1$ would follow. We make a step in this direction by showing that $\mathsf{NP} \subseteq \mathsf{NC}^1$ implies the existence of almost-formulas of almost linear size solving the worst-case $\mathsf{MCSP}[2^{n^{o(1)}}, 2^{n^{o(1)}}]$, cf. Theorem 29. This is established by a more detailed analysis of the proof from [39] extended with an application of the Valiant-Vazirani Isolation Lemma (cf. [6, Lemma 17.19]) in the process of selecting anticheckers. We also observe that almost-formulas of sub-quadratic size cannot solve PARITY, cf. Theorem 30. These results yield HM Frontier C1 and HM Frontier C3.

We start the presentation with a lemma needed to derive HM Frontier C1.

▶ **Lemma 27** (Anticheckers). *Assume* $NP \subseteq NC^1$. *Then for any* $\lambda \in (0,1)$ *there are circuits* $\{C_{2^n}\}_{n=1}^{\infty}$ *of size* $2^{n+O(n^\lambda)}$ *which given* $\mathsf{tt}(f) \in \{0,1\}^N$, *outputs* $2^{O(n^\lambda)}$ n*-bit strings* $y_1, \ldots, y_{2^{O(n^\lambda)}}$ *together with bits* $f(y_1), \ldots, f(y_{2^{O(n^\lambda)}})$ *forming a set of anticheckers for* f, *i.e. if* f *is hard for circuits of size* 2^{n^λ} *then every circuit of size* $2^{n^\lambda}/2n$ *fails to compute* f *on one of the inputs* $y_1, \ldots, y_{2^{O(n^\lambda)}}$. *Moreover, each* $y_i, f(y_i)$ *is generated by a subcircuit of* C_{2^n} *with inputs* $y_1, \ldots, y_{i-1}, f(y_1), \ldots, f(y_{i-1}), \mathsf{tt}(f)$ *whose only gates with fanout* > 1 *are* $y_1, \ldots, y_{i-1}, f(y_1), \ldots, f(y_{i-1})$.

Proof. This proof follows [39]. Our contribution here is the "moreover" part, but we also give a more succinct self-contained proof. For each Boolean function f the desired set of anticheckers is known to exist, the only problem is to find it with a circuit of the desired size and formula-like form. In order to do so, we will simulate the proof of the existence of anticheckers, but make the involved counting constructive by using linear hash functions and the assumption $\mathsf{NP} \subseteq \mathsf{NC}^1$. Additionally, for the "moreover" part of the lemma, we will employ the Valiant-Vazirani Isolation Lemma (cf. [6, Lemma 17.19]) in the process of selecting good anticheckers.

Let $\lambda \in (0,1)$ and f be a Boolean function with n inputs hard for circuits of size 2^{n^λ}. For j n-bit strings $y_1, \ldots, y_j$ and $s \in [0,1]$, define a predicate

$$P_f(y_1, \ldots, y_j)[s] \text{ iff } \leq s \text{ fraction of all circuits of size } 2^{n^\lambda}/2n \text{ compute } f \text{ on } y_1, \ldots, y_j.$$

Further, let $R_f(y_1, \ldots, y_j)$ be the number of circuits of size $2^{n^\lambda}/2n$ which do not make any error on $y_1, \ldots, y_j$ when computing f. Note that P_f and R_f depend on j values of f, not on the whole $\mathsf{tt}(f)$, but for simplicity we do not display them.

Suppose that given $\mathsf{tt}(f)$ we already generated $y_1, \ldots, y_{i-1}, f(y_1), \ldots, f(y_{i-1})$ such that $P_f(y_1, \ldots, y_{i-1})[(1-1/4n)^{i-1}]$ holds. For $i = 1$ the generated set is empty. We want to find $y_i, f(y_i)$ such that $P_f(y_1, \ldots, y_i)[(1-1/4n)^i]$. In order to do so, we will construct a formula $F(y_1, \ldots, y_i, f(y_1), \ldots, f(y_i))$ of size $2^{O(n^\lambda)}$ (if $i \leq 2^{O(n^\lambda)}$) such that under the assumption $R_f(y_1, \ldots, y_{i-1}) \geq 2n^2$,

$$F(y_1, \ldots, y_i, f(y_1), \ldots, f(y_i)) = 1 \quad \Rightarrow \quad P_f(y_1, \ldots, y_i)[(1-1/4n)^i]$$

$$P_f(y_1, \ldots, y_{i-1})[(1-1/4n)^{i-1}] \quad \Rightarrow \quad \exists y_i, F(y_1, \ldots, y_i, f(y_1), \ldots, f(y_i)) = 1.$$

Assume for now that we already have such a formula F. We firstly show how to find $y_i, f(y_i)$ given F by an exhaustive search through all n-bit strings in combination with Valiant-Vazirani Lemma.

Consider a $2^{O(n^\lambda)}$-size formula $F^{r,h}(y_1, \ldots, y_{i-1}, z, f(y_1), \ldots, f(y_{i-1}), f(z))$ computing the following predicate

$$F(y_1, \ldots, y_{i-1}, z, f(y_1), \ldots, f(y_{i-1}), f(z)) \wedge h(z) = 0^r \tag{1}$$

where $z \in \{0,1\}^n$, $r \leq n+2$ and $h \in \mathcal{H}_{n,r}$ for a pairwise independent efficiently computable hash function collection $\mathcal{H}_{n,r}$ from $\{0,1\}^n$ to $\{0,1\}^r$. Formula $F^{r,h}$ exists since $\mathsf{NP} \subseteq \mathsf{NC}^1$. By Valiant-Vazirani Lemma, for fixed $y_1, \ldots, y_{i-1}, f(y_1), \ldots, f(y_{i-1})$, if h is chosen randomly from $\mathcal{H}_{n,r}$ and r randomly from $\{2, \ldots, n+1\}$, then with probability $\geq 1/8n$, there is a unique z satisfying (1). Therefore, the probability that none of $2^{O(n^\lambda)}$ many randomly chosen tuples r, h guarantees a unique solution is $< (1-1/8n)^{2^{O(n^\lambda)}} \leq 1/2^{2^{O(n^\lambda)}/8n}$. That

is, there exist a set $\mathcal{R}$ of $2^{O(n^\lambda)}$ tuples r, h such that for each $y_1, \ldots, y_{i-1}, f(y_1), \ldots, f(y_{i-1})$, at least one tuple r, h from $\mathcal{R}$ will guarantee a unique solution. Consequently, for each $y_1, \ldots, y_{i-1}, f(y_1), \ldots, f(y_{i-1})$ for at least one $r, h \in \mathcal{R}$ the following $2^{n+O(n^\lambda)}$-size formula

$$\bigvee_{k=1,\ldots,2^n} (b^k_j \wedge F^{r,h}(y_1, \ldots, y_{i-1}, b^k, f(y_1), \ldots, f(y_{i-1}), f(b^k)),$$

where b^k_j is the jth bit of the kth n-bit string b^k (in the lexicographic order), outputs the jth bit of a good antichecker y_i. Since $\mathsf{NP} \subseteq \mathsf{NC}^1$, we can select the right y_i from the $2^{O(n^\lambda)}$ candidate strings corresponding to tuples r, h from $\mathcal{R}$ by applying a formula of size $2^{O(n^\lambda)}$ on top of them. Having y_i, a formula of size $poly(n)2^n$ with access to $\mathsf{tt}(f)$ can generate $f(y_i)$.

Iteratively, a circuit of size $2^{n+O(n^\lambda)}$ will generate $y_1, \ldots, y_{2^{O(n^\lambda)}}, f(y_1), \ldots, f(y_{2^{O(n^\lambda)}})$ such that $P_f(y_1, \ldots, y_{2^{O(n^\lambda)}})[(1-1/4n)^{2^{O(n^\lambda)}}]$ as long as $R_f(y_1, \ldots, y_{2^{O(n^\lambda)}}) \geq 2n^2$. Deciding whether $R_f(y_1, \ldots, y_i) \geq 2n^2$ is in $\mathsf{NP} \subseteq \mathsf{NC}^1$ (on input $y_1, \ldots, y_i, f(y_1), \ldots, f(y_i), 1^{2^{n^\lambda}}$), so there are formulas of size $2^{O(n^\lambda)}$ for it. Since $(1-1/4n)^{2^{O(n^\lambda)}} \leq 1/2^{2^{O(n^\lambda)}/4n}$, we reach $R_f(y_1, \ldots, y_i) < 2n^2$ with $i \leq 2^{O(n^\lambda)}$. When this happens, the remaining $< 2n^2$ circuits of size $2^{n^\lambda}/2n$ can be generated by an $\mathsf{NP}^{\mathsf{coNP}}$ algorithm, and since $\mathsf{NP} \subseteq \mathsf{NC}^1$, by a formula of size $2^{O(n^\lambda)}$. Finally, for each of the remaining circuits we can find an n bit string witnessing its error exhaustively by a formula of size $2^{n+O(n^\lambda)}$. Altogether, the desired antichecker $y_1, \ldots, y_{2^{O(n^\lambda)}}$ with bits $f(y_1), \ldots, f(y_{2^{O(n^\lambda)}})$ will be generated by a circuit of size $2^{n+O(n^\lambda)}$. Note that this circuit will have the desired formula-like structure because its only gates with fanout bigger than 1 are those computing tuples $y_i, f(y_i)$.

▷ Claim 28. If $P_f(y_1, \ldots, y_{i-1})[(1-1/4n)^{i-1}]$ and $R_f(y_1, \ldots, y_{i-1}) \geq 2n^2$, then for some y_i, $P_f(y_1, \ldots, y_i)[(1-1/4n)^{i-1}(1-1/2n)]$.

Claim 28 is proved by a standard counting argument, cf. [39, Claim 22]. Observe that with Claim 28 we can construct the desired formula F. Here we employ approximate counting with linear hash functions: if $X \subseteq \{0,1\}^m$ is a set of size s, there are matrices $A_1, \ldots, A_{\log(4s^c)}$ such that each A_j defines a linear function mapping a Cartesian power X^c to $(s(1+\varepsilon))^c/\log(4s^c)$, for $c = 2(\varepsilon^{-1}(\log\log s + \log\varepsilon^{-1}))$. Moreover, for each A_j there is $X^c_j \subseteq X^c$ satisfying $\forall x \in X^c_j \forall x' \in X^c$ $(x \neq x' \to A_j(x) \neq A_j(x'))$, and $\bigcup_j X^c_j = X^c$. Mapping $x \in X^c$ to $A_j(x)$ in the jth block of size $(s(1+\varepsilon))^c/\log(4s^c)$, for the first A_j with $x \in X^c_j$, thus defines an injection from X^c to $(s(1+\varepsilon))^c$ which witnesses that the size of X is $\leq s(1+\varepsilon)$. See e.g. [28, Section 3, 2nd paragraph] for details.

Therefore, once we have $P_f(y_1, \ldots, y_i)[(1-1/4n)^{i-1}(1-1/2n)]$ we can conclude that there are matrices $A_1, \ldots, A_{2^{O(n^\lambda)}}$ defining an injective mapping of a Cartesian power (with exponent of rate $poly(n)$) of the set of all circuits of size $2^{n^\lambda}/2n$ that compute f on $y_1, \ldots, y_i$ to the same Cartesian power of $(1-1/4n)^{i-1}(1-1/2n)(1+1/4n) \leq (1-1/4n)^i$ fraction of the set of all circuits of size $2^{n^\lambda}/2n$. The existence of such matrices, not only witnesses $P_f(y_1, \ldots, y_i)[(1-1/4n)^i]$ but is also an $\mathsf{NP}^{\mathsf{coNP}}$ property, and since $\mathsf{NP} \subseteq \mathsf{NC}^1$, decidable by a formula F of size $2^{O(n^\lambda)}$. ◀

▶ **Theorem 29** (Improved magnification via anticheckers). *Assume that MCSP$[2^{n^{1/2}}/2n, 2^{n^{1/2}}]$ is hard for circuits C (with 2^n inputs) of size $2^{n+O(n^{1/2})}$ with the following form. Given $\mathsf{tt}(f)$, subcircuits of C generate $y_1, \ldots, y_{2^{O(n^{1/2})}}, f(y_1), \ldots, f(y_{2^{O(n^{1/2})}})$ so that each $y_i, f(y_i)$ is generated by a subcircuit of C with inputs $y_1, \ldots, y_{i-1}, f(y_1), \ldots, f(y_{i-1}), \mathsf{tt}(f)$ whose only gates with fanout $>$ 1 are $y_1, \ldots, y_{i-1}, f(y_1), \ldots, f(y_{i-1})$. Having $y_1, \ldots, y_{2^{O(n^{1/2})}}, f(y_1), \ldots, f(y_{2^{O(n^{1/2})}})$, C applies a formula of size $2^{O(n^{1/2})}$ on top of these gates.*

Then $\mathsf{NP} \not\subseteq \mathsf{NC}^1$.

Proof. If $\mathsf{NP} \subseteq \mathsf{NC}^1$, then $\mathsf{MCSP}[2^{n^{1/2}}/2n, 2^{n^{1/2}}]$ can be solved by circuits of size $2^{n+O(n^{1/2})}$ of the required form: given a Boolean function f, apply Lemma 27 to generate a set of its anticheckers $y_1, \ldots, y_{2^{O(n^{1/2})}}$ together with bits $f(y_1), \ldots, f(y_{2^{O(n^{1/2})}})$ and using $\mathsf{NP} \subseteq \mathsf{NC}^1$ decide whether f is hard for circuits of size $2^{n^{1/2}}/2n$ on $y_1, \ldots, y_{2^{O(n^{1/2})}}$. ◀

Note that circuits from the assumption of hardness magnification via anticheckers, Theorem 29, are $2^{O(n^{1/2})}$-almost formulas of almost linear size which gives us HM Frontier C1. We can now complement it with HM Frontier C3.

Consider an s-almost formula. Each gate G of F with fanout larger than 1 is computed by a formula with inputs being either the original inputs of F or gates of F with fanout larger than 1. We call any maximal formula of this form a *principal* formula of G.

▶ **Theorem 30.** *PARITY* $\notin n^{\varepsilon}$-almost-*Formula*$[n^{2-9\varepsilon}]$, *if* $\varepsilon < 1$.

Proof Sketch. For the sake of contradiction, assume PARITY has n^{ε}-almost formulas of size $n^{2-9\varepsilon}$. Since there are only n^{ε} gates of fanout > 1, we can replace these gates by appropriate constants and obtain formulas F_n of size $n^{2-8\varepsilon}$ computing PARITY with probability $\geq 1/2 + 1/2^{n^{\varepsilon}}$. In more detail, each formula F_n checks if the principal formulas compute the fixed constants. If this is the case, then F_n outputs the output of the original almost-formula (since gates with fan-out larger than 1 are fixed, the output can be computed by a formula). Otherwise, F_n outputs a fixed constant, whichever is better on the majority of the remaining inputs. This does not increase the size of the resulting formula F_n by more than a constant factor. As pointed out by Komargodski-Raz [32], each boolean function f on n input bits can be approximated by a real polynomial of degree $O(t\sqrt{L(f)}\frac{\log n}{\log\log n})$ up to a point-wise additive error of 2^{-t}, and this can be shown to imply that each formula of size $o((n/t)^2(\log\log n/\log n)^2)$ computes PARITY over n input bits with probability at most $1/2 + 1/2^{t+O(1)}$ (for large enough t). Taking $t = n^{2\varepsilon}$ we get a contradiction. ◀

3.4 NP $\not\subseteq$ NC^1 and AC^0 Lower Bounds for $(n-k)$-Clique

In this Section, we discuss the proofs of some statements claimed in HM Frontier E from Section 1.1. Recall that we consider graphs on n vertices that are described in the adjacency matrix representation. The input graph is therefore represented using $m = \Theta(n^2)$ bits. We begin with the proof of the magnification result in HM Frontier E1.

▶ **Proposition 31.** *Let* $k(n) = (\log n)^C$ *for some constant* C. *If there exists* $\varepsilon > 0$ *such that for every depth* $d \geq 1$, $(n-k)$-Clique $\notin \mathsf{AC}^0_d[m^{1+\varepsilon}]$, *then* NP $\not\subseteq$ NC^1.

Proof. We use a straightforward reduction to the magnification theorem for k-Vertex-Cover established in [42, Theorem 7]. (We state Proposition 31 in a slightly weaker form just for simplicity.) Indeed, a graph G on n vertices has a vertex cover of size $\leq k$ if and only if G has an independent set of size $\geq n-k$. In turn, the latter is true if and only if the complement graph $\overline{G}$ has a clique of size $\geq n-k$. Therefore, by negating input literals, the complexities of $(n-k)$-Clique and k-Vertex-Cover are equivalent with respect to AC^0 circuits. For this reason, the hardness magnification theorem of [42] immediately implies Proposition 31. ◀

We state below conditional and unconditional lower bounds on the complexity of detecting very large cliques. The next proposition implies the lower bound claimed in HM Frontier E4.

▶ **Proposition 32** ([4]; see also [29, Section 9.2])**.** *For* $k(n) \leq n/2$, *every monotone circuit for* $(n-k)$-Clique *requires* $2^{\Omega(k^{1/3})}$ *gates.*

Interestingly, the problem can be solved by (bounded depth) polynomial size monotone circuits if $k \leq \sqrt{\log n}$ [4].

Finally, by the observation employed in the proof of Proposition 31, for non-monotone computations the complexities of detecting large cliques and small vertex covers are equivalent. A consequence of this is that one can show the following result, which implies the statement in HM Frontier E2.

▶ **Proposition 33.** *If* ETH *for non-uniform circuits holds, then* $(n-k)$-Clique $\notin$ P/poly *as long as* $\omega(\log n) \leq k \leq n/2$.

Indeed, under ETH the k-Vertex-Cover problem cannot be solved in time $2^{o(k)} \cdot \mathsf{poly}(m)$ (see [26] and [17, Theorem 29.5.9]). Further discussion on the conditional hardness of k-Vertex-Cover that also applies to $(n-k)$-Clique appears in [42].

4 Hardness Magnification and Natural Proofs

4.1 Equivalences

The main contribution of this section is new hardness magnification results showing non-learnability of circuit classes from slightly super-linear lower bounds for the approximate version of MCSP and the gap version of MCSP. We then use these magnification results to establish a series of equivalences.

▶ **Lemma 34** (Hardness Magnification for Learnability from Lower Bounds for Approximate MCSP)**.** *Let* $s, t : \mathbb{N} \to \mathbb{N}$ *be size functions such that* $n \leq s(n) \leq t(n)$ *and* ε, δ *be parameters such that* $\varepsilon < 1/2$, $0 \leq \delta \leq 1/9$. *If for infinitely many input lengths* $N = 2^n$, $\mathit{MCSP}[(s,0),(t,\varepsilon)] \notin \mathit{Circuit}[N \cdot \mathsf{poly}(t(n)/\varepsilon)]$, *then for infinitely many inputs* n, $\mathit{Circuit}[s(n)]$ *cannot be learnt up to error* $\varepsilon/2$ *with confidence* $1-\delta$ *by* $t(n)$*-size circuits using non-adaptive membership queries over the uniform distribution.*

We also show a related result which gives lower bounds for learnability of a circuit class $\mathcal{C}$ using $\mathcal{C}$-circuits by starting with a lower bound against worst-case MCSP instead of the average-case.

▶ **Lemma 35** (Hardness Magnification for Learnability from Lower Bounds for Gap MCSP)**.** *Let* $c \geq 1$ *be an arbitrary constant. If there is* $\varepsilon < 1/2$, *such that infinitely many input lengths* $N = 2^n$, $\mathit{MCSP}[n^c, 2^n/n^c] \notin \mathit{Circuit}[N^{1+\varepsilon}]$, *then for every* $\gamma \in (0,1)$, *for infinitely many inputs* n, $\mathit{Circuit}[n^c]$ *cannot be learnt up to error* $1/O(n^{2c})$ *with confidence* $1-1/n$ *by* $\mathit{Circuit}[2^{O(n^\gamma)}]$*-circuits using non-adaptive membership queries over the uniform distribution.*

Proof of Theorem 1. The following implications establish the desired equivalences.

$(a) \Longrightarrow (c)$: For the parameters c, γ, ε given by (a), we apply Lemma 34 for $s(n) = n^c$ and $t(n) = 2^{n^\gamma}$, to see that for some $\gamma' > 0$, Circuit$[n^c]$ cannot be learned by circuits of size $2^{O(n^{\gamma'})}$ via non-adaptive queries up to an error $O(1/n^c)$.

$(c) \Longrightarrow (d)$: We show the contrapositive of this implication. Suppose that for every $d \geq 1$, there exists a Circuit$[\mathsf{poly}(n)]$-natural property that is useful against Circuit$[n^d]$ for all large enough n. By Theorem 5, for every $c \geq 1$, we can learn Circuit$[n^c]$ by a sequence of oracle Circuit$[2^{O(n^{1/2})}]$-circuits up to an error of n^{-c}, by choosing $d = 2ac$ for the constant a from Theorem 5.

$(d) \Longrightarrow (a)$, $(d) \Longrightarrow (b)$: Trivial, using the fact that random functions are hard.

$(c) \Longrightarrow (e)$: Follows from the contrapositive of Theorem 10.

$(e) \implies (c)$: Follows from the non-uniform version of Proposition 29 in [41], using essentially the same proof.

$(b) \implies (c)$: For the parameter c given by (b), we apply Lemma 35 to see that $\mathsf{Circuit}[n^c]$ cannot be learned by circuits of size $2^{O(n^\gamma)}$ via non-adaptive queries up to an error $O(1/n^c)$, for any $\gamma \in (0,1)$. ◀

We now complete the proof of Theorem 1 by proving Lemmas 34 and 35.

Proof of Lemma 34. For the promise problem $\mathsf{MCSP}[(s,0),(t,\varepsilon)]$ over N inputs, define

$$\Pi_{yes} = \{y \in \{0,1\}^N \mid \exists \text{ circuit of size} \leq s(n) \text{ that computes } f_y\}$$
$$\Pi_{no} = \{y \in \{0,1\}^N \mid \text{no circuit of size} \leq t(n) \ (1-\varepsilon)\text{-approximates } f_y\}$$

We prove the contrapositive of the statement, by showing a reduction from $\mathsf{MCSP}[(s,0),(t,\varepsilon)]$ to a learning algorithm for $\mathsf{Circuit}[s(n)]$ using non-adaptive membership queries over the uniform distribution. For a fixed $\varepsilon < 1/2$ and $0 \leq \delta \leq 1/9$, let $\{D_n\}_{n\geq 1} \in \mathsf{Circuit}[t(n)]$ be the corresponding sequence of oracle circuits which learns $\mathsf{Circuit}[s(n)]$ up to error $\varepsilon/2$, where D_n makes non-adaptive queries to some function $f \in \mathsf{Circuit}[s(n)]$ over n inputs.

Let $q = q(n) = \frac{200}{\varepsilon^2}$. Define $F_N : \{0,1\}^N \times \{0,1\}^{nq(n)} \times \{0,1\}^{t(n)} \to \{0,1\}$ as the sequence of randomized circuits such that :

$$z \in \Pi_{yes} \implies \Pr_{y_1,w}\{F_N(z,y_1,w)=1\} > 2/3$$
$$z \in \Pi_{no} \implies \Pr_{y_1,w}\{F_N(z,y_1,w)=1\} < 1/3$$

The reduction F_N does the following. Let $Y = (x_1,\ldots,x_{t(n)})$ be the set of queries made by D_n. F_N runs the learner D_n with input w as its source of internal randomness and answers its oracle queries to f_z by using the other input $z \in \{0,1\}^N$. If the output string of the learner cannot be interpreted as a $t(n)$-sized circuit, then F_N outputs 0. Otherwise, let h be the $t(n)$-sized circuit on n inputs, which can interpret the hypothesis output by the learner as a $t(n)$-sized circuit. F_N then interprets the random input y_1 as a sequence of q random examples $v_1,\ldots,v_q \in \{0,1\}^n$ and computes h on each of these. It then forms a string $u \in \{0,1\}^q$, where for every $i \in [q], u_i = 1$ if and only if $h(v_i) = f_z(v_i)$. Finally, it uses a threshold gate on T on $q(n)$ inputs to check if the Hamming weight of u is at least $((1-3\varepsilon/4)q)$.

We now show the correctness of the reduction. If $z \in \Pi_{yes}$, then f_z is computed by some circuit of size at most $s(n)$. Thus, for every random choice of y_1 and w, D_n can learn the function f_z and with probability at least $(1-\delta)$, output a hypothesis h which has an error of at most $\varepsilon/2$ with respect to f_z. Now, for the q samples given by y_1, by an application of Hoeffding's inequality (Lemma 13), the probability that the Hamming weight of $u \in \{0,1\}^q$ is lesser than $(1-0.6\varepsilon)\,q$ is at most $2\exp(-2q\varepsilon^2/100)$ which is at most $1/4$ for our choice of q. When $\delta \leq 1/9$, we see that $T(u) = 1$ with probability at least $(1-\delta)3/4 \geq 2/3$.

On the other hand, if $z \in \Pi_{no}$, then no circuit of size at most $t(n)$ can even $(1-\varepsilon)$-approximate f_z. Thus, for any random choice of y_1 and w, any hypothesis h which D_n outputs is a circuit of size at most $t(n)$ and thus is at least ε-far from f_z. By a similar application of Hoeffding's inequality, we see that the probability that the Hamming weight of $u \in \{0,1\}^q$ is greater than $(1-0.9\varepsilon)\,q$ is at most $2\exp(-2q\varepsilon^2/100) \leq 1/4$. Therefore, $T(u) = 0$ with probability $2/3$.

For the next step, we need to derandomize the circuits F_N. Define E_N as

$$E_N : \{0,1\}^N \times \left(\{0,1\}^{n\cdot q+t(n)}\right)^R \to \{0,1\}$$
$$E_N(z, y^{(1)}, \ldots, y^{(R)}) = \mathsf{MAJ}_R(F_N(z, y^{(1)}), \ldots, F_N(z, y^{(R)}))$$

where $R = CN$ and each $y^{(j)} \in \{0,1\}^{n\cdot q+t(n)}$, for each $j \in [R]$.

When, $z \in \Pi_{yes}$, then using Hoeffding's inequality, we see that with probability at most 2^{-2N} (for suitably chosen C), the string $(F_N(z, y^{(1)}), \ldots, F_N(z, y^{(R)}))$ has Hamming weight $\leq 3R/5$. Similarly, when $z \in \Pi_{no}$, with probability at most 2^{-2N}, the string $(F_N(z, y^{(1)}), \ldots, F_N(z, y^{(R)}))$ has Hamming weight $\geq 2R/5$. Thus, the majority gate differentiates between the two cases except with probability at most 2^{-2N}. We use Adleman's trick [6] to fix a string $\alpha \in \{0,1\}^{R\cdot(n\cdot q+t(n))}$ which correctly derandomizes F_N on all inputs in Π_{yes} and Π_{no} and call the resulting circuit as E_N^* which computes the function $E_N^* : \{0,1\}^N \to \{0,1\}$.

We next compute the size of E_N^*. Each $F_N(z, y^{(i)})$ is fixed to $F_N(z, \alpha^{(i)})$, where $\alpha^{(i)} \in \{0,1\}^{(n\cdot q+t(n))}$ is the i^{th} section of the hardwired random string α. Observe that for the set of oracle queries Y made by D_n, it is enough to use appropriate literals from the input z whenever we need to access the truth table of f_z. Indeed, whenever D_n uses a random example, the randomness comes from $\alpha^{(i)}$ which is fixed non-uniformly and whenever it makes a membership query, the set of queries Y is fixed for D_n because of its non-adaptivity. Recall that the size of the circuit D_n is $t(n)$ and the hypothesis h output by the learner can be interpreted as a circuit and efficiently computed by another circuit of size $\mathsf{poly}(t(n))$. Thus, the circuit size to compute $F_N(z, \alpha)$ is at most $\mathsf{poly}(t(n) \cdot q)$ and the total circuit size to construct E_N^* is $O(N \cdot \mathsf{poly}(t(n)/\varepsilon))$. ◀

Proof sketch of Lemma 35. We show a two-sided error randomized reduction from $\mathsf{MCSP}[n^c, 2^n/n^c]$ to $\{D_n\}_{n\geq 1}$. Let $q = q(n) = O\left(n^{3c}\right)$. The reduction is almost the same as that of Lemma 34. Here we use a threshold gate on $q(n)$ inputs which answers 1 whenever the Hamming weight of its input is greater than $(1 - 1/n^{1.5c})q(n)$.

When the input to $\mathsf{MCSP}[n^c, 2^n/n^c]$ is a yes instance, with probability at least $(1 - 1/n)$, D_n outputs a hypothesis $h_n \in \mathsf{Circuit}[2^{n^\gamma}]$ which has error at most $1/O(n^{2c})$. Now for the $q(n)$ samples drawn uniformly at random, the probability that h agrees with the input instance on at least a $(1 - 1/n^{1.5c})q(n)$ samples is at least $(1 - 1/n)2/3$.

When the input to $\mathsf{MCSP}[n^c, 2^n/n^c]$ is a no instance, any hypothesis h which D_n outputs must have error greater than $1/O(n^{c+2})$. Indeed, if the error is less than $O(1/n^{c+2})$, then by hardwiring all the error inputs by using circuits of size at most $O\left(\frac{2^n}{n^{c+2}} \cdot n\right)$ we get a circuit of size at most $2^n/n^c$, which is a contradiction to the promise of the no instance. By Hoeffding's inequality, the probability that h agrees with the input instance on at most a $(1 - 1/n^{1.5c})q(n)$ samples is at least $2/3$.

The derandomization is the same as that of Lemma 34, obtained by repeating the above reduction $R = O(N)$ times and computing the majority over the R outputs of the reduction. The circuit size to compute $\mathsf{MCSP}[n^c, 2^n/n^c]$ is thus $O(N \cdot 2^{O(n^\gamma)} n^{3c}) = O(N^{1+\varepsilon})$, for $\varepsilon = o(1)$. ◀

4.2 Towards a More Robust Theory

The question of non-naturalizability of hardness magnification for $\mathsf{MCSP}[n^c/2n, n^c]$ is connected to the question of basing hardness of learning on the assumption $\mathsf{NP} \not\subseteq \mathsf{Circuit}[2^{O(n^\gamma)}]$.

▶ **Proposition 36.** *Assume that for every $\gamma \in (0,1)$ there is $d \geq 2$ such that $\mathsf{NP} \not\subseteq \mathsf{Circuit}[2^{O(n^\gamma)}]$ implies hardness of learning $\mathsf{Circuit}[n^d]$ by 2^{n^γ}-size circuits with error $1/n^d$. Then, there is a constant e such that for every $\gamma \in (0,1)$ and $c \geq 1$, $\mathsf{MCSP}[n^c/2n, n^c] \notin \mathsf{Circuit}[N^{1+e\gamma c}]$ implies that there is no $\mathsf{P/poly}$-natural property against $\mathsf{P/poly}$.*

Proof. By Theorem 5, $\mathsf{P/poly}$-natural property against $\mathsf{P/poly}$ implies that for every d there is $\gamma < 1/d$ and 2^{n^γ}-size circuits learning $\mathsf{Circuit}[n^d]$ with error $1/n^d$. By our assumption, this implies $\mathsf{NP} \subseteq \mathsf{Circuit}[2^{O(n^\gamma)}]$. We can now use $\mathsf{NP} \subseteq \mathsf{Circuit}[2^{O(n^\gamma)}]$ as the assumption in the proof of Theorem 29 to conclude that there is a constant e independent of γ such that for $c \geq 1$, $\mathsf{MCSP}[n^c/2n, n^c] \in \mathsf{Circuit}[N^{1+e\gamma c}]$. ◀

A form of the opposite implication holds as well if we assume NP-completeness of MCSP. Moreover, instead of the non-naturalizability of hardness magnification, we need to assume a reduction from worst-case MCSP to approximate MCSP.

▶ **Definition 37.** *A p-time algorithm A k-reduces $\mathsf{MCSP}[s,t]$ to $\mathsf{MCSP}[(s,0),(t,\varepsilon)]$ if it maps instances of $\mathsf{MCSP}[s,t]$ to instances of $\mathsf{MCSP}[(s,0),(t,\varepsilon)]$ and*

1. *For $f \in \mathsf{Circuit}[s]$, $A(\mathsf{tt}(f))$ is the truth-table of a Boolean function in $\mathsf{Circuit}[s^k]$.*
2. *For $f \notin \mathsf{Circuit}[t]$, $A(\mathsf{tt}(f))$ is not $(1-\varepsilon)$-approximable by circuits of size $t^{1/k}$.*

▶ **Proposition 38.** *Assume there is a p-time algorithm k-reducing $\mathsf{MCSP}[s,t]$ to $\mathsf{MCSP}[(s,0),(t,\varepsilon)]$ and that for all $0 < \alpha < \beta < 1$, $\mathsf{MCSP}[2^{\alpha n}, 2^{\beta n}]$ is NP-complete. If for every sufficiently small $\alpha > 0$ there is $\beta < 1/k$ and a $2^{\beta n}$-time algorithm learning $\mathsf{Circuit}[2^{\alpha n}]$ with error ε, then $\mathsf{P} = \mathsf{NP}$.*

Proof. Let A bet the p-time k-reduction from the statement and $\alpha > 0$ be sufficiently small. Assume we can learn in $2^{\beta n}$-time $\mathsf{Circuit}[2^{k\alpha n}]$ with error ε and $k\alpha < \beta < 1/k$. This implies that $\mathsf{MCSP}[(2^{k\alpha n}, 0), (2^{\beta n}, \varepsilon)]$ can be solved in p-time. Since A reduces an NP-complete problem $\mathsf{MCSP}[2^{\alpha n}, 2^{k\beta n}]$ to $\mathsf{MCSP}[(2^{k\alpha n}, 0), (2^{\beta n}, \varepsilon)]$, this shows that $\mathsf{P} = \mathsf{NP}$. ◀

5 The Locality Barrier

5.1 Lower Bounds Above Magnification Threshold

5.1.1 The Razborov-Smolensky Polynomial Approximation Method

In this section, we observe that the lower bound techniques of Razborov and Smolensky [44, 49] can be "localized." The following proposition instantiates the locality barrier for HM Frontier A.

▶ **Proposition 39** (Locality Barrier for HM Frontier A)**.** *The following results hold.*

- (A1$^{\mathcal{O}}$) (Oracle Circuits from Magnification) : $\mathsf{MKtP}[n^c, 2n^c] \in \mathsf{AND}\text{-}\mathcal{O}\text{-}\mathsf{XOR}[N^{1.01}]$. *More precisely, $\mathsf{MKtP}[n^c, 2n^c]$ is computed by circuits with $N^{1.01}$ gates and of the following form: the output gate is an AND gate of fan-in $O(N)$, at the middle layer are oracle gates of fan-in $\mathrm{poly}(n)$, and at the bottom layer are XOR gates.*
- (A3$^{\mathcal{O}}$) (Extension of Lower Bound Techniques) : *For a constant d, assume that $O_1, \cdots, O_d \in \mathbb{N}$ satisfy $\prod_{i=1}^{d} O_i \leq \sqrt{N}/\omega(\log N)^d$. Then* Majority *cannot be computed by a depth-d polynomial-size oracle $(\mathsf{AC}^0[\oplus])^{\mathcal{O}}$ circuit whose oracle gates on the i-th level have fan-in at most O_i.*

The first item is immediate from the proof of Theorem 14 in Section 3.1. In what follows, we prove the second item of Proposition 39.

Recall that the proof techniques of Razborov and Smolensky [44, 49] consist of two parts: The first lemma shows that any low degree polynomial cannot approximate Majority. (A simple proof sketch can be found in, e.g., [33].)

▶ **Lemma 40.** *For any polynomial $p \in \mathbb{F}_2[x_1, \cdots, x_N]$ of degree $\leq \sqrt{N}/4$,*

$$\Pr_{x \sim \{0,1\}^N}[p(x) \neq \mathsf{Majority}(x)] \geq \frac{1}{4}.$$

The second lemma shows that $\mathsf{AC}^0[\oplus]$ circuits can be approximated by low degree polynomials. We show that this argument can be localized.

▶ **Lemma 41.** *Let C be a depth-d polynomial-size oracle $\mathsf{AC}^0[\oplus]$ circuit whose oracle gates on the i-th level have fan-in at most O_i. Then there exists a polynomial $p \in \mathbb{F}_2[x_1, \cdots, x_N]$ of degree $\leq O(\log N)^d \cdot \prod_{i=1}^d O_i$ such that $\Pr_{x \sim \{0,1\}^N}[p(x) \neq C(x)] < \frac{1}{4}$.*

Proof Sketch. We convert each layer of the circuit C into a low degree probabilistic polynomial p that approximates C.

Consider the i-th level of a circuit C. NOT, OR, AND, and XOR gates can be converted into a probabilistic polynomial of degree $O(\log N)$ and error $1/\mathsf{poly}(N)$ in the standard way [44]. In order to represent an oracle gate $\mathcal{O}$ as a low-degree polynomial, we simply take the multilinear extension of the oracle gate $\mathcal{O}$. Note that, at the i-th level, the fan-in of the oracle gate $\mathcal{O}$ is bounded by O_i; thus the oracle gate at the i-th level can be represented as a polynomial of degree $\leq O_i$. Thus, in either cases, any gate at i-th level can be represented as a probabilistic polynomial of degree $\max\{O(\log N), O_i\}$. Continuing this for $i = 1, \cdots, d$ and composing resulting polynomials, we obtain a probabilistic polynomial of degree $\prod_{i=1}^d \max\{O(\log N), O_i\}$ that approximates C. This implies via standard techniques the existence of a (deterministic) polynomial of the same degree that correctly computes the circuit on most inputs. ◀

These two lemmas immediately imply the Majority lower bound for $(\mathsf{AC}^0[\oplus])^{\mathcal{O}}$:

Proof of *(A3$^{\mathcal{O}}$)* of Proposition 39. Suppose that there exists a depth-d polynomial-size oracle $\mathsf{AC}^0[\oplus]$ circuit that computes Majority and satisfies the condition of Proposition 39. By Lemma 41, there exists a polynomial p of degree at most $O(\log N)^d \cdot \prod_{i=1}^d O_i \leq o(\sqrt{N})$ that approximates Majority. However, this contradicts Lemma 40. ◀

Finally, we mention that an incomparable bound can be obtained by using a lower bound for $\mathsf{AC}^0[\oplus]$ interactive compression games.

▶ **Proposition 42** ([40, Corollary 5.3])**.** (A3$^{\mathcal{O}}$) Majority $\notin (\mathsf{AC}^0[\oplus])^{\mathcal{O}}[\mathsf{poly}(n)]$ *if the total number of input wires in the circuit feeding the $\mathcal{O}$-gates is $N/(\log N)^{\omega(1)}$.*

5.1.2 The Formula-XOR Lower Bound of [52]

This section captures an instantiation of the locality barrier for HM Frontier B. Throughout this section we use the $\{-1, 1\}$ realization of the Boolean domain (that is, -1 represents True and 1 represents False). Let Formula-XOR on variables $x_1, \ldots, x_n$ be the class of formulas where the input leaves are labeled by parity functions of arbitrary arity over $x_1, \ldots, x_n$.

▶ **Proposition 43** (Locality Barrier for HM Frontier B). *The following results hold.*

- (B1$^{\mathcal{O}}$) (Oracle Circuits from Magnification) : *For any $\varepsilon > 0$,* $\mathsf{MCSP}[2^{n^{1/3}}, 2^{n^{2/3}}] \in$ Formula-$\mathcal{O}$-XOR$[N^{1.01}]$*, where every oracle $\mathcal{O}$ has fan-in at most N^{ε} and appears in the layer right above the* XOR *leaves.*
- (B3$^{\mathcal{O}}$) (Extension of Lower Bound Techniques) : *For any $\delta > 0$,* InnerProduct *over N input bits cannot be computed by $N^{2-3\delta}$-size* Formula-$\mathcal{O}$-XOR *circuits with at most $N^{2-3\delta}$ oracle gates of fan-in N^{δ} in the layer right above the* XOR *leaves, for any oracle $\mathcal{O}$.*

To prove item 2 of Proposition 43, we adapt Tal's [52] lower bound for bipartite formulas[15], for which we need the following results.

▶ **Lemma 44** ([47, 52]). *Let F be a De Morgan formula of size s which computes $f : \{-1,1\}^n \to \{1,1\}$. Then, there exists a multilinear polynomial p over $\mathbb{R}$ of degree $O(\sqrt{s})$, such that for every $x \in \{-1,1\}^n$, $p(x) \in [F(x) - 1/3, F(x) + 1/3]$.*

For any function $f : \{-1,1\}^n \to \{-1,1\}$, f is ε-correlated with a parity $p_S(x) = \prod_{i \in S} x_i$, if $|\mathbb{E}_{x \in \{-1,1\}^n}[f(x) \cdot p_S(x)]| \geq \varepsilon$. We have

▶ **Lemma 45.** *For any $\delta > 0$, let $F(x_1, \ldots, x_n)$ be a* Formula-$\mathcal{O}$-XOR *formula of size s, where every oracle $\mathcal{O}$ has fan-in at most n^{δ} and appears in the layer right above the* XOR *leaves. Then the following hold true:*

1. *There exists a multi-linear polynomial $p(x_1, \ldots, x_n)$ over $\mathbb{R}$ with at most $s^{O(\sqrt{s})} \cdot 2^{n^{\delta} \cdot O(\sqrt{s})}$ monomials such that for every $x \in \{-1,1\}^n$, sign$(p(x)) = F(x)$.*
2. *There exists a parity function $f_T(x_1, \ldots, x_n)$ which is at least $\left(\frac{1}{s^{O(\sqrt{s})} \cdot 2^{n^{\delta} O(\sqrt{s})}}\right)$-correlated with F.*

Proof. We assume that the oracle function is a Boolean function on n^{δ} inputs. Let $t \leq s/n^{\delta}$ be the number of oracle gates in F. Let $p_1, \ldots, p_s$ be the leaves of F, where each p_i is an XOR gate over $x_1, \ldots, x_n$ and every oracle gate $g_1, \ldots, g_t$ is such that $g_i(x) = \mathcal{O}(p_{i1}(x), \ldots, p_{i\ell}(x))$, where $\ell = n^{\delta}$ and $p_{ij} \in \{p_1, \ldots, p_t\}$ for every $i \in [t], j \in [\ell]$.

Let F' be a De morgan formula obtained by replacing oracle gates in F with new variables z_i (for notational simplicity we assume that every leaf is an input to some oracle gate), for $i \in [t]$. We now use Lemma 44 on F' to get a degree $d = O(\sqrt{t})$ polynomial $q(z)$ such that for every $z \in \{-1,1\}^t$, sign$(q(z)) = F'(z)$. Expanding $q(z)$ as a multilinear polynomial :

$$q(z) = \sum_{S \subseteq [t], |S| \leq d} \hat{q}(S) \prod_{i \in S} z_i$$

To prove the first item, we replace each z_i by the original leaf and we get that for every $x \in \{-1,1\}^n$,

[15] A bipartite formula on variables $x_1, \ldots, x_n, y_1, \ldots, y_n$ is a formula such that each leaf computes an arbitrary function in either $(x_1, \ldots, x_n)$ or $(y_1, \ldots, y_n)$. Formula-XOR circuits are a subset of bipartite formulas as one can always write $\oplus(x_1, \ldots, x_{2n})$ as the parity of $\oplus(x_1, \ldots, x_n)$ and $\oplus(x_{n+1}, \ldots, x_{2n})$.

$$
\begin{aligned}
F(x) &= \operatorname{sign}\left(\sum_{S\subseteq[t],|S|\leq d} \hat{q}(S)\prod_{i\in S} g_i(x)\right)\\
&= \operatorname{sign}\left(\sum_{S\subseteq[t],|S|\leq d} \hat{q}(S)\prod_{i\in S}\left(\sum_{U\subseteq[\ell]} \hat{\mathcal{O}}(U)\prod_{j\in U} p_{ij}(x)\right)\right)\\
&= \operatorname{sign}\left(\sum_{\substack{S\subseteq[t]\\ S=\{i_1,\ldots,i_{|S|}\}\\ |S|\leq d}}\ \sum_{U_{i_1},\ldots,U_{i_{|S|}}\subseteq[\ell]} \hat{q}(S)\cdot\left(\prod_{1\leq k\leq|S|}\hat{\mathcal{O}}(U_{i_k})\prod_{j\in U_{i_k}} p_{i_kj}(x)\right)\right)
\end{aligned}
$$

where the second equality uses the fact that any Boolean function on ℓ inputs can be represented by a multilinear polynomial of degree at most ℓ where each coefficient is between $[-1,1]$. Clearly the number of monomials is at most $s^{O(\sqrt{s})}\cdot 2^{n^\delta\cdot O(\sqrt{s})}$.

To prove the second item, firstly observe that for every $z\in\{-1,1\}^t$, $q(z)\cdot F'(z)\in[2/3,4/3]$, because $|q(z)-F'(z)|\leq 1/3$ for every z. This also means that for the polynomial $r(x)=q(g_1(x),\ldots,g_t(x))$, $\mathbb{E}_{x\in\{-1,1\}^n}[r(x)\cdot F(x)]\geq 2/3$.

Given that $\hat{q}(S)=\mathbb{E}_{z\in\{-1,1\}^s}\left[q(z)\prod_{i\in S} z_i\right]$, we see that $|\hat{q}(S)|\leq 4/3$. We have

$$
\begin{aligned}
2/3 &\leq \mathop{\mathbb{E}}_{x\in\{-1,1\}^n}[F(x)\cdot r(x)]\\
&= \mathop{\mathbb{E}}_{x\in\{-1,1\}^n}\left[F(x)\cdot\sum_{S\subseteq[t],|S|\leq d}\hat{q}(S)\prod_{i\in S} g_i(x)\right]\\
&\leq \sum_{\substack{S\subseteq[t]\\ S=\{i_1,\ldots,i_{|S|}\}\\ |S|\leq d}}\ \sum_{U_{i_1},\ldots,U_{i_{|S|}}\subseteq[\ell]}\hat{q}(S)\prod_{1\leq k\leq|S|}\hat{\mathcal{O}}(U_{i_k})\mathop{\mathbb{E}}_{x\in\{-1,1\}^n}\left[F(x)\prod_{1\leq k\leq|S|}\prod_{j\in U_{i_k}} p_{i_kj}(x)\right]
\end{aligned}
$$

Since, $|\hat{q}(S)|\leq 4/3$ for every $S\subseteq[t]$ and $|\hat{\mathcal{O}}(U)|\leq 1$ for every $U\subseteq[\ell]$, we see that there exists a set S of size at most d and sets $U_{i_1},\ldots,U_{i_{|S|}}$ such that $\left|\mathbb{E}_{x\in\{-1,1\}^n}\left[F(x)\cdot\prod_{1\leq k\leq|S|}\prod_{j\in U_{i_k}} p_{i_kj}(x)\right]\right|\geq\frac{1}{t^{O(\sqrt{t})}\cdot 2^{n^\delta O(\sqrt{t})}}\geq\frac{1}{s^{O(\sqrt{s})}\cdot 2^{n^\delta O(\sqrt{s})}}$. Taking p_T be the parity of the parities given by $p_T=\prod_{1\leq k\leq|S|}\prod_{j\in U_{i_k}} p_{i_kj}(x)$, we see that p_T is $\frac{1}{s^{O(\sqrt{s})}\cdot 2^{n^\delta O(\sqrt{s})}}$-correlated with F. ◀

Define the Inner Product modulo 2 function, $\mathsf{InnerProduct}_n:\{-1,1\}^n\times\{-1,1\}^n\to\{-1,1\}$ as $IP_n(x,y)=(-1)^{\sum_{i=1}^n(1-x_i)(1-y_i)/4}$.

Proof Sketch of Proposition 43. The first item follows from an inspection of the proof of Theorem 25 in Section 3.2. Theorem 24 gives the same oracle circuit construction (with different oracles) under the assumption $\mathsf{QP}\subseteq\mathsf{P/poly}$.

The second item follows from Lemma 45. We observe that three different techniques used to show Formula-XOR lower bounds localize. Firstly, Tal's lower bound based on sign rank shows that the sign rank of any Formula-XOR circuit F is at most the number of monomials in the polynomial p given by the first item of lemma 45. Since this is at most $s^{O(\sqrt{s})}\cdot 2^{n^\delta\cdot O(\sqrt{s})}$ and InnerProduct has a sign rank which is at least $2^{n/2}$ [18], the lower

bound follows. Secondly, Tal's lower bound based on the discrepancy of a function also localizes, as he shows that the discrepancy of F is at least a constant times the correlation of F with the parity f_T given by item 2 of Lemma 45, which is at least $\Omega\Big(\frac{1}{s^{O(\sqrt{s})}\cdot 2^{n^{\delta}O(\sqrt{s})}}\Big)$, whereas the discrepancy of the inner product is at most $1/2^{n/2}$ (cf. [29, Lemma 14.5]), thus proving the given lower bound for inner product. Finally, we also observe that the lower bound technique of showing high correlation of F with some parity f_T and the fact that inner product has exactly $2^{-n/2}$-correlation with any parity also localizes to give the same lower bound. ◀

5.1.3 Almost-Formula Lower Bounds

This section captures an instantiation of the locality barrier for HM Frontier C. We recall the following definition. Consider an s-almost formula. Each gate G of F with fanout larger than 1 is computed by a formula with inputs being either the original inputs of F or gates of F with fanout larger than 1. We call any maximal formula of this form a *principal* formula of G.

▶ **Theorem 46** (Locality Barrier for HM Frontier C). *The following results hold.*

- (C1$^{\mathcal{O}}$) (Oracle Circuits from Magnification) : $\mathit{MCSP}[2^{n^{1/2}}/2n, 2^{n^{1/2}}]$ *is computable by* $2^{O(n^{1/2})}$*-almost formulas of size* $2^{n+O(n^{1/2})}$ *with oracles of fanin* $2^{O(n^{1/2})}$ *at the bottom layer of principal formulas computing gates with fanout larger than 1.*
- (C3$^{\mathcal{O}}$) (Extension of Lower Bound Techniques) : *For every* $\varepsilon < 1$, PARITY *is not in* n^{ε}-almost-$\mathit{Formula}[n^{2-9\varepsilon}]$ *even if the almost-formulas are allowed to use arbitrary oracles of fanin* $< n^{\varepsilon}$ *at the bottom layer of principal formulas computing gates with fanout larger than* 1.

Proof. The first item follows by inspecting the proof of Theorem 29. It is not hard to see that $\mathsf{MCSP}[2^{n^{1/2}}/2n, 2^{n^{1/2}}]$ is computable by $2^{O(n^{1/2})}$-almost formulas F_N of size $2^{n+O(n^{1/2})}$ with local oracles of fanin $2^{O(n^{1/2})}$. Moreover, the only gates of fanout larger than 1 are the gates computing anticheckers $y_1, \ldots, y_{2^{O(n^{1/2})}}$ with bits $f(y_1), \ldots, f(y_{2^{O(n^{1/2})}})$. We want to show that the local oracles are at the bottom of principal formulas generating gates with fanout larger than 1. In order to achieve this we need to modify formulas F_N a bit.

First, note that F_N contains an oracle which is applied on top of anticheckers $y_1, \ldots, y_{2^{O(n^{1/2})}}$ with bits $f(y_1), \ldots, f(y_{2^{O(n^{1/2})}})$. In order to ensure that this oracle is at the bottom of a principal formula computing a gate with fanout bigger than 1 we simply add dummy negation gates to the output gate and the gates computing anticheckers $y_1, \ldots, y_{2^{O(n^{1/2})}}$ with bits $f(y_1), \ldots, f(y_{2^{O(n^{1/2})}})$, if necessary.

Second, note that each y_{i+1}, $f(y_{i+1})$ is generated as follows: 1. if $R_f(y_1, \ldots, y_i) \geq 2n^2$ then a subformula F' generates anticheckers y_{i+1}, $f(y_{i+1})$, and 2. if $R_f(y_1, \ldots, y_i) < 2n^2$ then a subformula F'' generates anticheckers y_{i+1}, $f(y_{i+1})$. In both cases we replace predicates $R_f(y_1, \ldots, y_i) < 2n^2$ by oracles. In case 1, subformulas of F' with oracles at the bottom compute predicates $F^{r,h}$ from the proof of Lemma 27. This process generates a set of $2^{O(n^{1/2})}$ potential anticheckers. F' chooses the right antichecker by applying another oracle. In order to ensure that this top oracle is at the bottom of a principal formula, we add dummy negation gates to the gates generating the potential anticheckers. This increases the number of gates with fanout larger than 1 only by $2^{O(n^{1/2})}$. In case 2, y_{i+1}, $f(y_{i+1})$ is generated by oracles outputting circuits which have not been killed yet and evaluating them on all possible inputs. Here we ensure that the oracles are at the bottom by asking them to perform both tasks: choose the next alive circuit and evaluate it on a given input. The oracle selecting the right antichecker from the set of potential anticheckers is treated in the same way as in case 1. All in all, we obtain the desired oracle almost formulas.

The second item is proved analogously to Theorem 30. For the sake of contradiction assume PARITY has n^ε-almost formulas of size $n^{2-9\varepsilon}$ with local oracles at the bottom of principal formulas. Since there are only n^ε gates of fanout > 1, we can replace these gates by constants and obtain formulas F_n of size $n^{2-8\varepsilon}$ with local oracles at the bottom computing PARITY with probability $\geq 1/2 + 1/2^{n^\varepsilon}$. Let $L'(f)$ be the size (i.e. the number of leafs) of the smallest formula with local oracles at the bottom computing f. Since oracles have fanin $< n^\varepsilon$ and are located at the bottom, each function $f : \{-1,1\}^n \mapsto \{-1,1\}$ can be approximated by a polynomial of degree $O(t\sqrt{L'(f)}\frac{\log n}{\log\log n}n^\varepsilon)$ up to point-wise error of 2^{-t}. This implies that each formula of size $o((n/t)^2(\log\log n/\log n)^2(1/n^\varepsilon)^2)$ with local oracles at the bottom computes PARITY with probability at most $1/2 + 1/2^{t+O(1)}$ (for large enough t). Taking $t = n^{2\varepsilon}$ we get a contradiction. ◀

5.1.4 GapAND-Formula Lower Bounds

This section captures an instantiation of the locality barrier for HM Frontier D.

▶ **Theorem 47** (Locality Barrier for HM Frontier D). *The following results hold.*

1. (D1$^{\mathcal{O}}$) (Oracle Circuits from Magnification) : $\mathit{MCSP}[2^{\sqrt{n}}] \in \mathsf{GapAND}_{O(N)}\text{-}O_{N^{o(1)}}\text{-}\mathsf{Formula}[N^2]$.
2. (D3$^{\mathcal{O}}$) (Extension of Lower Bound Techniques) : *For* $0 < \beta < \varepsilon < 1$, $\mathsf{Andreev}_N \notin \mathsf{GapAND}_{O(N)}\text{-}O_{N^\beta}\text{-}\mathsf{Formula}[N^{3-\varepsilon}]$.
3. (D3$^{\mathcal{O}}$) : *Furthermore,* $\mathit{MCSP}[2^n/n^4] \notin \mathsf{GapAND}_{O(N)}\text{-}O_{N^\beta}\text{-}\mathsf{Formula}[N^{3-\varepsilon}]$, *for* $0 < \beta < \varepsilon < 1$.

Item 1 of the theorem above follows directly from Theorem 25.

Next we show that the classical $N^{3-o(1)}$ formula size lower bound for the Andreev's function [21, 51] localizes, even in the presence of a GapAND gate of bounded fan-in at the top of the formula.

Proof of Item 2. Let $m = N/2$, recall that $\mathsf{Andreev}_N$ is defined on a $2m$-bit string $z = x \circ y$, where $x, y \in \{0,1\}^m$. For simplicity, we assume m is a power of 2 in the following.

$\mathsf{Andreev}_N(x,y)$ first partitions x into $\log m$ blocks $x_1, x_2, \ldots, x_{\log m}$, each of length $m/\log m$. After that, it computes $i \in \{0,1\}^{\log m}$ as $i = \mathsf{PARITY}(x_1) \circ \mathsf{PARITY}(x_2) \circ \ldots \mathsf{PARITY}(x_{\log m})$. It then treats i as an integer from $[m]$, and outputs y_i.

Now, suppose there is a $\mathsf{GapAND}_{O(N)}\text{-}O_{N^\beta}\text{-}\mathsf{Formula}[N^{3-\varepsilon}]$ formula for $\mathsf{Andreev}_N$. Suppose we fix the y variables to a string $w \in \{0,1\}^m$, and apply a random restriction keeping exactly one variable from each block alive to x variables, then w.p. 0.9, we obtain a $\mathsf{GapAND}_{O(N)}\text{-}O_{N^\beta}\text{-}\mathsf{Formula}[N^{1-\varepsilon}\cdot\mathrm{polylog}(N)]$ formula computing $f_w : \{0,1\}^{\log m} \to \{0,1\}$ [51].

That is, for all $w \in \{0,1\}^m$, there exists an $O_{N^\beta}\text{-}\mathsf{Formula}[N^{1-\varepsilon}\cdot\mathrm{polylog}(N)]$ formula 0.8-approximating f_w. Note that there are at most $2^{N^{1-\varepsilon+\beta}\cdot\mathrm{polylog}(N)}$ such $O_{N^\beta}\text{-}\mathsf{Formula}[N^{1-\varepsilon}\cdot\mathrm{polylog}(N)]$ formulas, and there are 2^N possible w's (Note that O is a fixed oracle which does not depend on w). Since each $O_{N^\beta}\text{-}\mathsf{Formula}[N^{1-\varepsilon}\cdot\mathrm{polylog}(N)]$ formula can only 0.8-approximate $2^{\alpha\cdot N}$ many functions from $\{0,1\}^{\log m} \to \{0,1\}$ for a constant $\alpha < 1$, there must exist a w such that f_w cannot be 0.8-approximated by such formulas, contradiction. ◀

Next, we observe that the $N^{3-o(1)}$ formula lower bound for MCSP [16] also localizes.

Proof of Item 3. We first observe that the PRG construction of [16] also works for oracle formulas. (We omit the details of this proof.)

▷ Claim 48 ([16]). For $0 < \beta < \varepsilon < 1$, there is $M = N^{1-\Omega_{\beta,\varepsilon}(1)}$ and a PRG $G : \{0,1\}^M \to \{0,1\}^N$ such that the following hold.

1. For each fixed $z \in \{0,1\}^M$, $G(z)$, when interpreted as a function from $\{0,1\}^{\log N} \to \{0,1\}$, can be computed by a circuit of size $N^{1-\Omega(1)}$.
2. For all O_{N^β}-Formula$[N^{3-\varepsilon}]$ formulas C, we have

$$\left| \Pr_{z\in\{0,1\}^N}[C(z)=1] - \Pr_{z\in\{0,1\}^M}[C(G(z))=1] \right| \le 0.01.$$

Now, suppose MCSP$[2^n/n^4]$ on $N = 2^n$ bits can be computed by a GapAND$_{O(N)}$-O_{N^β}-Formula$[N^{3-\varepsilon}]$ formula C. Let $C_1, C_2, \ldots, C_{b\cdot N}$ be the O_{N^β}-Formula$[N^{3-\varepsilon}]$ sub-formulas of C under the top GapAND gate, where b is a constant.

We know that

$$\Pr_{z\in\{0,1\}^N}[\mathsf{MCSP}[2^n/n^4](z)=1] = o(1).$$

Since C computes MCSP$[2^n/n^4]$, and $C(x) = 0$ implies $C_i(x) = 0$ for at least a 0.9 fraction of $i \in [b \cdot N]$. We have that

$$\Pr_{i\in[b\cdot N],\, z\in\{0,1\}^N}[C_i(z)=1] \le 0.2.$$

On the other side, by the definition of MCSP$[2^n/n^4]$, and the Item (1) of Claim 48, it follows that

$$\Pr_{z\in\{0,1\}^M}[\mathsf{MCSP}[2^n/n^4](G(z))=1] = 1.$$

Again, since C computes MCSP$[2^n/n^4]$, and $C(x) = 1$ implies $C_i(x) = 1$ for all $i \in [b \cdot N]$. We have that

$$\Pr_{i\in[b\cdot N],\, z\in\{0,1\}^M}[C_i(G(z))=1] = 1.$$

Therefore, there must exist an i such that

$$\left| \Pr_{z\in\{0,1\}^N}[C_i(z)=1] - \Pr_{z\in\{0,1\}^M}[C_i(G(z))=1] \right| \ge 0.5,$$

which is a contradiction to Item (2) of Claim 48. ◀

Finally, we show that there is a language in E which cannot be computed by GapAND$_{O(N)}$-Formula$[N^{3-\varepsilon}]$ formulas, but it *can* be computed by an $O_{N^{o(1)}}$-Formula$[N^2]$ formula. Therefore, this lower bound does not localize in the sense of Theorem 47.

▶ **Theorem 49.** *There is a language $L \in$ E, such that $L \notin$ GapAND$_{O(N)}$-Formula$[N^{3-\varepsilon}]$ for all constants $\varepsilon > 0$, but $L \in O_{N^{o(1)}}$-Formula$[N^2]$.*

Proof. The function L is very similar to the Andreev's function. On an input x of length N, let $m = \log N$ (we assume N is a power of 2 for simplicity). To avoid the second input to Andreev$_N$, we want to find a function $f_{\mathsf{hard}} : \{0,1\}^m \to \{0,1\}$ which cannot be 0.8-computed by $N^{1-\varepsilon/2}$ formulas in $2^{O(N)}$ time (such a function exists by a simple counting argument). To find f_{hard}, we simply enumerate all possible functions $f : \{0,1\}^m \to \{0,1\}$, and check whether it can be 0.8-approximated by an $N^{1-\varepsilon/2}$-size formula.

There are $2^{2^m} = 2^N$ possible functions on m bits, and $(N^{1-\varepsilon/2})^{O(N^{1-\varepsilon/2})} = 2^{N^{1-\varepsilon/2}\cdot \mathrm{polylog}(N)}$ many formulas of $N^{1-\varepsilon/2}$ size. Hence, a straightforward implementation of the algorithm runs in $2^{O(N)}$ time.

Next, L partitions x into m blocks $x_1, x_2, \ldots, x_m$, each of length N/m. After that, it computes $i \in \{0,1\}^m$ as $i = \mathsf{PARITY}(x_1) \circ \mathsf{PARITY}(x_2) \circ \ldots \mathsf{PARITY}(x_m)$. It then outputs $f_{\mathsf{hard}}(i)$.

Now, suppose there is a $\mathsf{GapAND}_{O(N)}$-$\mathsf{Formula}[N^{3-\varepsilon}]$ for L. We apply a random restriction keeping exactly one variable from each block alive, then w.p. 0.9, we obtain a $\mathsf{GapAND}_{O(N)}$-$\mathsf{Formula}[N^{1-\varepsilon} \cdot \mathrm{polylog}(N)]$ formula for f_{hard} [51], which implies that there is an $N^{1-\varepsilon} \cdot \mathrm{polylog}(N)$-size formula 0.8-approximating f_{hard}, contradiction.

Finally, it is easy to verify that $L \in \mathsf{E}$ and $L \in O_{N^{o(1)}}$-$\mathsf{Formula}[N^2]$. ◀

5.1.5 AC^0 Lower Bounds via Random Restrictions

This section states and proves a result capturing an instantiation of the locality barrier for HM Frontier E.

▶ **Proposition 50** (Locality Barrier for HM Frontier E). *The following results hold.*

- ($\mathrm{E1}^{\mathcal{O}}$) (Oracle Circuits from Magnification) : *For each $k = (\log n)^C$ and every large enough depth d, $(n-k)$-$\mathsf{Clique} \in (\mathsf{AC}^0_d)^{\mathcal{O}}[m^{1+\varepsilon_d}]$, where $\varepsilon_d \to 0$ as $d \to \infty$, and the corresponding circuit employs a single oracle gate $\mathcal{O}$ of fan-in at most $O((\log n)^{4C})$.*
- ($\mathrm{E3}^{\mathcal{O}}$) (Extension of Lower Bound Techniques) : $\mathsf{Parity} \notin (\mathsf{AC}^0)^{\mathcal{O}}[\mathsf{poly}(n)]$ *if the total number of input wires in the circuit feeding the $\mathcal{O}$-gates is $n/(\log n)^{\omega(1)}$.*

Proof. The first item is established by inspection of the proof of Proposition 31, which relies on the circuit construction from [42] and a straightforward translation between vertex cover and clique detection. Recall that the circuit in [42] simulates a well-known kernelization algorithm for k-Vertex-Cover. This algorithm produces a graph H containing $O(k^2)$ vertices and a new parameter $k_H \leq k$. This graph can be described by a string of length $O(k^4)$, and the pair (H, k_H) becomes the input string to the single oracle $\mathcal{O}$ that is necessary in the oracle circuit construction. (If $\mathcal{O}$ solves vertex cover, the resulting oracle circuit correctly solves $(n-k)$-Clique.)

The second item easily follows by simulating oracle circuits via interactive compression games (see e.g. [40, Section 5]). In other words, one can view a circuit with oracles as an interactive protocol between two parties, where one of them has unbounded computational power, and the other is restricted to computations in a fixed circuit class. The total number of wires feeding the oracle gates corresponds to the number of bits sent to the unbounded party. The desired lower bound for oracle circuits then follows immediately from the main result from [9], which shows that the random restriction method can be extended to establish limitations on circuits with oracle gates of large fan-in. ◀

Informally, the main difficulty with the use of random restrictions in connection to HM Frontier E is that as soon as one simplifies a boolean circuit so that the oracle gate $\mathcal{O}$ is directly fed by input literals, one can fix just $(\log n)^{O(C)}$ input variables and eliminate this gate. Sacrificing such a small number of coordinates won't affect a typical worst-case lower bound based on the random restriction method.

5.1.6 Lower Bounds Through Reductions

Consider a reduction of PARITY to $\mathsf{MCSP}[2^{n^{1/2}}/2n, 2^{n^{1/2}}]$ by subquadratic-size n^{ε}-almost formulas with $n^{\varepsilon'}$ MCSP (possibly non-local) oracles at the bottom of each principal formula computing a gate with fanout > 1. By Theorem 46, such a reduction would imply $\mathsf{MCSP}[2^{n^{1/2}}/2n, 2^{n^{1/2}}] \notin n^{\varepsilon}$-almost-$\mathsf{Formula}[N^{1.1}]$ assuming that after replacing all oracles by

n^ε-almost formulas of size $N^{1.1}$ the total size of the resulting circuit remains $< N^{2-9(\varepsilon+\varepsilon')}$. In combination with hardness magnification, this would give us $\mathsf{NP} \not\subseteq \mathsf{NC}^1$. Unfortunately, Theorem 46 rules this possibility out.

▶ **Corollary 51.** *PARITY is not computable by subquadratic-size n^ε-almost formulas with $n^{\varepsilon'}$ oracle gates computing $\mathsf{MCSP}[2^{n^{1/2}}/2n, 2^{n^{1/2}}]$, assuming that after replacing all oracles by n^ε-almost formulas of size $N^{1.1}$ the total size of the resulting circuit remains $< N^{2-9(\varepsilon+\varepsilon')}$ for $\varepsilon + \varepsilon' < 1$.*

Proof. Assume the reduction in question exists. By Theorem 29, for every $\varepsilon > 0$ and all sufficiently big n, $\mathsf{MCSP}[2^{n^{1/2}}/2n, 2^{n^{1/2}}]$ is computable by $N^{1.1}$-size n^ε-almost formulas with local oracles at the bottom of principal formulas computing gates with fanout > 1. By the assumption, if we replace the MCSP oracles in the reduction by almost-formulas with local oracles, the resulting circuit is an $n^{\varepsilon+\varepsilon'}$-almost formula of size $N^{2-9(\varepsilon+\varepsilon')}$ with oracles of bounded fan-in. This contradicts the second item of Theorem 46. ◀

Analogous arguments rule out the possibility of establishing strong lower bounds via reductions also in other HM frontiers.

5.2 Lower Bounds Below Magnification Threshold

The localizations presented in this section show that one cannot obtain strong circuit lower bounds by "lowering the threshold" in certain hardness magnification proofs. As a consequence of one of our results (Theorem 59 in Section 5.2.2), we also refute the Anti-Checker Hypothesis from [39].

5.2.1 AC^0 Lower Bounds via Pseudorandom Restrictions

In this section we show that the AC^0 lower bounds proved for MCSP (MKtP) via pseudorandom restrictions [16] (see also Section 3.1) localize in a very strong sense.

We use $\mathsf{AC}^0_d[O_1, O_2, \ldots, O_d]$ to denote AC^0_d circuits extended with arbitrary oracles, such that oracle gates on the i-th level (the gates whose distance from the inputs is i) have fan-in at most O_i.

▶ **Theorem 52.** *There is a constant c such that for all $\varepsilon > 0$, constants d, and $O_1, O_2, \ldots, O_d$ such that $\prod_{i=1}^{d} O_i \leq N/(\log N)^{\omega(1)}$, $\mathsf{MCSP}[n^c, n^{2c}] \notin \mathsf{AC}^0_d[O_1, O_2, \ldots, O_d][\mathsf{poly}(N)]$.*

▶ Remark 53. We remark that the constraint on oracles in the above theorem is incomparable to the second item of Proposition 50. Here we focus on the maximum oracle fan-in at each level, while there the focus is on the total fan-in of all oracles. A lower bound result for an explicit problem with parameters similar to Theorem 52 is not known for AC^0 oracle circuits extended with parity gates *(see [40] for results in this direction)*.

We are going to apply Lemma 17, together with the following well-known results on k-wise independence fooling CNFs.

▶ **Lemma 54** ([7, 53])**.** *$k = O(\log(M/\varepsilon) \cdot \log(M))$-wise independent distribution ε-fools M-clauses CNFs.*

Combining Lemma 17 and Lemma 54, we have the following lemma.

▶ **Lemma 55.** *Let φ be a t-width M-clause CNF formula over N inputs, $p = 2^{-q}$ for some $q \in \mathbb{N}$, and $\varepsilon_0 > 0$ be a real. There is a p-regular,*

$$k = \Theta(\log(M \cdot 2^{t(q+1)}/\varepsilon_0) \cdot \log(M \cdot 2^{t(q+1)}) \cdot q^{-1})\text{-wise}$$

independent random restriction $\boldsymbol{\rho}$ such that

$$\Pr_{\rho \sim \boldsymbol{\rho}}[\mathsf{DT}(\varphi\restriction_\rho) > s] \leq 2^{s+t+1}(5pt)^s + \varepsilon_0 \cdot 2^{(s+1)(2t+\log M)}.$$

Moreover, $\boldsymbol{\rho}$ is samplable with $O(t \cdot q \cdot \mathrm{polylog}(M, N) \cdot \log(1/\varepsilon_0))$ bits, and each output coordinate of the random restriction can be computed in time polynomial in the number of random bits.

The moreover part follows from standard construction of k-wise independent distributions (see e.g. [55]).

We also need the following lemma which states that an arbitrary oracle with inputs being small-size decision trees shrinks to a small-size decision tree with high probability, under suitable pseudorandom restrictions.

▶ **Lemma 56.** *Let $O : \{0,1\}^T \to \{0,1\}$ be an arbitrary function, and $D_1, D_2, \ldots, D_T$ be T k-query decision trees on variables $x_1, x_2, \ldots, x_N$. Let $F := O \circ (D_1, D_2, \ldots, D_T)$ be their compositions. For $s \in \mathbb{N}$, and all $k(s+1)$-wise independent $1/(T \cdot k^2)$-regular random restriction $\boldsymbol{\rho}$, we have*

$$\Pr_{\rho \sim \boldsymbol{\rho}}[\mathsf{DT}(F\restriction_\rho) > s] \leq \left(\frac{k(s+1)}{2e^2}\right)^{-(s+1)}.$$

Proof. We focus on the following particular decision tree for evaluating $\{D_1, D_2, \ldots, D_T\}$ with respect to a restriction $\rho : [N] \to \{0, 1, *\}$:

Algorithm 1 $\mathsf{Eval}(\rho, D_1, D_2, \ldots, D_T)$.

- For i from 1 to T:
  - Simulate decision tree D_i with restriction ρ. That is, when D_i queries an index j, we feed ρ_j to D_i if $\rho_j \in \{0,1\}$, and query the j-th bit otherwise.
- Let α_i be the output of the i-th decision tree, we output $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_T)$.

To obtain a decision tree for $F\restriction_\rho$, we can run $\mathsf{Eval}(\rho, D_1, D_2, \ldots, D_T)$ to obtain α first and output $F(\alpha)$ at the end.

Let $\widetilde{\mathsf{DT}}(F\restriction_\rho)$ be the query complexity of the above decision tree. Since $\mathsf{DT}(F\restriction_\rho) \leq \widetilde{\mathsf{DT}}(F\restriction_\rho)$ ($\mathsf{DT}(F\restriction_\rho)$ is the minimum complexity among all decision trees computing $F\restriction_\rho$), it suffices to bound

$$\Pr_{\rho \sim \boldsymbol{\rho}}[\widetilde{\mathsf{DT}}(F\restriction_\rho) > s].$$

Consider the event that $\widetilde{\mathsf{DT}}(F\restriction_\rho) > s$, it is equivalent to that there exists a string $w \in \{0,1\}^s$, such that if we fix the first s queried unrestricted bits in ρ according to w, Eval ends up querying $> s$ bits. (Note that since we only care about whether $\widetilde{\mathsf{DT}}(F\restriction_\rho) > s$, we can force the algorithm to abort if it tries to make the $(s+1)$-th query.)

Now, suppose we fix the string w, then the number of queries made by Eval only depends on ρ. Suppose the algorithm has queried at least $s+1$ bits, we let $D'_1, D'_2, \ldots, D'_t$ ($t \leq s+1$) be the decision trees in which the algorithm made queries during the first $s+1$ queries. This implies that if we run $\mathsf{Eval}(\rho, D'_1, D'_2, \ldots, D'_t)$ with respect to the same string w, the algorithm also makes at least $s+1$ queries.

Now, since $\boldsymbol{\rho}$ is $k(s+1)$-wise independent. The probability that $\mathsf{Eval}(\rho, D'_1, D'_2, \ldots, D'_t)$ makes at least $s+1$ queries with respect to the fixed string w is bounded by

$$\begin{aligned}
&(T \cdot k^2)^{-(s+1)} \cdot \binom{t \cdot k}{s+1} \\
\leq &(T \cdot k^2)^{-(s+1)} \cdot \left(\frac{t \cdot k \cdot e}{s+1}\right)^{s+1} \\
\leq &\left(\frac{T \cdot k \cdot (s+1)}{t \cdot e}\right)^{-(s+1)} \leq \left(\frac{T \cdot k}{e}\right)^{-(s+1)}.
\end{aligned}$$

Putting everything together, we have

$$\begin{aligned}
&\Pr_{\rho \sim \boldsymbol{\rho}}[\widetilde{\mathsf{DT}}(F\restriction_\rho) > s] \\
\leq &2^s \cdot \left(\frac{T \cdot k}{e}\right)^{-(s+1)} \cdot \sum_{t=0}^{s+1} \binom{T}{t} \\
\leq &2^s \cdot \left(\frac{T \cdot k}{e}\right)^{-(s+1)} \cdot \left(\frac{T \cdot e}{s+1}\right)^{s+1} \\
\leq &\left(\frac{k \cdot (s+1)}{2e^2}\right)^{-(s+1)}.
\end{aligned}$$

◀

▶ Remark 57. Clearly, Lemma 56 also holds when $\boldsymbol{\rho}$ is $k(s+1)$-wise independent and p-regular, for $p \leq \frac{1}{T \cdot k^2}$.

Now we are ready to prove Theorem 52.

Proof of Theorem 52. We assume N and $\log N$ are both powers of 2 for simplicity. Let $p = 1/\log^5 N$, $\varepsilon_0 = 2^{-\log^6 N}$, $s = t = 10\log^2 N$, $M = 2^s \cdot N^{\log N}$, and $\boldsymbol{\rho}$ be the k-wise independent p-regular random restriction guaranteed by Lemma 55. Note that we have $k = \omega(\log^6 N)$ and $k = \log^{O(1)} N$.

Let $C \in \mathsf{AC}^0_d[O_1, O_2, \ldots, O_d]$ be a circuit with S gates computing $\mathsf{MCSP}[n^c, n^{2c}]$. For each $i \in [d]$, let S_i be the number of gates at level i (i.e., the gates whose distance from the input gates is i). Recall that O_i is the maximum oracle fan-in at level i. We are going to prove the stronger claim that $S = \Omega(N^{\log N})$. Now, suppose for the sake of contradiction that $S \leq N^{\log N}/8$.

Now we proceed in d iterations. We will ensure that at the end of the i-th iteration, all gates at level i become s-query decision trees with high probability. At the i-th iteration, we apply ρ

$$\tau_i = \lceil \log_{1/p} O_i \rceil + 1$$

times. It is straightforward to see that the composition of τ_i independent restrictions from $\boldsymbol{\rho}$ is a k-wise independent p_i-regular random restriction for $p_i = p^{\tau_i} \leq \frac{1}{O_i \cdot \log^5 N}$.

Note that each oracle gate at original level i has inputs computed by s-query decision trees (at the first step, one can treat the input variables as 1-query decision trees). By Lemma 56 and noting that $k \geq s(s+1)$ and $O_i \cdot \log^5 N \geq O_i \cdot s^2$, with probability at least

$$1 - S_i \cdot \left(\frac{s(s+1)}{2e^2}\right)^{-(s+1)} \geq 1 - S_i \cdot N^{-\log N},$$

all oracle gates at level i become s-query decision trees after these τ_i restrictions.

Similarly, note that each AND / OR gate at level i are equivalent to a CNF or DNF with width-s and size at most $2^s \cdot S$. By Lemma 55, again with probability at least

$$\begin{aligned}
&1 - S_i \cdot \left(2^{s+t+1}(5pt)^s + \varepsilon_0 \cdot 2^{(s+1)(2t+\log M)}\right) \\
&\geq 1 - S_i \cdot \Big(2^{20\log^2 N+1}(5 \cdot (1/\log^5 N) \cdot 10\log^2 N)^{10\log^2 N} \\
&\qquad + 2^{-\log^6 N} \cdot 2^{(10\log^2 N+1)(20\log^2 N+\log(N^{\log N} \cdot 2^{10\log^2 N}))}\Big) \\
&\geq 1 - S_i \cdot N^{-\log N},
\end{aligned}$$

all AND / OR gates at level i become s-query decision tree after these τ_i restrictions.

Finally, note that in total we have applied $\boldsymbol{\rho}$ at most

$$\tau_{\mathsf{total}} = 2d + \log_{1/p}\left(\prod_{i=1}^{d} O_i\right) = \log_{1/p} N - \omega(1)$$

times, and the final output gate shrinks to an s-query decision tree with probability at least

$$1 - 2 \cdot S \cdot N^{-\log N}.$$

Since $S \leq N^{\log N}/8$, with probability at least $3/4$, after all these restrictions, C is equivalent to an s-query decision tree.

Now let $p_{\mathsf{end}} = p^{\tau_{\mathsf{total}}} = N^{-1} \cdot p^{-\omega(1)}$. By Chebyshev's inequality, the number of unrestricted variables at the end of the restriction is at least $N_{\mathsf{remain}} = \frac{1}{2} \cdot p_{\mathsf{end}} \cdot N = (\log N)^{\omega(1)}$ with probability at least $1/2$. Therefore, with probability at least $1/4$, at the end of the restrictions, it holds that the remaining circuit C is equivalent to an s-query decision tree D, and the number of unrestricted variables is at least N_{remain}.

Suppose we fix all these remaining unrestricted variables to be 0 to get an input x^*, since each restriction from $\boldsymbol{\rho}$ can be computed by a $\mathsf{poly}(n)$-size circuit, x^* has a circuit of $\mathsf{poly}(n) \cdot \log N = \mathsf{poly}(n) \leq n^c$ size (now we set c). Let S be the set of input variables that D queries on the input x^*. Note that there are at least $2^{N_{\mathsf{remain}}-|S|}$ ways of assigning values to unrestricted variables while keeping variables in S all 0. And we can see that F's output on x^* is the same as its output on all of these assignments. But there must exist at least one assignment such the MCSP value is at least $(\log N)^{2c} = n^{2c}$ ($2^{N_{\mathsf{remain}}-|S|} = 2^{n^{\omega(1)}}$), contradiction to the assumption that C computes $\mathsf{MCSP}[n^c, n^{2c}]$. ◀

5.2.2 The Nearly Quadratic Formula Lower Bound of [23]

In this section, we prove that the nearly quadratic formula lower bound of [23] localizes, and thereby proving the third item of Theorem 2. This localization indeed refutes a family of possible approaches to establish circuit lower bounds through hardness magnification via "lowering the threshold".

More concretely, consider the following hypothesized approach. Suppose we can compute $\mathsf{MCSP}[2^{\sqrt{n}}]$ by a formula F with NP oracles, such that when we replace every oracle O with fan-in β in F by a formula of size β^k which reads all its inputs exactly β^{k-1} times, the size of the new formula is less than $N^{1.99}$. Then we know that NP cannot be computed by formulas of size n^k which reads all its inputs exactly n^{k-1} times, as otherwise we get an $N^{1.99}$-size formula for $\mathsf{MCSP}[2^{\sqrt{n}}]$, which is a contradiction to the lower bound in [23]. If this holds for all $k > 0$, then we would have $\mathsf{NP} \not\subset \mathsf{Formula}[n^k]$ for all k.

In the following, by localizing [23], we show that there is no such oracle formula construction for MCSP even if the oracles can be arbitrary. This excludes magnification theorems obtained by approaches that unconditionally produce circuits with oracles, and essentially addresses a question from [39]. It also suggests that the consideration of almost-formulas in HM Frontier C is unavoidable.

A Size Measure on Oracle Formulas and A Potential Approach to Formula Size Lower Bound

We first introduce a size measure Size_t on oracle formulas to formalize the previous discussion.

For a parameter t and an oracle formula F, we define $\mathsf{Size}_t(F)$ as the size of the formula, if we replace every oracle O with fan-in β in F by a formula of size β^t which reads all its inputs exactly β^{t-1} times.

More formally,

$$\mathsf{SIZE}_t(F) := \begin{cases} \mathsf{SIZE}_t(F_1) + \mathsf{SIZE}_t(F_2) & F = F_1 \wedge F_2 \text{ or } F = F_1 \vee F_2, \\ \beta^{t-1} \cdot \left(\sum_{i=1}^{\beta} \mathsf{SIZE}_t(F_i)\right) & F = O(F_1, F_2, \ldots, F_\beta). \end{cases}$$

▶ **Proposition 58.** *For a constant $k > 0$, if there is an NP oracle formula F (all oracles are languages in NP) for $\mathsf{MCSP}[2^{\sqrt{n}}]$ such that $\mathsf{SIZE}_{k+1}(F) \le N^{2-\varepsilon}$ for a constant $\varepsilon > 0$, then $\mathsf{NP} \not\subseteq \mathsf{Formula}[n^k]$.*

Proof. Suppose $\mathsf{NP} \subseteq \mathsf{Formula}[n^k]$ for the sake of contradiction. Then in particular each NP language can be computed by a size-n^{k+1} formula which reads all its inputs exactly n^k times by adding some dummy nodes in the formula. Therefore, by replacing all NP oracles in F by such formulas, we have an $N^{2-\varepsilon}$-size formula for $\mathsf{MCSP}[2^{\sqrt{n}}]$, in contradiction to the lower bound in [23]. ◀

Localization of [23]

Our following theorem shows that the above approach is not viable even with $k = 3$ by localizing [23], with a moderate constraint on the adaptivity of the oracle circuits.

▶ **Theorem 59.** *There is a universal constant c such that for all constants $\varepsilon > 0$ and $\alpha > 2$, $\mathsf{MCSP}[n^c, 2^{(\varepsilon/\alpha)\cdot n}]$ cannot be computed by oracle formulas F with $\mathsf{SIZE}_3(F) \le N^{2-\varepsilon}$ and adaptivity $o(\log N/\log\log N)$ (that is, on any path from root to a leaf, there are at most $o(\log N/\log\log N)$ oracles).*

▶ Remark 60. It is not hard to see that the adaptivity can be at most $O(\log N)$ given the condition $\mathsf{SIZE}_3(F) \le N^{2-\varepsilon}$.

Before proving Theorem 59, we first show it refutes the Anti-Checker Hypothesis (restated below) from [39].

The Anti-Checker Hypothesis. *For every $\lambda \in (0,1)$, there are $\varepsilon > 0$ and a collection $\mathcal{Y} = \{Y_1, \ldots, Y_\ell\}$ of sets $Y_i \subseteq \{0,1\}^n$, where $\ell = 2^{(2-\varepsilon)n}$ and each $|Y_i| = 2^{n^{1-\varepsilon}}$, for which the following holds.*

If $f : \{0,1\}^n \mapsto \{0,1\}$ and $f \notin \mathsf{Circuit}[2^{n^\lambda}]$, then some set $Y \in \mathcal{Y}$ forms an anti-checker for f: For each circuit C of size $2^{n^\lambda}/10n$, there is an input $y \in Y$ such that $C(y) \neq f(y)$.

▶ **Corollary 61.** *The Anti-Checker Hypothesis is false.*

Proof. It is easy to see that, assuming the Anti-Checker Hypothesis, we can solve $\mathsf{MCSP}[2^{n^{1/3}}, 2^{n^{2/3}}]$ with a formula F of $N^{2-\varepsilon}$ size which uses $N^{2-\varepsilon}$ oracles of fan-in $\mathsf{poly}(n)2^{n^{1-\varepsilon}} = \mathrm{polylog}(N) \cdot 2^{(\log N)^{1-\varepsilon}} = N^{o(1)}$ only at the layer above the inputs, for some $\varepsilon > 0$. However, since $\mathsf{SIZE}_3(F) \leq N^{2-\varepsilon+o(1)}$, F cannot compute $\mathsf{MCSP}[2^{n^{1/3}}, 2^{n^{2/3}}]$ by Theorem 59, contradiction. ◀

Now we are ready to prove Theorem 59.

Proof of Theorem 59. Let $k = \log^3 N$, and $\boldsymbol{\rho}$ be the k-wise independent $(1/\sqrt{k})$-regular random restriction guaranteed by Lemma 12.

For an oracle formula F and a sub-formula G of it, we say G is a maximal sub-formula if G is an entire subtree rooted at either the root, an oracle gate, or a gate whose father is an oracle.

We are going to apply $t = \Theta(\log_k N)$ independent pseudorandom restrictions $\boldsymbol{\rho}_1, \boldsymbol{\rho}_2, \ldots, \boldsymbol{\rho}_t$, each distributed identically to $\boldsymbol{\rho}$, where t will be set precisely later.

The Overall Proof Structure

To analyze the size of the oracle formula under the random restriction sequence $\boldsymbol{\rho}_1, \boldsymbol{\rho}_2, \ldots, \boldsymbol{\rho}_t$, we define a potential function Φ inductively for all maximal sub-formulas of the given formula F. As it will be clear from the definition, Φ is not only a function of the structure of the oracle formula, but also depends on the history of the pseudorandom restrictions.

Formally, for each maximal sub-formula G of the given formula F, and for each integer $0 \leq i \leq t$, we define a random variable $\Phi_{G,i}$, which denotes the potential function of G after the first i pseudorandom restrictions and only depends on $\boldsymbol{\rho}_1, \boldsymbol{\rho}_2, \ldots, \boldsymbol{\rho}_i$.

Definition of Tiny formulas and Blow up

For an oracle formula, if the top gate is an oracle, we say it is tiny if it depends on at most $\log N$ variables. Otherwise, we say it is tiny if it depends on at most $c_{\mathsf{tiny}} \cdot k$ variables, for a constant c_{tiny} to be specified later.

After each pseudorandom restriction, for a formula with an oracle gate at the top, when it depends on at most $b = 20$ variables, we blow it up to a formula of size $B = 2^b$ (note that if there are two oracle gates u, v such that u and v both depend on at most b variables and u is an ancestor of v, then it suffices to only blow up u).

The above two definitions (tiny formulas and the process of blowing up) may not seem easy to understand at first. Let us explain the motivation behind them. The key difficulty of the proof is to handle the oracle gates properly. The process of blowing up ensures that whenever an oracle becomes too small, we just replace it with a constant size normal formula, so it becomes easier to deal with.

The definition of tiny formulas is more subtle. As it will be clearly in Case II and Case III of the inductive definition of Φ, setting the threshold of being tiny to $\log N$ for oracle formulas with top oracle gates ensures that the corresponding event of becoming tiny happens with high probability, which is indeed crucial in our proof.

The properties of Φ

We require the following properties on Φ.

1. For an oracle formula F, Φ is multiplied by a factor of $\frac{c_F}{k}$ under $\boldsymbol{\rho}$ in expectation, where c_F depends on F but it is upper bounded by a universal constant.
2. With probability $1 - p_F$, for all stages, and all maximal sub-formulas G of F, $\Phi = 0$ for G implies that G is tiny, where p_F depends on F but it is upper bounded by N^{-2}.
3. It holds that either $\Phi = 0$ or $\Phi \geq 1$. Together with the second item, it implies that if the oracle formula is not tiny then $\Phi \geq 1$.

With these carefully designed properties of Φ, the overall proof is straightforward. We first show that Φ of F is closely related to $\mathsf{SIZE}_3(F)$, and our conditions on the oracle formula imply that Φ of the whole oracle formula is bounded by $N^{2-\varepsilon+o(1)}$ at the beginning. Then after roughly $t \approx \log_k(N^{2-\varepsilon+o(1)})$ rounds of restrictions from $\boldsymbol{\rho}$, Φ becomes 0 with a good probability, which also implies the whole oracle formula becomes tiny (only depend on $\mathrm{polylog}(N)$ bits).

But then we argue that after t rounds of restrictions from $\boldsymbol{\rho}$, with high probability the number of unrestricted variables is still at least $N^{\Omega(1)}$. Using a similar argument as from [23, 42, 39], we show that the tiny oracle formula left behind cannot compute $\mathsf{MCSP}[n^c, 2^{\varepsilon/\alpha \cdot n}]$ on the remaining variables, which concludes the proof.

The Inductive Definition of the Potential Function Φ

In the following, we gradually develop the definition of the potential function Φ. We remark that Case I and Case II below are actually special cases of Case III and Case IV respectively. We discuss them first in the hope that they provide some intuitions and make it easier to understand the more complicated Case III and Case IV.

Case I: Φ for a Pure Formula

We begin with the simplest case of pure formulas F (formulas with no oracles) of size S. We define

$$\Phi = \begin{cases} S & S \geq 100 \cdot k, \\ 0 & \text{otherwise.} \end{cases}$$

It follows from the shrinkage lemma [21], formula decomposition [51, Claim 6.2], and the k-wise independence of $\boldsymbol{\rho}$ that, when $S \geq 100 \cdot k$, the *expected size* of S drops by a factor of at least k/c_{Tal}, for a universal constant c_{Tal} (we can set $c_F = c_{\mathsf{Tal}}$). Otherwise, the formula is tiny. It is straightforward to verify that all three properties of Φ are satisfied (we can set $p_F = 0$ in this case).

Case II: Φ for a Pure Oracle

Next we consider the case that F is a pure oracle O with fan-in T (pure oracle means each input to O is just a variable). We set

$$\Phi = T^2 \cdot k^3$$

at the beginning. And set $\Phi \leftarrow \Phi/k$ after each $\boldsymbol{\rho}$. Whenever it happens $\Phi < 1$, we set $\Phi = 0$ afterwards. Here, we can simply set $c_F = 1$.

Now we argue that with probability at least $1 - N^{-5}$ (that is, we set $p_F = N^{-5}$), when $\Phi = 0$, O only depends on at most $\log N$ variables and therefore becomes tiny.

Note that $\Phi = 0$ means at least $\log_k T^2$ rounds of random restrictions have been applied.[16] Their composition is a k-wise independent restriction which keeps a variable unrestricted with probability at most T^{-1}. Therefore, the probability that the number of alive variable is larger than $\log N$ is smaller than

$$\binom{T}{\log N} \cdot T^{-\log N} \leq \left(\frac{e \cdot T}{\log N}\right)^{\log N} \cdot T^{-\log N} \leq \left(\frac{e}{\log N}\right)^{\log N} \leq N^{-5}.$$

Note that in the above inequalities we can safely assume $T > \log N$.

Case III: Φ for an Oracle Formula with an Oracle Top Gate

Then we move to the case of a maximal sub-formula F with an oracle top gate O with fan-in T. Let Φ_i be the corresponding potential function of the maximal sub-formula with root being the i-th input to O. We set

$$\Phi = \max\left(\sum_{i=1}^{T} \Phi_i, 1/k\right) \cdot T^2 \cdot k^4,$$

at the beginning.

When $\sum_{i=1}^{T} \Phi_i > 0$, we still let $\Phi = \left(\sum_{i=1}^{T} \Phi_i\right) \cdot T^2 \cdot k^4$. When $\sum_{i=1}^{T} \Phi_i$ first becomes 0 (this could happen before the first restriction, if $\sum_{i=1}^{T} \Phi_i = 0$ at the beginning), we set $\Phi = T^2 \cdot k^3$ and decrease it by a factor of k during each later restriction, and set it to 0 if it becomes < 1.

Here, we set c_F to be the maximum of $c_{F'}$ for all maximal sub-formulas F' whose root is an input to the top oracle gate O in F.

First let us argue that Φ is multiplied by a factor of $\frac{c_F}{k}$ after each $\boldsymbol{\rho}$ in expectation. When $\sum_{i=1}^{T} \Phi_i = 0$, it is evident from the way we set Φ (note that $c_F \geq 1$). When $\sum_{i=1}^{T} \Phi_i > 0$, it follows from the induction as each Φ_i is multiplied by a factor of $\frac{c_F}{k}$ after each $\boldsymbol{\rho}$ in expectation. In the borderline case when $\sum_{i=1}^{T} \Phi_i > 0$ before $\boldsymbol{\rho}$ and becomes 0 afterwards. One can see Φ drops from at least $T^2 \cdot k^4$ to at most $T^2 \cdot k^3$.

Moreover, when $\sum_{i=1}^{T} \Phi_i = 0$, with probability at least $1 - \sum_{i=1}^{T} p_{F_i}$ (F_i is the i-th sub-formula whose root is an input to the top oracle gate O in F) all the sub-formulas are tiny, so at this time the oracle depends on at most $O(T \cdot k)$ variables.

Therefore, when Φ drops to 0, with probability at least $1 - \sum_{i=1}^{T} p_{F_i} - N^{-5}$ the whole oracle formula becomes tiny, by a calculation similar to the pure oracle case. Therefore, we can set $p_F = \sum_{i=1}^{T} p_{F_i} + N^{-5}$.

Case IV: Φ for a Formula with Oracle Leaves

Finally, we deal with the most complicated case when the maximal sub-formula F is a formula with oracle leaves. Suppose F is a formula of size S with m oracle leaves. Let Φ_i be the potential function of the sub-formula corresponding to the i-th oracle leaf. Also, let c_{drop} be the maximum of the c_F's of all the sub-formulas corresponding to the oracle leaves.

[16] Note that for this argument, a potential function of $T \cdot k$ already suffices. We use $T^2 \cdot k^3$ here to make it consistent with Case III.

The difficulty in analyzing this case is that there could be many oracles which are tiny but have not blown up yet, and we have to keep track of the number of such oracles. Let N_{active} be the number of remaining active tiny oracles (oracles which are tiny but have not blown up). Clearly, $N_{\mathsf{active}} \leq S$ at the beginning.

We set

$$\Phi = S + N_{\mathsf{active}} \cdot k^2 + \sum_{i=1}^{m} \Phi_i \cdot k^4,$$

at the beginning. When $S \leq 100 \cdot k$ happens, we change Φ to be

$$N_{\mathsf{active}} \cdot k^2 + \sum_{i=1}^{m} \Phi_i \cdot k^4$$

afterwards.

After each $\boldsymbol{\rho}$, if $S \geq 100 \cdot k$, the expected size of S becomes at most

$$c_1 \cdot S/k + c_2 \cdot k \cdot \left(\sum_{i=1}^{m} \Phi_i + N_{\mathsf{active}} \right),$$

for two universal constants c_1 and c_2. This bound holds because, by Claim 4.4 of [24], a formula of size S can be decomposed into $6S/k$ sub-formulas, each of size at most k, and each formula has at most 2 sub-formula children.

The number of active oracle leaves (which are not blown up) is at most $\sum_{i=1}^{m} \Phi_i + N_{\mathsf{active}}$. Hence, at least $6 \cdot S/k - \sum_{i=1}^{m} \Phi_i - N_{\mathsf{active}}$ sub-formulas do not contain an active oracle leaf, and their total expected size is $O(S/k)$ after $\boldsymbol{\rho}$ (by Lemma 4.1 and Lemma 4.3 of [24], and [51]). For those sub-formulas containing active oracle leaves, their total size is at most $(\sum_{i=1}^{m} \Phi_i + N_{\mathsf{active}}) \cdot O(k)$ after $\boldsymbol{\rho}$ (this takes account of the worst case situation that all these active oracle leaves blow up).

Also, we can see that after $\boldsymbol{\rho}$, N_{active} becomes at most

$$N_{\mathsf{active}}/k^2 + \sum_{i=1}^{m} \Phi_i$$

in expectation. This is because for a tiny active oracle depending on at most $\log N$ variables, the probability that it does not blow up after $\boldsymbol{\rho}$ is at most

$$\binom{\log N}{b} \cdot k^{-b/2} \leq (\log N)^{b - 1.5 \cdot b} = (\log N)^{-10} \leq 1/k^2.$$

By induction, we also have that $\sum_{i=1}^{m} \Phi_i$ is multiplied by a factor of $\frac{c_{\mathsf{drop}}}{k}$ in expectation as well after each $\boldsymbol{\rho}$. Therefore, after $\boldsymbol{\rho}$, the expectation of Φ can be bounded by

$$\begin{aligned} & c_1 \cdot S/k + c_2 k \left(\sum_{i=1}^{m} \Phi_i + N_{\mathsf{active}} \right) + \left(N_{\mathsf{active}}/k^2 + \sum_{i=1}^{m} \Phi_i \right) \cdot k^2 + \left(\sum_{i=1}^{m} \Phi_i \right) \cdot \frac{c_{\mathsf{drop}}}{k} \cdot k^4 \\ \leq \; & S \cdot \frac{c_1}{k} + N_{\mathsf{active}} \cdot k^2 \cdot \frac{c_2 + 1/k}{k} + \sum_{i=1}^{m} \Phi_i \cdot k^4 \cdot \left(\frac{c_{\mathsf{drop}} + c_2/k^2 + 1/k}{k} \right). \end{aligned}$$

We can set

$$c_F = \max(c_1, c_2 + 1/k, c_{\mathsf{drop}} + c_2/k^2 + 1/k).$$

Recall that when $S \leq 100 \cdot k$ happens, we change Φ to be

$$N_{\mathsf{active}} \cdot k^2 + \sum_{i=1}^{m} \Phi_i \cdot k^4$$

afterwards.

By the previous discussion, after this Φ still drops by a factor of k/c_F in expectation after each $\boldsymbol{\rho}$. Note that when $\Phi = 0$, we can see the size of the whole formula is smaller than $B \cdot 100 \cdot k = O(k)$, therefore it is tiny (here we set $c_{\mathsf{tiny}} = B \cdot 100$). This is because $\Phi = 0$ implies $S \leq 100 \cdot k$ happened at some point, and also $N_{\mathsf{active}} = \sum_{i=1}^{m} \Phi_i = 0$. They together imply that all oracles have blown up, and the size bound follows since each oracle adds at most B leaves.

Let F_i be the sub-formula with root being the i-th oracle leaf. In this case, we can set $p_F = \sum_{i=1}^{m} p_{F_i}$.

The MCSP Lower Bound

Let F be an oracle formula with $\mathsf{SIZE}_3(F) \leq N^{2-\varepsilon}$ and adaptivity $\tau = o(\log N / \log\log N)$. We first need to verify that c_F is upper bounded by a universal constant. One can upper bound

$$c_F \leq \max(c_1, c_2 + 1/k, c_{\mathsf{Tal}}) + \tau \cdot (c_2/k^2 + 1/k) \leq \max(c_1, c_2 + 1/k, c_{\mathsf{Tal}}) + o(1) = O(1).$$

We can also upper bound p_F by $p_F \leq N^{-5} \cdot N^2 = N^{-3}$.

By the inductive definition of the potential function Ψ on maximal sub-formulas, it is not hard to show that

$$\Phi \leq \mathsf{SIZE}_3(F) \cdot k^{O(\tau)} \leq N^{2-\varepsilon+o(1)}.$$

Note that this inequality crucially employs the definition of $\mathsf{SIZE}_3(\cdot)$.

After each $\boldsymbol{\rho}$, Φ is reduced by a factor of k/c_F. After

$$t = \lceil \log_{k/c_F} \Phi \rceil + 2$$

rounds of $\boldsymbol{\rho}$, the expected Φ of the overall formula becomes $< 1/10$, which means with probability $0.9 - p_F \geq 0.8$ it is tiny and only depends on at most $O(k) = O(\log^3 N)$ variables.

Note that by definition

$$(k/c_F)^t \leq \Phi \cdot k^3,$$

and therefore

$$k^t \leq \Phi \cdot k^3 \cdot (c_F)^t \leq N^{2-\varepsilon+o(1)},$$

as $(c_F)^t = (c_F)^{O(\log N / \log\log N)} = N^{o(1)}$.

The composition of t independent $\boldsymbol{\rho}$ keeps a variable unrestricted with probability $k^{-t/2} \geq N^{-1+\varepsilon/2-o(1)}$, and is clearly pairwise independent. By Chebyshev's inequality, after t restrictions from $\boldsymbol{\rho}$, with probability 0.5, at least

$$1/2 \cdot N \cdot N^{-1+\varepsilon/2-o(1)} \geq N^{\varepsilon/2-o(1)}$$

variables remain active. So with probability at least 0.3, after t restrictions from $\boldsymbol{\rho}$, the remaining formula F only depends on $O(\log^3 N)$ variables, and the number of remaining unrestricted variables is at least $N^{\varepsilon/2-o(1)}$.

Suppose we fix all these remaining unrestricted variables to be 0 to get an input x^*. Since each restriction from $\boldsymbol{\rho}$ can be computed by a $\mathsf{poly}(n)$-size circuit, x^* has a circuit of $\mathsf{poly}(n) \cdot t = \mathsf{poly}(n) \leq n^c$ size (here we set c). Let S be the set of input variables that F depends on. Note that there are at least $2^{N^{\varepsilon/2-o(1)}-|S|}$ ways of assigning values to unrestricted variables while keeping variables in S all 0. Since F only depends on S, F's output on x^* is the same as its output on all of these assignments. But there must exist at least one assignment such the MCSP value is at least $N^{\varepsilon/\alpha} = 2^{(\varepsilon/\alpha)\cdot n}$ as $\alpha > 2$. Therefore, F cannot compute $\mathsf{MCSP}[n^c, 2^{(\varepsilon/\alpha)\cdot n}]$. ◀

References

1 Miklós Ajtai. Σ_1^1-formulae on finite structures. *Annals of Pure and Applied Logic*, 24(1):1–48, 1983.

2 Eric Allender, Lisa Hellerstein, Paul McCabe, Toniann Pitassi, and Michael E. Saks. Minimizing Disjunctive Normal Form Formulas and AC^0 Circuits Given a Truth Table. *SIAM J. Comput.*, 38(1):63–84, 2008. doi:10.1137/060664537.

3 Eric Allender and Michal Koucký. Amplifying lower bounds by means of self-reducibility. *J. ACM*, 57(3):14:1–14:36, 2010.

4 Alexander E. Andreev and Stasys Jukna. Very large cliques are easy to detect. *Discrete Mathematics*, 308(16):3717–3721, 2008. doi:10.1016/j.disc.2007.07.036.

5 Benny Applebaum, Boaz Barak, and David Xiao. On Basing Lower-Bounds for Learning on Worst-Case Assumptions. In *Symposium on Foundations of Computer Science* (FOCS), pages 211–220, 2008.

6 Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

7 Louay M. J. Bazzi. Polylogarithmic Independence Can Fool DNF Formulas. *SIAM J. Comput.*, 38(6):2220–2272, 2009. doi:10.1137/070691954.

8 Marco L. Carmosino, Russell Impagliazzo, Valentine Kabanets, and Antonina Kolokolova. Learning Algorithms from Natural Proofs. In *Conference on Computational Complexity* (CCC), pages 10:1–10:24, 2016. doi:10.4230/LIPIcs.CCC.2016.10.

9 Arkadev Chattopadhyay and Rahul Santhanam. Lower Bounds on Interactive Compressibility by Constant-Depth Circuits. In *Symposium on Foundations of Computer Science* (FOCS), pages 619–628, 2012.

10 Lijie Chen, Ce Jin, and Ryan Williams. Hardness Magnification for all Sparse NP Languages. In *Symposium on Foundations of Computer Science* (FOCS), 2019.

11 Lijie Chen, Ce Jin, and Ryan Williams. Sharp Threshold Results for Computational Complexity. In *Unpublished manuscript*, 2019.

12 Lijie Chen, Dylan M. McKay, Cody D. Murray, and R. Ryan Williams. Relations and Equivalences Between Circuit Lower Bounds and Karp-Lipton Theorems. In *Computational Complexity Conference* (CCC), 2019.

13 Lijie Chen and Roei Tell. Bootstrapping Results for Threshold Circuits "Just Beyond" Known Lower Bounds. In *Symposium on Theory of Computing* (STOC), 2019.

14 Xi Chen, Igor Carboni Oliveira, and Rocco A. Servedio. Addition is exponentially harder than counting for shallow monotone circuits. In *Symposium on Theory of Computing* (STOC), pages 1232–1245, 2017.

15 Mahdi Cheraghchi, Elena Grigorescu, Brendan Juba, Karl Wimmer, and Ning Xie. $AC^0 \circ MOD_2$ lower bounds for the Boolean Inner Product. *J. Comput. Syst. Sci.*, 97:45–59, 2018.

16 Mahdi Cheraghchi, Valentine Kabanets, Zhenjian Lu, and Dimitrios Myrisiotis. Circuit Lower Bounds for MCSP from Local Pseudorandom Generators. In *International Colloquium on Automata, Languages, and Programming* (ICALP), pages 39:1–39:14, 2019. doi:10.4230/LIPIcs.ICALP.2019.39.

17 Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013. doi:10.1007/978-1-4471-5559-1.

18 Jürgen Forster. A linear lower bound on the unbounded error probabilistic communication complexity. *J. Comput. Syst. Sci.*, 65(4):612–625, 2002. doi:10.1016/S0022-0000(02)00019-3.

19 Merrick L. Furst, James B. Saxe, and Michael Sipser. Parity, Circuits, and the Polynomial-Time Hierarchy. *Mathematical Systems Theory*, 17(1):13–27, 1984. doi:10.1007/BF01744431.

20 Mika Göös, Pritish Kamath, Robert Robere, and Dmitry Sokolov. Adventures in Monotone Complexity and TFNP. In *Innovations in Theoretical Computer Science Conference* (ITCS), pages 38:1–38:19, 2019.

21 Johan Håstad. The Shrinkage Exponent of de Morgan Formulas is 2. *SIAM J. Comput.*, 27(1):48–64, 1998. doi:10.1137/S0097539794261556.

22 Shuichi Hirahara. Non-Black-Box Worst-Case to Average-Case Reductions within NP. In *Symposium on Foundations of Computer Science* (FOCS), pages 247–258, 2018.

23 Shuichi Hirahara and Rahul Santhanam. On the Average-Case Complexity of MCSP and Its Variants. In *Computational Complexity Conference* (CCC), pages 7:1–7:20, 2017. doi:10.4230/LIPIcs.CCC.2017.7.

24 Russell Impagliazzo, Raghu Meka, and David Zuckerman. Pseudorandomness from Shrinkage. *J. ACM*, 66(2):11:1–11:16, 2019. doi:10.1145/3230630.

25 Russell Impagliazzo, Ramamohan Paturi, and Michael E. Saks. Size-Depth Tradeoffs for Threshold Circuits. *SIAM J. Comput.*, 26(3):693–707, 1997.

26 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which Problems Have Strongly Exponential Complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001. doi:10.1006/jcss.2001.1774.

27 Russell Impagliazzo and Avi Wigderson. Randomness vs Time: Derandomization under a Uniform Assumption. *J. Comput. Syst. Sci.*, 63(4):672–688, 2001. doi:10.1006/jcss.2001.1780.

28 Emil Jerábek. Approximate counting by hashing in bounded arithmetic. *J. Symb. Log.*, 74(3):829–860, 2009. doi:10.2178/jsl/1245158087.

29 Stasys Jukna. *Boolean Function Complexity - Advances and Frontiers*. Springer, 2012.

30 Jørn Justesen. Class of constructive asymptotically good algebraic codes. *IEEE Trans. Information Theory*, 18(5):652–656, 1972. doi:10.1109/TIT.1972.1054893.

31 Mauricio Karchmer and Avi Wigderson. Monotone Circuits for Connectivity Require Super-Logarithmic Depth. *SIAM J. Discrete Math.*, 3(2):255–265, 1990. doi:10.1137/0403021.

32 Ilan Komargodski and Raz Ran. Average-case lower bounds for formula size. In *Symposium on Theory of Computing* (STOC), 2013.

33 Swastik Kopparty. On the complexity of powering in finite fields. In *Proceedings of the Symposium on Theory of Computing* (STOC), pages 489–498, 2011. doi:10.1145/1993636.1993702.

34 Jan Krajíček. *Forcing with random variables and proof complexity*. Cambridge University Press, 2011.

35 Richard J. Lipton and Ryan Williams. Amplifying circuit lower bounds against polynomial time, with applications. *Computational Complexity*, 22(2):311–343, 2013.

36 Dylan M. McKay, Cody D. Murray, and R. Ryan Williams. Weak Lower Bounds on Resource-Bounded Compression Imply Strong Separations of Complexity Classes. In *Symposium on Theory of Computing* (STOC), 2019.

37 Moritz Müller and Ján Pich. Feasibly constructive proofs of succinct weak circuit lower bounds. *Electronic Colloquium on Computational Complexity* (ECCC), 24:144, 2017. URL: https://eccc.weizmann.ac.il/report/2017/144.

38 Igor Carboni Oliveira. Randomness and Intractability in Kolmogorov Complexity. In *International Colloquium on Automata, Languages, and Programming* (ICALP), pages 32:1–32:14, 2019.

39 Igor Carboni Oliveira, Ján Pich, and Rahul Santhanam. Hardness magnification near state-of-the-art lower bounds. In *Computational Complexity Conference* (CCC), 2019.

40 Igor Carboni Oliveira and Rahul Santhanam. Majority is Incompressible by $\mathrm{AC}^0[p]$ Circuits. In *Conference on Computational Complexity* (CCC), pages 124–157, 2015.

41 Igor Carboni Oliveira and Rahul Santhanam. Conspiracies Between Learning Algorithms, Circuit Lower Bounds, and Pseudorandomness. In *Computational Complexity Conference* (CCC), pages 18:1–18:49, 2017. doi:10.4230/LIPIcs.CCC.2017.18.

42 Igor Carboni Oliveira and Rahul Santhanam. Hardness Magnification for Natural Problems. In *Symposium on Foundations of Computer Science* (FOCS), pages 65–76, 2018. doi:10.1109/FOCS.2018.00016.

43 Alexander A. Razborov. Lower bounds on the monotone complexity of some Boolean functions. *Doklady Akademii Nauk SSSR*, 281:798–801, 1985. English translation in: Soviet Mathematics Doklady 31:354–357, 1985.

44 Alexander A. Razborov. Lower bounds on the size of constant-depth networks over a complete basis with logical addition. *Mathematicheskie Zametki*, 41(4):598–607, 1987.

45 Alexander A. Razborov. Pseudorandom Generators Hard for k-DNF Resolution and Polynomial Calculus Resolution. *Annals of Mathematics*, 181(2):415–472, 2015.

46 Alexander A. Razborov and Steven Rudich. Natural Proofs. *J. Comput. Syst. Sci.*, 55(1):24–35, 1997. doi:10.1006/jcss.1997.1494.

47 Ben W. Reichardt. Reflections for quantum query algorithms. In *Symposium on Discrete Algorithms* (SODA), pages 560–569, 2011.

48 Michael Sipser and Daniel A. Spielman. Expander codes. *IEEE Trans. Information Theory*, 42(6):1710–1722, 1996. doi:10.1109/18.556667.

49 Roman Smolensky. Algebraic Methods in the Theory of Lower Bounds for Boolean Circuit Complexity. In *Symposium on Theory of Computing* (STOC), pages 77–82, 1987.

50 Aravind Srinivasan. On the approximability of clique and related maximization problems. *J. Comput. Syst. Sci.*, 67(3):633–651, 2003.

51 Avishay Tal. Shrinkage of De Morgan Formulae by Spectral Techniques. In *Symposium on Foundations of Computer Science* (FOCS), pages 551–560, 2014. doi:10.1109/FOCS.2014.65.

52 Avishay Tal. Formula lower bounds via the quantum method. In *Symposium on Theory of Computing* (STOC), pages 1256–1268, 2017. doi:10.1145/3055399.3055472.

53 Avishay Tal. Tight Bounds on the Fourier Spectrum of AC^0. In *Computational Complexity Conference* (CCC), pages 15:1–15:31, 2017. doi:10.4230/LIPIcs.CCC.2017.15.

54 Luca Trevisan and Tongke Xue. A Derandomized Switching Lemma and an Improved Derandomization of AC^0. In *Conference on Computational Complexity* (CCC), pages 242–247, 2013. doi:10.1109/CCC.2013.32.

55 Salil P. Vadhan. Pseudorandomness. *Foundations and Trends in Theoretical Computer Science*, 7(1-3):1–336, 2012. doi:10.1561/0400000010.

56 Andrew Chi-Chih Yao. Circuits and Local Computation. In *Symposium on Theory of Computing* (STOC), pages 186–196, 1989. doi:10.1145/73007.73025.

A Review of Hardness Magnification in Circuit Complexity

A.1 Previous Work

We focus on some representative examples. For definitions and more details, check Section 2 or consult the original papers.

Srinivasan [50] (Informal). If there exists $\varepsilon > 0$ such that $n^{1-o(1)}$-approximating MAX-CLIQUE requires boolean circuits of size at least $m^{1+\varepsilon}$ (where $m = \Theta(n^2)$), then $\mathsf{NP} \not\subseteq \mathsf{Circuit}[\mathsf{poly}]$.

Allender-Koucký [3] and Chen-Tell [13]. The following results hold.

- Let $\Pi \in \{\mathsf{BFE}, \mathsf{W}_{\mathsf{S}_5}, \text{W5-STCONN}\}$. Suppose that for each $c > 1$ there exist infinitely many $d \in \mathbb{N}$ such that TC^0 circuits of depth d require more than $n^{1+c^{-d}}$ wires to solve Π. Then, $\mathsf{NC}^1 \not\subseteq \mathsf{TC}^0$.
- Suppose that for each $c > 1$ there exist infinitely many $d \in \mathbb{N}$ such that MAJ cannot be computed by ACC^0 circuits of depth d with $n^{1+c^{-d}}$ wires. Then $\mathsf{MAJ} \notin \mathsf{ACC}^0$, and consequently $\mathsf{TC}^0 \not\subseteq \mathsf{ACC}^0$.

Lipton-Williams [35]. If there is $\varepsilon > 0$ such that for every $\delta > 0$ we have $\mathsf{CircEval} \notin \mathsf{Size\text{-}Depth}[n^{1+\varepsilon}, n^{1-\delta}]$, then for every $k \geq 1$ and $\gamma > 0$ we have $\mathsf{CircEval} \notin \mathsf{Size\text{-}Depth}[n^k, n^{1-\gamma}]$ (in particular $\mathsf{P} \not\subseteq \mathsf{NC}$).

Oliveira-Santhanam [42]. The following results hold.

- Let $s(n) = n^k$ and $\delta(n) = n^{-k}$, where $k \in \mathbb{N}$. If $\mathsf{MCSP}[(s,0),(s,\delta)] \notin \mathsf{Formula}[N^{1+\varepsilon}]$ for some $\varepsilon > 0$, then there is $L \in \mathsf{NP}$ over m-bit inputs and $\delta > 0$ such that $L \notin \mathsf{Formula}[2^{m^\delta}]$.
- Suppose there exists $k \geq 1$ such that for every $d \geq 1$ there is $\varepsilon_d > 0$ such that $\mathsf{MCSP}[(s,0),(s,\delta)] \notin \mathsf{AC}^0_d[N^{1+\varepsilon_d}]$, where $s(n) = n^k$ and $\delta(n) = n^{-k}$. Then $\mathsf{NP} \not\subseteq \mathsf{NC}^1$.
- Let $k(n) = n^{o(1)}$. If there exists $\varepsilon > 0$ such that k-Vertex-Cover $\notin \mathsf{DTISP}[m^{1+\varepsilon}, m^{o(1)}]$, where the input is an n-vertex graph represented by an adjacency matrix of bit length $m = \Theta(n^2)$, then $\mathsf{P} \neq \mathsf{NP}$.
- Let $k(n) = (\log n)^C$, where $C \in \mathbb{N}$ is arbitrary. If for every $d \geq 1$ there exists $\varepsilon > 0$ such that k-Vertex-Cover $\notin \mathsf{AC}^0_d[m^{1+\varepsilon}]$, then $\mathsf{NP} \not\subseteq \mathsf{NC}^1$.

Oliveira-Pich-Santhanam [39] and McKay-Murray-Williams [36] (Informal). If there exists $\varepsilon > 0$ such that for every small enough $\beta > 0$,

- $\mathsf{MCSP}[2^{\beta n}] \notin \mathsf{Circuit}[N^{1+\varepsilon}]$, then $\mathsf{NP} \not\subseteq \mathsf{Circuit}[\mathrm{poly}]$.
- $\mathsf{MKtP}[2^{\beta n}] \notin \mathsf{TC}^0[N^{1+\varepsilon}]$, then $\mathsf{EXP} \not\subseteq \mathsf{TC}^0[\mathrm{poly}]$.
- $\mathsf{MKtP}[2^{\beta n}] \notin U_2\text{-}\mathsf{Formula}[N^{3+\varepsilon}]$, then $\mathsf{EXP} \not\subseteq \mathsf{Formula}[\mathrm{poly}]$.
- $\mathsf{MKtP}[2^{\beta n}] \notin B_2\text{-}\mathsf{Formula}[N^{2+\varepsilon}]$, then $\mathsf{EXP} \not\subseteq \mathsf{Formula}[\mathrm{poly}]$.
- $\mathsf{MKtP}[2^{\beta n}] \notin \mathsf{Formula\text{-}XOR}[N^{1+\varepsilon}]$, then $\mathsf{EXP} \not\subseteq \mathsf{Formula}[\mathrm{poly}]$.
- $\mathsf{MKtP}[2^{\beta n}] \notin \mathsf{BP}[N^{2+\varepsilon}]$, then $\mathsf{EXP} \not\subseteq \mathsf{BP}[\mathrm{poly}]$.
- $\mathsf{MKtP}[2^{\beta n}] \notin (\mathsf{AC}^0[6])[N^{1+\varepsilon}]$, then $\mathsf{EXP} \not\subseteq \mathsf{AC}^0[6]$.

Many results for MKtP admit analogues for MrKtP, which considers a randomized version of Kt complexity introduced by [38]. An advantage of MrKtP is that strong unconditional lower bounds against uniform computations are known, while the hardness of problems such as MCSP and MKtP currently relies on cryptographic assumptions.

Chen-McKay-Murray-Williams [12]. The following results hold.

- If there is $\varepsilon > 0$, $c \geq 1$, and an n^c-sparse language $L \in \mathsf{NP}$ such that $L \notin \mathsf{Circuit}[n^{1+\varepsilon}]$, then $\mathsf{NE} \not\subseteq \mathsf{Circuit}[2^{\delta \cdot n}]$ for some $\delta > 0$.
- If there is $\varepsilon > 0$ such that for every $\beta > 0$ there is a 2^{n^β}-sparse language $L \in \mathsf{NTIME}[2^{n^\beta}]$ such that $L \notin \mathsf{Circuit}[n^{1+\varepsilon}]$, then $\mathsf{NEXP} \not\subseteq \mathsf{Circuit}[\mathrm{poly}]$.

More recently, [10] established that many hardness magnification theorems for problems such as MCSP and MKtP hold in fact under the assumption that a *sufficiently sparse and explicit language* admits weak lower bounds. We refer to their work for more details.

A.2 Hardness Magnification Through the Lens of Oracle Circuits

We can view the results from Appendix A.1 as *unconditional* upper bounds on the size of small fan-in oracle circuits solving the corresponding problems, for a certain choice of oracle gates. In a magnification theorem, it is important to upper bound the uniform complexity of the oracle gates. For our discussion, this is not going to be relevant.

We repeat here a definition from Section 2, for convenience of the reader.

▶ **Definition 62** (Local circuit classes). *Let $\mathcal{C}$ be a circuit class (such as $\mathsf{AC}^0[s]$, $\mathsf{TC}^0_d[s]$, $\mathsf{Circuit}[s]$, etc). For functions $q, \ell, a \colon \mathbb{N} \to \mathbb{N}$, we say that a language L is in $[q, \ell, a]$–$\mathcal{C}$ if there exists a sequence $\{E_n\}$ of oracle circuits for which the following holds:*

- **(i)** *Each oracle circuit E_n is a circuit from $\mathcal{C}$.*
- **(ii)** *There are at most $q(n)$ oracle gates in E_n, each of fan-in at most $\ell(n)$, and any path from an input gate to an output gate encounters at most $a(n)$ oracle gates.*
- **(iii)** *There exists a language $\mathcal{O} \subseteq \{0,1\}^*$ such that the sequence $\{E^{\mathcal{O}}_n\}$ (E_n with its oracle gates set to $\mathcal{O}$) computes L.*

In the definition above, q stands for *quantity*, ℓ for *locality*, and a for *adaptivity* of the corresponding oracle gates.

The fact that existing magnification theorems produce such circuits is a consequence of the algorithmic nature of the underlying proofs, which show how to reduce an instance of a problem to shorter instances of another related problem. By inspection of each proof, it is possible to establish a variety of upper bounds. We explicitly state some of them below.

▶ **Proposition 63.** *The following results hold.*

- [3] *For every $\Pi \in \{\mathsf{BFE}, \mathsf{W}_{\mathsf{S}_5}, \text{W5-STCONN}\}$ and every $\beta > 0$, $\Pi_n \in \left[O\left(n^{1-\beta}\right), n^{\beta}, O\left(\frac{1}{\beta}\right)\right]$–$\mathsf{TC}^0[O(n)]$.*
- [35] *For every $\delta > 0$, $\mathsf{CircEval}_n \in \left[n \cdot \mathsf{poly}(\log n), n^{\delta}, n^{1-\delta}\right]$–$\mathsf{Circuit}[n \cdot \mathsf{poly}(\log n)]$.*
- [42] *For every constructive function $n \leq s(n) \leq 2^n/\mathsf{poly}(n)$ and parameter $0 < \delta(n) < 1/2$, $\mathsf{MCSP}[(s,0),(s,\delta)] \in [N, \mathsf{poly}(s/\delta), 1]$–$\mathsf{Formula}[N \cdot \mathsf{poly}(s/\delta)]$.*
- [42] *Let $k = (\log n)^C$, where $C \in \mathbb{N}$. Then k-Vertex-Cover $\in [1, (\log n)^{4C}, 1]$–$\mathsf{AC}^0_d[m^{1+\varepsilon}]$, where $\varepsilon_d \to 0$ as $d \to \infty$.*
- [39] *For every $\beta > 0$ and for every constructive function $s(n) \leq 2^{\beta n}$, Gap-MKtP $\in [N, \mathsf{poly}(s), 1]$–$\mathsf{Formula\text{-}XOR}[N \cdot \mathsf{poly}(s)]$.*
- [39] *For every constructive function $s(n) \leq 2^n/\mathsf{poly}(n)$, it follows that Gap-MCSP $\in [N \cdot \mathsf{poly}(s), \mathsf{poly}(s), \mathsf{poly}(s)]$–$\mathsf{Circuit}[N \cdot \mathsf{poly}(s)]$.*
- [36] *For every constructive function $s(n) \leq 2^n/\mathsf{poly}(n)$, we have $\mathit{MCSP}[s(n)] \in [O(N/\mathsf{poly}(s)), \mathsf{poly}(s), O(n/\log(s))]$–$\mathsf{Circuit}[N/\mathsf{poly}(s)]$.*

We stress however that not every hardness magnification theorem needs to lead to an unconditional construction of efficient oracle circuits. (All the proofs that we know of produce such circuits though.)

Separating Two-Round Secure Computation from Oblivious Transfer

Benny Applebaum
Tel-Aviv University, Israel
bennyap@post.tau.ac.il

Zvika Brakerski
Weizmann Institute of Science, Rehovot, Israel
zvika.brakerski@weizmann.ac.il

Sanjam Garg
UC Berkeley, CA, USA
sanjamg@berkeley.edu

Yuval Ishai
Technion – Israel Institute of Technology, Haifa, Israel
yuvali@cs.technion.ac.il

Akshayaram Srinivasan
UC Berkeley, CA, USA
akshayaram@berkeley.edu

Abstract

We consider the question of minimizing the round complexity of protocols for secure multiparty computation (MPC) with security against an arbitrary number of semi-honest parties. Very recently, Garg and Srinivasan (Eurocrypt 2018) and Benhamouda and Lin (Eurocrypt 2018) constructed such 2-round MPC protocols from minimal assumptions. This was done by showing *a round preserving reduction* to the task of secure *2-party* computation of the oblivious transfer functionality (OT). These constructions made a novel non-black-box use of the underlying OT protocol. The question remained whether this can be done by only making black-box use of 2-round OT. This is of theoretical and potentially also practical value as black-box use of primitives tends to lead to more efficient constructions.

Our main result proves that such a black-box construction is impossible, namely that non-black-box use of OT is necessary. As a corollary, a similar separation holds when starting with any 2-party functionality other than OT.

As a secondary contribution, we prove several additional results that further clarify the landscape of black-box MPC with minimal interaction. In particular, we complement the separation from 2-party functionalities by presenting a complete 4-party functionality, give evidence for the difficulty of ruling out a complete 3-party functionality and for the difficulty of ruling out black-box constructions of 3-round MPC from 2-round OT, and separate a relaxed "non-compact" variant of 2-party *homomorphic secret sharing* from 2-round OT.

2012 ACM Subject Classification Theory of computation → Computational complexity and cryptography; Theory of computation → Cryptographic protocols

Keywords and phrases Oracle Separation, Oblivious Transfer, Secure Multiparty Computation

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.71

Funding *Benny Applebaum*: Supported by the European Union's Horizon 2020 Programme (ERC-StG-2014-2020) under grant agreement no. 639813 ERC-CLC, and the Check Point Institute for Information Security.
Zvika Brakerski: Supported by the Binational Science Foundation (Grant No. 2016726), and by the European Union Horizon 2020 Research and Innovation Program via ERC Project REACT (Grant 756482) and via Project PROMETHEUS (Grant 780701).
Sanjam Garg: Supported in part from AFOSR Award FA9550-19-1-0200, AFOSR YIP Award, NSF CNS Award 1936826, DARPA and SPAWAR under contract N66001-15-C-4065, a Hellman Award

and research grants by the Okawa Foundation, Visa Inc., and Center for Long-Term Cybersecurity (CLTC, UC Berkeley). The views expressed are those of the authors and do not reflect the official policy or position of the funding agencies.
Yuval Ishai: Supported by ERC Project NTSC (742754), NSF-BSF grant 2015782, BSF grant 2018393, and a grant from the Ministry of Science and Technology, Israel and Department of Science and Technology, Government of India.
Akshayaram Srinivasan: Supported in part from AFOSR Award FA9550-19-1-0200, AFOSR YIP Award, NSF CNS Award 1936826, DARPA and SPAWAR under contract N66001-15-C-4065, a Hellman Award and research grants by the Okawa Foundation, Visa Inc., and Center for Long-Term Cybersecurity (CLTC, UC Berkeley). The views expressed are those of the authors and do not reflect the official policy or position of the funding agencies.

Acknowledgements We thank Iftach Haitner, Mohammad Mahmoody, and Rotem Tsabary for helpful discussions.

1 Introduction

Secure multiparty computation (MPC) allows mutually distrusting parties to compute a joint function f of their private inputs without revealing anything more than the output to each other.

In this paper we consider the simplest setting for MPC with *no honest majority*, namely MPC with an arbitrary number of corrupted parties. We focus on the *semi-honest* (aka passive) security model, where corrupted parties follow the protocol but try to (jointly) learn additional information on inputs of uncorrupted parties from the messages they observe. We also assume that the parties can communicate over *secure point-to-point channels* and that corruptions are *non-adaptive* (i.e., the set of corrupted parties is fixed before the protocol's execution). All of the above assumptions make negative results stronger.

The design and analysis of MPC protocols crucially rely on the notion of *secure reductions*. In particular, classical completeness results [54, 34] have shown that the problem of securely computing a general n-party functionality f efficiently reduces to the problem of securely computing the elementary finite 2-party *Oblivious Transfer* (OT) functionality [51, 26]. (Similar results have been proven for active adversaries as well [44, 43].) Perhaps surprisingly, for 2-party secure computation (2PC), Yao's reduction is *round preserving*. That is, it incurs no overhead in the round complexity. It additionally requires the parties to make a black-box use of any pseudorandom generator (PRG).

▶ **Theorem 1** (Round-optimal 2PC [54]). *Every 2-party functionality g admits an MPC protocol that only makes parallel calls to an OT oracle and a black-box use of a PRG.*

In more detail, the OT functionality F_{OT} involves two parties referred to as *Receiver* and *Sender*. The functionality takes a bit x from the Receiver and a pair of bits (more generally, strings) (m_0, m_1) from the Sender, and delivers to the Receiver the message m_x. This is done while hiding m_{1-x} from the Receiver and hiding x from the Sender.

Yao's reduction makes a single round of parallel calls to F_{OT}.[1] Using a suitable composition theorem for MPC in the semi-honest model (see, e.g., [20, 33]), this can be securely replaced by parallel invocations of any *OT protocol*, namely a secure 2-party protocol for F_{OT}. The

[1] If both parties should receive an output, the reduction uses parallel OTs in both directions, where each party acts both as a Sender and as a Receiver.

resulting construction of 2-party MPC from a 2-party OT protocol is *black-box*.[2] This means that the MPC protocol does not depend on the code of the underlying OT protocol, and moreover the security proof is black-box in the sense that any adversary "breaking" the MPC protocol can be used as a black-box to break the OT protocol. Instantiating with one of several natural known 2-round OT protocols (whose existence follows from standard intractability assumptions), we get a 2-round 2-party MPC protocol, which is clearly optimal.

Round Complexity in the Multiparty Setting. In contrast to the 2-party setting, progress on the round complexity of general MPC has been slow and some of the questions still remain unanswered. As already mentioned, the completeness of OT in the multiparty setting was first established by Goldreich, Micali, and Wigderson (GMW) [34]. However, their reduction suffered from large round complexity (proportional to the circuit depth of the target function). The question of achieving a constant-round protocol has been considered by Beaver, Micali, and Rogaway [12], who extended Yao's garbled circuit technique to the multiparty setting. Combined with the GMW result, this yields a reduction to OT with constant overhead in the round complexity.

▶ **Theorem 2** (Constant-round MPC [34, 12]). *Every n-party functionality admits a constant-round protocol, making black-box use of a PRG, given an oracle access to F_{OT}.*

In more concrete terms, the most round-efficient current MPC protocol that makes a black-box use of a 2-round OT protocol requires 4 rounds of interaction [1]. The above results left a gap between the round complexity of 2PC and MPC protocols. In a recent breakthrough, this gap was partially closed.

▶ **Theorem 3** (2-round MPC from minimal assumption [31, 13]). *Suppose a 2-round OT protocol exists. Then every n-party functionality admits a 2-round MPC protocol.*

The theorem settles the high-order bit about the minimal assumptions needed for 2-round MPC by showing that 2-round OT is sufficient. (Being a special case of 2-round MPC, it is clearly necessary.) However, quite surprisingly, the MPC protocol in these works inherently makes use of the code of the underlying OT protocol. This situation is quite rare in the context of MPC protocols and in cryptography in general (see Section 1.2), and it is not clear whether this non-black-box use of OT is inherent. This calls for the following natural question:

> Is it possible to reduce general n-party MPC to a 2-party OT protocol in a round-preserving black-box way? In particular, is there a black-box construction of 2-round MPC from 2-round OT?

The above question is not only of a theoretical interest, but is also potentially relevant to practice. Indeed, black-box use of cryptographic primitives tends to lead to more efficient constructions. The goal of obtaining efficient 2-round MPC protocols is very well motivated, since such protocols have *qualitative* advantages over similar protocols with a bigger number of rounds. Indeed, in a 2-round MPC protocol, each party can send its first-round messages and then go offline until all second-round messages are received and the output can be computed. Moreover, the first-round messages can be potentially reused for several computations in which a party's input remains unchanged. This is analogous to the qualitative advantage of public-key encryption over interactive key agreement.

[2] The notion of a black-box construction used in this paper (also referred to as a black-box reduction) corresponds to the notion of a *fully black-box reduction* in the taxonomy of [52].

In this paper, we will provide a negative answer to the above question, showing that there is a real gap between the power of round-preserving black-box (RPBB) reductions and round-preserving non-black-box reductions. Our findings also reveal a rich and somewhat unexplored world of cryptographic protocols that use a minimal amount of interaction. We will exhibit some of the black-box and non-black-box connections among these primitives and relate them to standard ones.

1.1 Our Results

We now give a more detailed account of our results.

1.1.1 Separating 3-Party Functionalities from 2-Party Functionalities

Our first result shows that 2-round MPC protocols cannot be based on 2-round OT in a black-box way. In fact, we show that even 3-party computation of fairly simple functionalities is unachievable via black-box use of 2-round OT.

▶ **Theorem 4** (Main Result). *There exists a 3-party functionality f that cannot be securely computed by a 2-round protocol with black-box use of 2-round OT.*

We note that we do not just rule out round-preserving reductions to the ideal-OT functionality, but rather rule out all such black-box constructions from an *OT protocol*. (Indeed, much of the technical work is devoted to coping with the latter model; see Section 2.) Moreover, the theorem holds even for constructions in the private-channel setting (where each pair of parties is connected via a private channel), and even when the parties have an access to a public common reference string (CRS), and to a random oracle.

1.1.2 A Complete 4-Party Functionality

OT turns out to be incomplete for MPC under RPBB reductions. Given this state of affairs, one may try to prove a completeness result for some other finite functionality. We show that this is indeed possible. Specifically, let $(3,4)-\mathsf{MULTPlus}$ denote the 4-party functionality that takes a pair of bits (x_i, z_i) from each of the first three parties (and no input from the fourth party) and delivers the value $x_1x_2x_3+z_1+z_2+z_3$ to all four parties where addition and multiplication are over the binary field. We prove that $(3,4)-\mathsf{MULTPlus}$ is MPC-complete under RPBB reductions. (Related results have been proved in other settings [18, 28].) In fact, we prove an "ideal-oracle" completeness result just like in Yao's theorem.

▶ **Theorem 5** (Round Optimal MPC from Finite Ideal Functionality). *Every n-party functionality f can be realized using parallel calls to a $(3,4)-\mathsf{MULTPlus}$ oracle and a black-box use of a PRG.*

It's worth noting that $(3,4)-\mathsf{MULTPlus}$ is related to the standard 2-party OT functionality. In general, for $d \le p$, let $(d,p)-\mathsf{MULTPlus}$ denote the p-party functionality in which each of the first d parties holds an input (x_i, z_i) and the product-sum $\prod_i x_i + \sum_i z_i$ is delivered to all p parties. Then, standard 2-party OT is equivalent (under RPBB reductions) to $(2,2)-\mathsf{MULTPlus}$.[3]

[3] First observe that the receiver's output in OT can be written as $m_0 + x(m_1 - m_0)$ where addition and multiplication are over the binary field. Therefore, we can implement OT based on $(2,2)-\mathsf{MULTPlus}$ by letting the receiver (resp., sender) play the role of the first party (resp., second party) with inputs $x_1 = x$

1.1.3 The Land of Three-Party Functionalities

The finite $(3,4)-\mathsf{MULTPlus}$ therefore stands at the entry point to the general MPC mainland. Across the ocean, lies the island of two party functionalities (including the complete OT) and one cannot cross it in a black-box round-preserving vessel. We move on and explore the mysterious land of three party functionalities.

Given the incompleteness of 2-party functionalities and the completeness of four party functionalities (under round-preserving BB reductions), it is natural to ask whether 3-party functionalities are complete. We show that the answer to this question is related to a well-known open problem in information-theoretic cryptography.

▶ **Question 6** ([41]). *Does every finite function admit a degree-2 statistical randomized encoding?*

A randomized encoding (RE) of a function [41, 6] $f(x)$ is a randomized function $\hat{f}(x;r)$ that, in addition to the input x, takes an additional random input r. For any input x, the random variable $\hat{f}(x)$, induced by a random choice of r, should reveal the value of $f(x)$ and hide everything else. The power of REs stems from the fact that even complicated functions can be encoded by simple encoding. In the context of MPC, it is known that every finite function can be encoded by a function $\hat{f}(x;r)$ that each of its outputs can be written as a degree-3 polynomial over the indeterminates (x,r). While some negative results are known for perfectly-private degree-2 encodings [41], the feasibility of statistically-private degree-2 encodings (that are allowed to have a small non-zero privacy error) has remained open for almost 20 years. (See also the surveys [40, 3].) We relate this longstanding open problem to the completeness of 3-party functionalities under RPBB reductions.

▶ **Proposition 7.** *A positive answer to Question 6 implies that every n-party finite functionality g can be realized using parallel calls to an oracle that implements the 3-party functionality* $(2,3)-\mathsf{MULTPlus}$.

The proposition implies that we cannot rule out the completeness of 3-party functionalities without ruling out the existence of general degree-2 (statistical) randomized encoding. Similar barriers have been established in the context of degree-2 cryptographic hash functions [5]. We note that the completeness of $(2,3)-\mathsf{MULTPlus}$ follows even from the existence of general degree-2 fully-secure *multiparty randomized encoding* [4] – a seemingly weaker variant of RE whose existence is also open. (See the discussion in [4].)

External output functionalities. The $(2,3)-\mathsf{MULTPlus}$ is a special case of an *external-output* 3-party functionality. Formally, let $g(x,y)$ be a 2-party functionality. The *external version* of g, is the 3-party functionality f_g that takes x from Alice, y from Bob and delivers $g(x,y)$ to Alice and Bob, and to Carol who holds no input.[4] Two-round protocols for

and a random bit z_1 (resp., $x_2 = m_1 - m_0$ and $z_2 = m_0$). The receiver gets the required output (by subtracting z_1 from the output of the $(2,2)-\mathsf{MULTPlus}$), and the sender learns noting (since it receives a random bit). In the other direction, first observe that the one-sided variant of $(2,2)-\mathsf{MULTPlus}$, where only the first party has to learn the value $x_1x_2 + z_1 + z_2$, is RPBB-reducible to OT. Indeed, let player 1 (resp., player 2) play the role of the receiver (resp., sender) with inputs $x = x_1$ (resp., $m_0 = z_2$ and $m_1 = z_2 + x_2$), and set the output of player 1 to be the output of the OT plus z_1. Next, observe that standard $(2,2)-\mathsf{MULTPlus}$ can be constructed by making two parallel calls to the one-sided variant.

[4] In fact, for all of our purposes, an even weaker version suffices. In this relaxed version, all parties are allowed to learn the output (for purposes of privacy), but only Carol is *required* to learn it (for purposes of correctness). Since this leads to a cumbersome definition, we stick to the simpler version described above.

such functionalities turn out to have interesting properties. Specifically, at the core of our main impossibility result (Theorem 4), lies the following constructive theorem for external functionalities.

▶ **Theorem 8** (Conversion Theorem). *Let $g(x, y)$ be a 2-party functionality. The external version of g is the 3-party functionality f_g that takes x from Alice, y from Bob and delivers $g(x, y)$ to Alice and Bob, and to Carol who holds no input.*

Suppose that the functionality f_g can be securely computed in 2 rounds by making a black-box use of 2-round OT over private channels. Then, f_g can be securely implemented over random inputs given only an access to a Random Oracle over private channels. Moreover, in the resulting protocol Carol sends no message and so it yields a two-party protocol for computing g over random inputs given only an access to a Random Oracle.

Haitner et al. [36] showed that any 2-party functionality that can be securely realized in the Random Oracle (RO) model is *trivial* in the sense that it admits an unconditional 2-party protocol over random inputs with security against computationally-unbounded adversaries. As a corollary, we derive the following stronger version of Theorem 4.

▶ **Corollary 9.** *Every external functionality f_g that is based on a non-trivial 2-party functionality f cannot be computed by a 2-round protocol that makes a black-box use of 2-round OT even in the private-channel setting.*

A notable example for such a non-trivial 2-party functionality is the AND functionality [21, 46].

Corollary 9 is tight in terms of round complexity. With one additional round, f_g can be black-box reduced to 2-round OT. (Specifically, one can use Yao's theorem to pass the value of $f(x, y)$ to Alice and Bob in two rounds, and then exploit the additional round to send this value to Carol.) The 2-party completeness of OT (Theorem 1) also implies that Corollary 9 holds when the OT functionality is replaced by an arbitrary 2-party functionality $h(x, y)$. Overall, we get a separation between all 2-party functionalities and all external functionalities f_g whose underlying g is non-trivial.

Relation with homomorphic secret sharing. Two-round MPC for extended-output functionalities can be seen as closely related to the problem of homomorphic secret sharing (HSS) [16, 18]. HSS is the secret-sharing analogue of fully homomorphic encryption. A (2-party) HSS scheme allows local computation of a function $g(x, y)$ on independently shared inputs x and y, where the output $g(x, y)$ can be decoded from the pair of output shares. The standard notions of HSS require either *additive* decoding over a group or, more generally, that the output shares be *compact* in the sense that they are shorter than the inputs. A natural variant is to replace compactness by the requirement that the pair of output shares give no information except $g(x, y)$, even from the point of view of one of the input holders. This security requirement is easily obtained from additive HSS via a simple additive refreshing of the output shares. This flavor of non-compact HSS easily implies (in a black-box way) a 2-round external output protocol for g (in the private-channel setting), which by Corollary 9 can be separated from 2-round OT. On the other hand, a non-black-box construction of non-compact HSS from 2-round OT follows from [31, 28].

1.2 Discussion

In this section we give some further perspective on our results and some future research directions which they motivate.

1.2.1 Why is the multiparty setting different than the 2-party setting?

It is instructive to reconsider the round complexity of MPC in light of our results. Protocols with low round complexity are based on two types of reductions.

1. A *degree reduction* that takes a general n-party functionality f and reduces it (via RPBB reduction) to a degree-d functionality for a constant d. Specifically, the standard machinery of randomized-encoding leads to degree 3. In the special case of two parties, we can trivially reduce the degree down to 2, and so we get a degree reduction to $d = \min(3, n)$.
2. A *player reduction* that takes an n-party functionality of degree d and reduces it to the (d, p)–MULTPlus functionality. We show that p can be dropped down to $d + 1$ and, in any case, it is no larger than n, leading to an expression of the form $p = \min(d + 1, n)$.

In the special case of two parties $n = 2$ we get an RPBB-reduction to $(2, 2)$–MULTPlus which is equivalent to OT. For large n's, this leads to the completeness of $(3, 4)$–MULTPlus (Theorem 5). In order to prove that OT is complete (under RPBB-reductions) we have to bypass two barriers: A *Degree barrier* (prove completeness of degree 2 functionalities) and a *Player reduction barrier* (reducing the 3-party functionality (2,3)–MULTPlus to (2,2)–MULTPlus). While the first barrier is well-known, the second one appears to be new to this work. Clearly, both barriers are bypassed by non-black-box techniques (Theorem 3). We show that this is inherent for the second "player reduction" barrier, and leave the possibility of breaking the degree-barrier via RPBB-reduction open.

1.2.2 On the Role of Non-Black-Box Constructions in Cryptography

Our main result provides a very natural example of a pair of cryptographic primitives for which a non-black-box construction of one from the other exists but a black-box construction can be ruled out. Thus, our work further demonstrates the essential role of non-black-box techniques in cryptography.

To give some historical perspective, following the seminal result of Impagliazzo and Rudich [39] and subsequent works on black-box separations in cryptography [53, 32, 52], the question of finding a pair of "natural" cryptographic primitives for which a non-black-box reduction is provably necessary has been put forward as a desirable but elusive goal.[5] For some of the conjectured candidate examples, such as constructing "malicious OT" from "semi-honest OT," black-box constructions were subsequently found [35]. However, in recent years several such provable examples emerged. We survey some of the most notable ones below.

- *Non-interactive commitments from OWFs*: Mahmoody and Pass [48] showed that non-interactive commitments cannot be constructed from so-called "hitting-OWFs" in a black-box manner, even though a non-black-box construction was previously shown [10]. One nice feature of this example is that a non-interactive commitment is a very basic primitive. However, in comparison hitting-OWFs have found little other applications in cryptography. Furthermore, the separation here is intuitively weak since knowing the *circuit size* of the OWF enables a black-box construction. This is contrasted with the non-black-box constructions of 2-round MPC from OT [31, 13], which make an essential use of the full code of the OT protocol.

[5] The question is informal due to the subjective nature of the term "natural primitive." It should not be confused with the question of black-box vs. non-black-box *simulation*, for which Barak's breakthrough non-black-box simulation technique [8] gave the first such natural examples.

- *Two-round OT extension*: Beaver gave a construction of two-round OT extension [11] making a non-black-box use of one-way functions. This construction can be cast in the OT-hybrid model. However, very recently, Garg et al. [29] showed that a back-box variant of such a constriction is impossible. They showed that such constructions are not possible even when black-box use of a random oracle (and not just a one-way function) is allowed. One limitation of this example is that the separation is only proved for protocols in the OT-hybrid model.
- *IBE from CDH (or Generic Groups)*: In a recent result, Döttling and Garg [24] show that Identity-Based Encryption (IBE) can be realized under the Computational Diffie-Hellman (CDH) assumption, while black-box constructions of the same had been previously ruled out [15, 50]. However, in this case both the positive and the negative result use strong "structured" primitives.
  In another very related example, Döttling and Garg [23] showed a generic non-black-box construction of hierarchical-IBE from IBE but we can expect a black-box impossibility for the same using techniques from [15].[6]
- *Constructions of IO*: In a very recent work, Garg et al. [30] showed that indistinguishability obfuscation (IO) [27] cannot be constructed from compact functional encryption (FE) in a black-box manner, even though non-black-constructions achieving this were already known [14, 2].
- *Secret-Key FE vs Public-Key FE*: In a recent work, Kitagawa et al. [45] showed that public-key FE can be constructed from secret-key FE in a non-black-box manner, even though black-box positive constructions had been previously ruled out [7].

In comparison with the above works, our main result has the advantage that it considers two very natural and simple primitives. Our separation lives entirely in the "passive adversary" world, and does not depend on the input domain being super-polynomial. For instance, our separation is also meaningful for MPC with a uniform input distribution over a constant-size domain. Thus, it is arguably similar in spirit to the Impagliazzo-Rudich separation of key agreement from one-way functions [39], except that in the latter case no analogous non-black-box construction is known.

1.3 Open Problems

While we settle the main open question concerning black-box round-optimal MPC, our work leaves several interesting directions for future research. We highlight a few below, focusing on our current setting of semi-honest security with no honest majority.

- **3-round MPC from black-box OT.** Our main result rules out 2-round MPC protocols making a black-box use of 2-round OT. On the other hand, a previous result of Ananth et al. [1] shows that such 4-round protocols exist. What about 3-round protocols? In the full version of the paper, we give evidence that extending our negative result to the 3-round case would require settling Question 6 in the negative. This barrier does not seem to apply to 3-round protocols in which the first round messages do not depend on the inputs, or alternatively 2-round protocols with a public-key infrastructure (PKI) setup.
- **Black-box use of stronger primitives.** Can our negative result be bypassed by replacing OT with stronger or more structured primitives? It is known that 2-round MPC can make black-box use of different flavors of multi-key homomorphic encryption [49, 22]

[6] Even though we expect such an impossibility to hold, we are not aware of a work that gives a full proof of this claim.

or homomorphic secret sharing [17, 18]. However, this is almost immediate from the definitions of such primitives. Using simpler structured primitives, such as a "DDH-hard" group or a generic group, we have black-box 2-round protocols that require a PKI setup [28]. Can we get similar group-based constructions in the plain model? Alternatively, can the separation be bypassed by using stronger variants of 2-round OT, such as OT with high information rate [25] or OT with a stronger notion of receiver privacy [42]?

- **Minimal complete primitive for 2-round MPC.** We have shown the existence of a 4-party functionality such that general MPC reduces to *parallel* calls to this functionality without further interaction. We have also ruled out such a 2-party functionality. This leaves open the 3-party case. As in the case of 3-round MPC from 2-round OT, we can show that proving a *negative* result would require settling Question 6 in the negative.
- **Standard MPC vs. client-server MPC.** Our main negative result automatically carries over to the stronger *client-server model* for MPC, where n clients interact with n servers who have no inputs or outputs. It is known that 2-round client-server MPC can be constructed in a non-black-box way from standard 2-round MPC [28]. Whether such a black-box construction exists remains open.

2 Technical Overview

In this section, we give a high-level overview of our techniques in proving the main result (Theorem 4). To keep the exposition simple, we restrict ourselves to proving the impossibility result for securely computing external-AND.

External-AND Functionality. Let us denote the three parties by (P_1, P_2, P_3). The private input of P_1 is a bit x, the private input of P_2 is a bit y and P_3 does not have any private inputs. The functionality $f_\times$ outputs $x \cdot y$ to all the parties. Specifically, $f_\times(x, y, \perp) = x \cdot y$.

Main Idea. To prove the impossibility result, we define a set of oracles such that 2-round oblivious transfer exists with respect to these oracles, but there exists no 2-round, semi-honest protocol for securely computing $f_\times$. This is sufficient to rule out a black-box transformation from 2-round oblivious transfer to 2-round, 3-party semi-honest protocols for general functionalities. Below, we describe these oracles (throughout this overview, sec denotes the security parameter):

- OT_1 is a random length tripling function that takes in the receiver's choice bit $b \in \{0,1\}$ and its random tape $r \in \{0,1\}^{\text{sec}}$ and outputs the receiver's message otm_1.
- OT_2 is a random length tripling function that takes in the receiver's message otm_1, the sender's inputs $m_0, m_1 \in \{0,1\}$, its random tape $s \in \{0,1\}^{\text{sec}}$ and outputs the sender's message otm_2.
- OT_3 is a function that takes the transcript $(\mathsf{otm}_1, \mathsf{otm}_2)$ along with (b, r) as input and outputs m_b if there exists unique (m_0, m_1, s) for which $\mathsf{OT}_1(b, r) = \mathsf{otm}_1$ and $\mathsf{OT}_2(\mathsf{otm}_1, m_0, m_1, s) = \mathsf{otm}_2$. Otherwise, it outputs $\perp$.

As observed by [37], the oracles $(\mathsf{OT}_1, \mathsf{OT}_2, \mathsf{OT}_3)$ naturally give rise to a 2-round oblivious transfer protocol. Specifically, letting b, r denote the input/randomness of the receiver, and letting $(m_0, m_1), s$ denote the input/randomness of the sender, the protocol proceeds as follows: The receiver sends $\mathsf{otm}_1 = \mathsf{OT}_1(b, r)$ to the sender, who responds with $\mathsf{otm}_2 = \mathsf{OT}_2(\mathsf{otm}_1, m_0, m_1, s)$, allowing the receiver to output the value $\mathsf{OT}_3(\mathsf{otm}_1, \mathsf{otm}_2, b, r)$.

In this work, we prove that the existence of a 2-round protocol for external-AND w.r.t. $(\mathsf{OT}_1, \mathsf{OT}_2, \mathsf{OT}_3)$ implies a *two-party* protocol for computing $g(x, y) = x \cdot y$ in the random oracle model. (Note that we start with a three-party protocol for an external functionality,

and show a two-party protocol for a related functionality.) The existence of such two-party protocol is known to be impossible [21, 46, 36, 47] and therefore the original protocol can also not exist. This proves Theorem 8 discussed above, and implies Theorem 4 as a corollary.

Outline. The above result is proven using a sequence of transformations depicted in Figure 1.

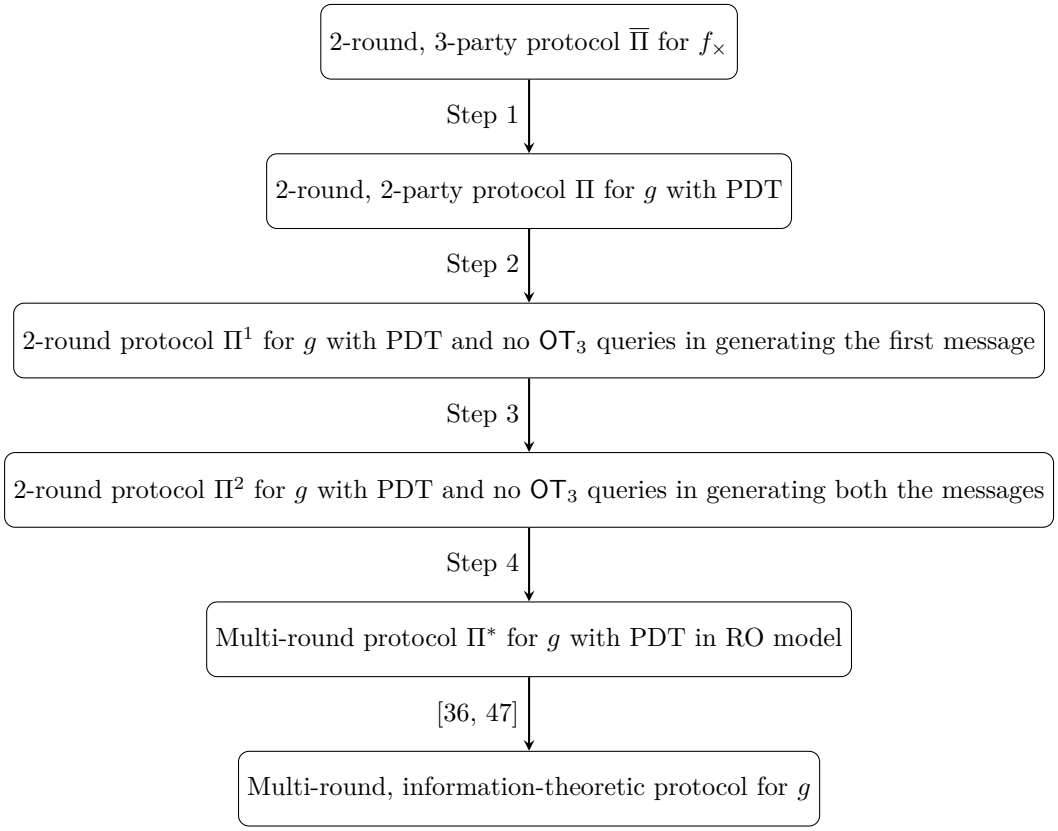


Figure 1 Key Steps in the Proof. Here, PDT denotes publicly decodable transcript, $g(x, y) = x \cdot y$ and $f_\times(x, y, \perp) = g(x, y)$.

Step-1: Publicly Decodable Transcript. Let $\overline{\Pi}$ be a 2-round protocol for securely computing $f_\times$ w.r.t. $(\mathsf{OT}_1, \mathsf{OT}_2, \mathsf{OT}_3)$. We first show that this implies a 2-round, 2-party protocol Π for computing the two-party functionality $g = g(x, y) = x \cdot y$, which has an additional special property – the output is *publicly decodable* from the transcript. More formally, there exists a deterministic algorithm Dec that computes the output of the functionality given the transcript of the two-party protocol. In particular, if there exists a protocol Π that computes g with publicly decodable transcript, then Dec on input $\mathbb{T}$ (which is the transcript of the protocol Π) outputs $g(x, y)$. In terms of security, Π is required to have the standard security properties of a two-party (semi-honest) protocol, i.e., the corrupted party does not learn any information about the other party's input except the output.

To transform a 2-round protocol for $f_\times$ into a 2-round protocol Π for g with publicly decodable transcript, we use a player emulation technique.[7] Concretely, we ask P_1 to choose a uniform random tape for P_3 and send this random tape in the first round. Using this random

[7] The idea of player emulation goes back to [19]; see also [38].

tape, P_1 and P_2 can generate the messages P_3 would have sent in the original protocol. Additionally, P_1 and P_2 forwards all its outgoing messages that are sent to each other as well the messages sent to P_3 in the original protocol.

This protocol satisfies public decodability since given the transcript of the protocol (which includes the entire view of P_3 in the original protocol), one can run the output computing algorithm of P_3 to learn $g(x, y)$. Further, the security follows directly from the security of the original protocol when (P_1, P_3) and (P_2, P_3) are corrupted.[8]

Remaining Steps – Removing OT_3 Queries. In the remainder of the proof, we show that use of the oracle OT_3 can be removed. More specifically, we show how to convert any two-round two-party secure computation protocol Π with access to $(\mathsf{OT}_1, \mathsf{OT}_2, \mathsf{OT}_3)$ and publicly decodable transcript into a two-party protocol that computes the same functionality, but with a few differences. The oracles OT_3 will no longer be used in the new protocol, but this will come at a cost, both in round-complexity and in security:

- The round complexity of the protocol will grow by a polynomial factor (essentially upper bounded by the query complexity of Dec).
- The correctness and security guarantees will only be with respect to random inputs (we call this "weak security"). One instructive way to think about weak security is to think of a protocol between parties that have no input, and at the beginning of the execution they sample a random input using their local random tape (or shared randomness) and proceed to execute the protocol. Note that this makes simulation easier since we no longer need to worry about consistency with an adversarially chosen (or sampled) input.

In other words, the new protocol Π^* only makes queries to $(\mathsf{OT}_1, \mathsf{OT}_2)$ which are essentially random oracles. Therefore, Π^* securely computes g in the random oracle model. However, it follows from [36, 47] that such a protocol can be used to securely compute g in the information-theoretic setting and this is known to be impossible for the AND functionality [21, 46] (even with weak security as described above).

The remainder of the overview describes this transformation. We transform Π to Π^* through a sequence of steps. We first transform Π to Π^1 in which the first message function of the protocol does not make any OT_3 queries (Step 2 below). Then, we transform Π^1 to Π^2 such that the first and second message functions of the protocol do not make any OT_3 queries (Step 3 below). Finally, we transform Π^2 to Π^* such that the decoder Dec does not make any OT_3 queries (Step 4 below). It is the final step that incurs the blow-up in the round complexity. Additional details follow.

Step-2: $\Pi \Rightarrow \Pi^1$. The first message function of Π has access to $(\mathsf{OT}_1, \mathsf{OT}_2, \mathsf{OT}_3)$ oracles and may make multiple queries to all of them. In order to perform this transformation, we devise a mechanism to emulate the OT_3 oracle without making actual queries to it. Recall that any query to the OT_3 oracle contains $((\mathsf{otm}_1, \mathsf{otm}_2), (b, r))$ and it outputs m_b if and only if there exists (m_0, m_1, s) for which $\mathsf{OT}_1(b, r) = \mathsf{otm}_1$ and $\mathsf{OT}_2(\mathsf{otm}_1, m_0, m_1, s) = \mathsf{otm}_2$. The first step of the OT_3 oracle is easy to emulate; we can query OT_1 on (b, r) and check if the output is otm_1. To emulate the second step, we maintain a list of all the queries/responses made by the first message function to OT_2. If we find an entry $(\mathsf{otm}_1, m_0, m_1, s, \mathsf{otm}_2)$ in this list, we output m_b; else, we output $\perp$. Note that since OT_2 is length tripling, it is

[8] It is easy to see that the security of the transformed protocol requires security against collusion of P_1, P_3 since P_1 has the entire view of P_3. We also require security against (P_2, P_3) collusion since P_1 forwards all its messages sent to P_3 in the second-round of the protocol.

injective with overwhelming probability. Thus, if we find such an entry then our emulation is correct. On the other hand, if we don't find such an entry, we output $\perp$ and it can be easily shown that the original oracle also outputs $\perp$ except with negligible probability. Thus, our emulation is statistically close to the real oracle.

Step-3: $\Pi^1 \Rightarrow \Pi^2$. It might be tempting to conclude that a similar strategy as before should work even for OT_3 queries made in the second round. That is, maintain the list of queries to the OT_2 oracle and when the second message function makes an OT_3 query, check if there is entry in this list with the response equal to otm_2. If such an entry is found, output the corresponding m_b; else, output $\perp$. This strategy fails because in the second round it is possible that the relevant OT_2 query was made by the other party and therefore it is not possible for each party to only consider the list of OT_2 queries made locally. Note, however, that only one simultanous round of communication has been made by the parties so far. Therefore, it must be the case that the party that made the OT_2 query also made the respective OT_1 query.

To take care of such queries, that we call "correlated queries", we modify the first round of Π^1 as follows. The parties will prepare an additional list L that contains all correlated queries that are "likely" to be asked by the other party. (No OT_3 calls will be made while preparing this list.)

The parties will now send this list L along with the first round message of Π^1. Now, when the second message function of a party in Π^1 attempts to make an OT_3 query on $(\mathsf{otm}_1, \mathsf{otm}_2, (b, r))$, we first check if otm_1 is valid (by querying OT_1) and then answer this query as follows. If otm_2 is a result of a local query then find the response using the list of local queries/responses. If otm_2 is a correlated query, use the list L sent by the other party to answer. If we don't find any entry in the local list or the correlated list, we output $\perp$. We show that with overwhelming probability, the real oracle also outputs $\perp$ in this case. We also prove that sending this additional list of "likely" correlated queries does not harm the security of Π^2.

To conclude, we describe how the list L is generated, say by P_1. Note that the list needs to be generated at a point where P_1 already decided on its first Π^1 message; now it just needs to come up with L. To this end, P_1 executes many copies of Π^1 executions of P_2, each time with fresh randomness and random input. Then the list L contains the responses to the list of all correlated OT_2 queries, i.e., the valid queries made to OT_3 by "virtual" P_2 such that both OT_1 and OT_2 have been generated by P_1. This will allow to preserve correctness on an average input, and does not violate privacy since given the first Π^1 messages, anyone can sample such executions.

Step-4: $\Pi^2 \Rightarrow \Pi^*$. At the end of step-3, we have a protocol where the first and the second message functions do not make any queries to the OT_3 oracle. However, for the parties to learn the output, they must run the decoder Dec on the transcript, and this decoder might make queries to OT_3. Recall that Dec is a deterministic decoding function whose input is the transcript of the interaction. Further recall that Π^* will be a protocol that does not use OT_3 but will have many communication rounds.

In Π^*, the parties will first execute the two rounds of Π^2 to obtain a transcript. Then they will jointly execute the decoder, where for each OT_3 query that the decoder needs to make, the party that made the relevant OT_2 query will "help out" by sending the decoding value to the other party. This will proceed for as many rounds as the number of queries that Dec needs to make, but eventually it will allow both parties to complete the execution of Dec locally and compute the output of the functionality. We will then need to show that privacy is not harmed in this process. Details follow.

Let us go back to the point where both parties finished executing the two rounds of Π^2 now wish to engage in joint decoding. One of the parties, say P_1, starts running the decoder on the transcript, and along the way maintain the list of $\mathsf{OT}_1, \mathsf{OT}_2$ made in this process. When the decoder attempts to make an OT_3 query on input $((\mathsf{otm}_1, \mathsf{otm}_2), (b, r))$, P_1 checks if otm_1 is valid (by making a query to OT_1). It then checks if there is an entry $(\mathsf{otm}_1, m_0, m_1, s, \mathsf{otm}_2)$ in the list of OT_2 queries made by the decoder and in the case such an entry is found, it answers with m_b. If such an entry is not found, P_1 checks its local list of queries/responses made to OT_2 during the generation of the first two messages. If it finds an entry $(\mathsf{otm}_1, m_0, m_1, s, \mathsf{otm}_2)$ in that list, it answers with m_b. If even this list does not contain an entry, there are 3 possibilities.

1. otm_2 is not in the image of OT_2 oracle in which case P_1 has to output $\perp$.
2. otm_2 is in the image of OT_2 oracle and P_2 has made this query.
3. otm_2 is in the image of OT_2 oracle and P_2 has not made this query.

The probability that case-3 happens can be shown to be negligible for similar reasons to ones discussed above: if neither party made the relevant OT_2 query then the value otm_2 is almost surely invalid. Thus, P_1 must decide whether it is in case-1 or case-2 and if it is in case-2, it must give the corresponding m_b. To accomplish this, P_1 sends a message to P_2 with (b, otm_2) and asks P_2 to see if there is an entry of the form $(\mathsf{otm}_1, m_0, m_1, s, \mathsf{otm}_2)$ in its local list of queries to OT_2 oracle. If yes, P_2 responds with m_b; else, it responds with $\perp$. P_1 just gives P_2's message as the corresponding response to that query. This blows up the number of rounds of the protocol Π^* proportional to the number of queries made by the decoder.

Observe that Π^* does not make any queries to the OT_3 oracle. At the end, P_1 learns the output $g(x, y)$ and it can send this as the last round message to P_2. Thus, Π^* also has publicly decodable transcript. The correctness of this transformation directly follows since we prove that case-3 happens with negligible probability and if OT_2 is injective (which occurs with overwhelming probability), it follows that if an entry is found in either of the lists of the two parties or on the local list of the decoder, the response given by the emulation is correct.

To see why this transformation is secure, notice that the query $((\mathsf{otm}_1, \mathsf{otm}_2), (b, r))$ is made by the Dec by just looking at the transcript. Hence, there is no harm in P_1 sending (b, otm_2) to the other party. Similarly, if P_2 has indeed made a query to OT_2 such that the response obtained is otm_2, it should follow from the security of Π^2 that the P_2's privacy is not affected if it sends m_b to P_1. Indeed, this information is efficiently learnable given the transcript and an access to the OT_3 oracle. However, there is a subtle issue with this argument which we elaborate next.

Problem of Intersecting Queries. A subtle issue arises when we try to formally reduce the security of Π^* to the security of Π^2. To illustrate this, let us assume the case where P_2 is corrupted. To get a reduction to the security of Π^2, we must give an algorithm that takes the view of P_2 in Π^2 and efficiently generates the view of P_2 in Π^*. In particular, it must generate the additional messages in Π^* given only the view of P_2 in Π^2. This algorithm is allowed to make OT_3 queries as we are trying to give a reduction to the security of Π^2. For the sake of illustration, assume that the Dec makes a single OT_3 query. A natural approach for this algorithm is to take the transcript available in the view of P_2 and start running the decoder on the transcript. When the decoder makes an OT_3 query, the algorithm uses the real OT_3 oracle to respond to this query. However, notice that the algorithm must generate the messages that correspond to answering this OT_3 query in Π^*. Recall that in Π^*, P_1 first checks in its local list whether there an entry of the form $(\mathsf{otm}_1, m_0, m_1, s, \mathsf{otm}_2)$, and only if such an entry is not found, P_1 sends the message (b, otm_2) to P_2. Thus, to generate the

transcript of Π^*, the algorithm must somehow decide whether P_1 would find this entry in its local list or not. However, the algorithm is only given the view of P_2 and does not have any information about the queries that P_1 has made to OT_2.

We see that the problem arises when there is an OT_2 query that potentially was made by both parties. To handle this issue, we resort to the notion of *intersection queries* taken from the key-agreement impossibility result [39, 9]. These works show that it is possible, in polynomial time, to recover a superset of all oracle queries made by both parties (with all but small probability). Given this algorithm, we modify the transformation as follows. The parties will first run the two rounds of communication of Π^2. Then they will run the intersection query finder to recover the intersection query superset. We assume for the purpose of this outline this process is deterministic.[9] Now, upon each potential OT_3 query of the decoder, P_1 will look for the preimage query not only in its query history, but also in the superset of intersection queries, and send a message to P_2 only if the preimage is found in either of this. In particular this means that if the preimage is in the intersection query superset, then we are guaranteed that P_1 will not send a message.

The above modified protocol can be efficiently simulated, since the simulator can also run the intersection query finder and recover the same superset as the parties. Now, if OT_3 gives a valid answer, the simulator looks for a preimage in the intersection query superset. If it finds one, then it concludes that P_1 will not send any message to P_2. If not, then it knows that (except with small probability) exactly one of the parties made the preimage query, and it furthermore knows the internal state of one of the parties, so it knows whether this party made the preimage query. This allows the simulator to always deduce which is the party that made the preimage query and simulate appropriately.

References

1 Prabhanjan Ananth, Arka Rai Choudhuri, and Abhishek Jain. A New Approach to Round-Optimal Secure Multiparty Computation. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part I*, volume 10401 of *Lecture Notes in Computer Science*, pages 468–499. Springer, Heidelberg, August 2017. doi:10.1007/978-3-319-63688-7_16.

2 Prabhanjan Ananth and Abhishek Jain. Indistinguishability Obfuscation from Compact Functional Encryption. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *Advances in Cryptology – CRYPTO 2015, Part I*, volume 9215 of *Lecture Notes in Computer Science*, pages 308–326. Springer, Heidelberg, August 2015. doi:10.1007/978-3-662-47989-6_15.

3 Benny Applebaum. Garbled Circuits as Randomized Encodings of Functions: a Primer. In Yehuda Lindell, editor, *Tutorials on the Foundations of Cryptography.*, pages 1–44. Springer International Publishing, 2017. doi:10.1007/978-3-319-57048-8_1.

4 Benny Applebaum, Zvika Brakerski, and Rotem Tsabary. Perfect Secure Computation in Two Rounds. In *TCC 2018: 16th Theory of Cryptography Conference, Part I*, Lecture Notes in Computer Science, pages 152–174. Springer, Heidelberg, March 2018. doi:10.1007/978-3-030-03807-6_6.

5 Benny Applebaum, Naama Haramaty, Yuval Ishai, Eyal Kushilevitz, and Vinod Vaikuntanathan. Low-Complexity Cryptographic Hash Functions. In Christos H. Papadimitriou, editor, *8th Innovations in Theoretical Computer Science Conference, ITCS 2017, January 9-11, 2017, Berkeley, CA, USA*, volume 67 of *LIPIcs*, pages 7:1–7:31. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017. doi:10.4230/LIPIcs.ITCS.2017.7.

[9] We require that the intersection query finder can be ran consistently by various parties, so if it is randomized it suffices that one of the parties sends a random string that will be used as random tape by all relevant parties.

6 Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography in NC^0. In *45th Annual Symposium on Foundations of Computer Science*, pages 166–175. IEEE Computer Society Press, October 2004. doi:10.1109/FOCS.2004.20.

7 Gilad Asharov and Gil Segev. Limits on the Power of Indistinguishability Obfuscation and Functional Encryption. In Venkatesan Guruswami, editor, *56th Annual Symposium on Foundations of Computer Science*, pages 191–209. IEEE Computer Society Press, October 2015. doi:10.1109/FOCS.2015.21.

8 Boaz Barak. How to Go Beyond the Black-Box Simulation Barrier. In *42nd Annual Symposium on Foundations of Computer Science*, pages 106–115. IEEE Computer Society Press, October 2001. doi:10.1109/SFCS.2001.959885.

9 Boaz Barak and Mohammad Mahmoody-Ghidary. Merkle puzzles are optimal - an $O(n^2)$-query attack on any key exchange from a random oracle. In Shai Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 374–390. Springer, Heidelberg, August 2009. doi:10.1007/978-3-642-03356-8_22.

10 Boaz Barak, Shien Jin Ong, and Salil P. Vadhan. Derandomization in Cryptography. In Dan Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 299–315. Springer, Heidelberg, August 2003. doi:10.1007/978-3-540-45146-4_18.

11 Donald Beaver. Correlated Pseudorandomness and the Complexity of Private Computations. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, pages 479–488, 1996. doi:10.1145/237814.237996.

12 Donald Beaver, Silvio Micali, and Phillip Rogaway. The Round Complexity of Secure Protocols (Extended Abstract). In *22nd Annual ACM Symposium on Theory of Computing*, pages 503–513. ACM Press, May 1990. doi:10.1145/100216.100287.

13 Fabrice Benhamouda and Huijia Lin. k-Round Multiparty Computation from k-Round Oblivious Transfer via Garbled Interactive Circuits. In *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part II*, pages 500–532, 2018. doi:10.1007/978-3-319-78375-8_17.

14 Nir Bitansky and Vinod Vaikuntanathan. Indistinguishability Obfuscation from Functional Encryption. In Venkatesan Guruswami, editor, *56th Annual Symposium on Foundations of Computer Science*, pages 171–190. IEEE Computer Society Press, October 2015. doi:10.1109/FOCS.2015.20.

15 Dan Boneh, Periklis A. Papakonstantinou, Charles Rackoff, Yevgeniy Vahlis, and Brent Waters. On the Impossibility of Basing Identity Based Encryption on Trapdoor Permutations. In *49th Annual Symposium on Foundations of Computer Science*, pages 283–292. IEEE Computer Society Press, October 2008. doi:10.1109/FOCS.2008.67.

16 Elette Boyle, Niv Gilboa, and Yuval Ishai. Breaking the Circuit Size Barrier for Secure Computation Under DDH. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016, Part I*, volume 9814 of *Lecture Notes in Computer Science*, pages 509–539. Springer, Heidelberg, August 2016. doi:10.1007/978-3-662-53018-4_19.

17 Elette Boyle, Niv Gilboa, and Yuval Ishai. Group-Based Secure Computation: Optimizing Rounds, Communication, and Computation. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology – EUROCRYPT 2017, Part II*, volume 10211 of *Lecture Notes in Computer Science*, pages 163–193. Springer, Heidelberg, April/May 2017. doi:10.1007/978-3-319-56614-6_6.

18 Elette Boyle, Niv Gilboa, Yuval Ishai, Huijia Lin, and Stefano Tessaro. Foundations of Homomorphic Secret Sharing. In Anna R. Karlin, editor, *ITCS 2018: 9th Innovations in Theoretical Computer Science Conference*, volume 94, pages 21:1–21:21. LIPIcs, January 2018. doi:10.4230/LIPIcs.ITCS.2018.21.

19 Gabriel Bracha. An O(log n) expected rounds randomized byzantine generals protocol. *J. ACM*, 34(4):910–920, 1987. doi:10.1145/31846.42229.

20 Ran Canetti. Security and Composition of Multiparty Cryptographic Protocols. *Journal of Cryptology*, 13(1):143–202, January 2000. doi:10.1007/s001459910006.

21 Benny Chor and Eyal Kushilevitz. A Zero-One Law for Boolean Privacy (extended abstract). In *21st Annual ACM Symposium on Theory of Computing*, pages 62–72. ACM Press, May 1989. doi:10.1145/73007.73013.

22 Yevgeniy Dodis, Shai Halevi, Ron D. Rothblum, and Daniel Wichs. Spooky Encryption and its Applications. *IACR Cryptology ePrint Archive*, 2016:272, 2016. URL: http://eprint.iacr.org/2016/272.

23 Nico Döttling and Sanjam Garg. From Selective IBE to Full IBE and Selective HIBE. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017: 15th Theory of Cryptography Conference, Part I*, volume 10677 of *Lecture Notes in Computer Science*, pages 372–408. Springer, Heidelberg, November 2017. doi:10.1007/978-3-319-70500-2_13.

24 Nico Döttling and Sanjam Garg. Identity-Based Encryption from the Diffie-Hellman Assumption. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part I*, volume 10401 of *Lecture Notes in Computer Science*, pages 537–569. Springer, Heidelberg, August 2017. doi:10.1007/978-3-319-63688-7_18.

25 Nico Döttling, Sanjam Garg, Yuval Ishai, Giulio Malavolta, Tamer Mour, and Rafail Ostrovsky. Trapdoor Hash Functions and Their Applications. In *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part III*, pages 3–32, 2019. doi:10.1007/978-3-030-26954-8_1.

26 Shimon Even, Oded Goldreich, and Abraham Lempel. A Randomized Protocol for Signing Contracts. *Commun. ACM*, 28(6):637–647, 1985. doi:10.1145/3812.3818.

27 Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate Indistinguishability Obfuscation and Functional Encryption for all Circuits. In *54th Annual Symposium on Foundations of Computer Science*, pages 40–49. IEEE Computer Society Press, October 2013. doi:10.1109/FOCS.2013.13.

28 Sanjam Garg, Yuval Ishai, and Akshayaram Srinivasan. Two-Round MPC: Information-Theoretic and Black-Box. In *TCC 2018: 16th Theory of Cryptography Conference, Part I*, Lecture Notes in Computer Science, pages 123–151. Springer, Heidelberg, March 2018. doi:10.1007/978-3-030-03807-6_5.

29 Sanjam Garg, Mohammad Mahmoody, Daniel Masny, and Izaak Meckler. On the Round Complexity of OT Extension. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018, Part III*, volume 10993 of *Lecture Notes in Computer Science*, pages 545–574. Springer, Heidelberg, August 2018. doi:10.1007/978-3-319-96878-0_19.

30 Sanjam Garg, Mohammad Mahmoody, and Ameer Mohammed. When Does Functional Encryption Imply Obfuscation? In Yael Kalai and Leonid Reyzin, editors, *TCC 2017: 15th Theory of Cryptography Conference, Part I*, volume 10677 of *Lecture Notes in Computer Science*, pages 82–115. Springer, Heidelberg, November 2017. doi:10.1007/978-3-319-70500-2_4.

31 Sanjam Garg and Akshayaram Srinivasan. Two-Round Multiparty Secure Computation from Minimal Assumptions. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part II*, volume 10821 of *Lecture Notes in Computer Science*, pages 468–499. Springer, Heidelberg, April/May 2018. doi:10.1007/978-3-319-78375-8_16.

32 Yael Gertner, Sampath Kannan, Tal Malkin, Omer Reingold, and Mahesh Viswanathan. The Relationship between Public Key Encryption and Oblivious Transfer. In *FOCS 2000*, pages 325–335, 2000.

33 Oded Goldreich. *The Foundations of Cryptography - Volume 2: Basic Applications*. Cambridge University Press, 2004. doi:10.1017/CBO9780511721656.

34 Oded Goldreich, Silvio Micali, and Avi Wigderson. How to Play any Mental Game or A Completeness Theorem for Protocols with Honest Majority. In Alfred Aho, editor, *19th Annual ACM Symposium on Theory of Computing*, pages 218–229. ACM Press, May 1987. doi:10.1145/28395.28420.

35 Iftach Haitner, Yuval Ishai, Eyal Kushilevitz, Yehuda Lindell, and Erez Petrank. Black-Box Constructions of Protocols for Secure Computation. *SIAM J. Comput.*, 40(2):225–266, 2011. doi:10.1137/100790537.

36 Iftach Haitner, Eran Omri, and Hila Zarosim. Limits on the Usefulness of Random Oracles. In Amit Sahai, editor, *TCC 2013: 10th Theory of Cryptography Conference*, volume 7785 of *Lecture Notes in Computer Science*, pages 437–456. Springer, Heidelberg, March 2013. doi:10.1007/978-3-642-36594-2_25.

37 Danny Harnik, Joe Kilian, Moni Naor, Omer Reingold, and Alon Rosen. On Robust Combiners for Oblivious Transfer and Other Primitives. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 96–113. Springer, Heidelberg, May 2005. doi:10.1007/11426639_6.

38 Martin Hirt and Ueli M. Maurer. Player Simulation and General Adversary Structures in Perfect Multiparty Computation. *J. Cryptology*, 13(1):31–60, 2000. doi:10.1007/s001459910003.

39 Russell Impagliazzo and Steven Rudich. Limits on the Provable Consequences of One-Way Permutations. In *21st Annual ACM Symposium on Theory of Computing*, pages 44–61. ACM Press, May 1989. doi:10.1145/73007.73012.

40 Yuval Ishai. Randomization Techniques for Secure Computation. In Manoj Prabhakaran and Amit Sahai, editors, *Secure Multi-Party Computation*, volume 10 of *Cryptology and Information Security Series*, pages 222–248. IOS Press, 2013. doi:10.3233/978-1-61499-169-4-222.

41 Yuval Ishai and Eyal Kushilevitz. Randomizing Polynomials: A New Representation with Applications to Round-Efficient Secure Computation. In *41st Annual Symposium on Foundations of Computer Science*, pages 294–304. IEEE Computer Society Press, November 2000. doi:10.1109/SFCS.2000.892118.

42 Yuval Ishai and Anat Paskin. Evaluating Branching Programs on Encrypted Data. In *Theory of Cryptography, 4th Theory of Cryptography Conference, TCC 2007, Amsterdam, The Netherlands, February 21-24, 2007, Proceedings*, pages 575–594, 2007. doi:10.1007/978-3-540-70936-7_31.

43 Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Founding Cryptography on Oblivious Transfer - Efficiently. In David Wagner, editor, *Advances in Cryptology – CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 572–591. Springer, Heidelberg, August 2008. doi:10.1007/978-3-540-85174-5_32.

44 Joe Kilian. Founding Cryptography on Oblivious Transfer. In *20th Annual ACM Symposium on Theory of Computing*, pages 20–31. ACM Press, May 1988. doi:10.1145/62212.62215.

45 Fuyuki Kitagawa, Ryo Nishimaki, and Keisuke Tanaka. Obfustopia Built on Secret-Key Functional Encryption. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part II*, volume 10821 of *Lecture Notes in Computer Science*, pages 603–648. Springer, Heidelberg, April/May 2018. doi:10.1007/978-3-319-78375-8_20.

46 Eyal Kushilevitz. Privacy and Communication Complexity. In *30th Annual Symposium on Foundations of Computer Science*, pages 416–421. IEEE Computer Society Press, October-/November 1989. doi:10.1109/SFCS.1989.63512.

47 Mohammad Mahmoody, Hemanta K. Maji, and Manoj Prabhakaran. Limits of random oracles in secure computation. In Moni Naor, editor, *ITCS 2014: 5th Conference on Innovations in Theoretical Computer Science*, pages 23–34. Association for Computing Machinery, January 2014. doi:10.1145/2554797.2554801.

48 Mohammad Mahmoody and Rafael Pass. The Curious Case of Non-Interactive Commitments - On the Power of Black-Box vs. Non-Black-Box Use of Primitives. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 701–718. Springer, Heidelberg, August 2012. doi:10.1007/978-3-642-32009-5_41.

49 Pratyay Mukherjee and Daniel Wichs. Two Round Multiparty Computation via Multi-key FHE. In *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II*, pages 735–763, 2016. doi:10.1007/978-3-662-49896-5_26.

50 Periklis A. Papakonstantinou, Charles W. Rackoff, and Yevgeniy Vahlis. How powerful are the DDH hard groups? Cryptology ePrint Archive, Report 2012/653, 2012. URL: https://eprint.iacr.org/2012/653.

51 M. Rabin. How to exchange secrets by oblivious transfer. Technical Report TR-81, Harvard Aiken Computation Laboratory, 1981.

52 Omer Reingold, Luca Trevisan, and Salil P. Vadhan. Notions of Reducibility between Cryptographic Primitives. In Moni Naor, editor, *TCC 2004: 1st Theory of Cryptography Conference*, volume 2951 of *Lecture Notes in Computer Science*, pages 1–20. Springer, Heidelberg, February 2004. doi:10.1007/978-3-540-24638-1_1.

53 Daniel R. Simon. Finding Collisions on a One-Way Street: Can Secure Hash Functions Be Based on General Assumptions? In Kaisa Nyberg, editor, *Advances in Cryptology – EUROCRYPT'98*, volume 1403 of *Lecture Notes in Computer Science*, pages 334–345. Springer, Heidelberg, May/June 1998. doi:10.1007/BFb0054137.

54 Andrew Chi-Chih Yao. How to Generate and Exchange Secrets (Extended Abstract). In *27th Annual Symposium on Foundations of Computer Science*, pages 162–167. IEEE Computer Society Press, October 1986. doi:10.1109/SFCS.1986.25.

Trade-Offs Between Size and Degree in Polynomial Calculus

Guillaume Lagarde
LaBRI, Bordeaux, France
https://guillaume-lagarde.github.io/
guillaume.lagarde@labri.fr

Jakob Nordström
University of Copenhagen, Denmark
KTH Royal Institute of Technology, Stockholm, Sweden
https://www.csc.kth.se/~jakobn/
jn@di.ku.dk

Dmitry Sokolov
Lund University, Sweden
University of Copenhagen, Denmark
https://www.csc.kth.se/~sokolovd
sokolov.dmt@gmail.com

Joseph Swernofsky
KTH Royal Institute of Technology, Stockholm, Sweden
https://www.kth.se/profile/josephsw/
josephsw@kth.se

Abstract

Building on [Clegg et al. '96], [Impagliazzo et al. '99] established that if an unsatisfiable k-CNF formula over n variables has a refutation of size S in the polynomial calculus resolution proof system, then this formula also has a refutation of degree $k + O(\sqrt{n \log S})$. The proof of this works by converting a small-size refutation into a small-degree one, but at the expense of increasing the proof size exponentially. This raises the question of whether it is possible to achieve both small size and small degree in the same refutation, or whether the exponential blow-up is inherent. Using and extending ideas from [Thapen '16], who studied the analogous question for the resolution proof system, we prove that a strong size-degree trade-off is necessary.

2012 ACM Subject Classification Theory of computation → Proof complexity

Keywords and phrases proof complexity, polynomial calculus, polynomial calculus resolution, PCR, size-degree trade-off, resolution, colored polynomial local search

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.72

Acknowledgements The first, second, and fourth authors were funded by the Knut and Alice Wallenberg grant KAW 2016.0066 and the third author by the Knut and Alice Wallenberg grant KAW 2016.0433. In addition, the second author was supported by the Swedish Research Council grants 621-2012-5645 and 2016-00782.

1 Introduction

The main task of proof complexity is to quantify the amount of different resources required to prove that some given formula is unsatisfiable. The particular resources examined depend on the proof system under study, but it is believed that no proof system can have proofs that are both efficiently verifiable and short – that is, polynomial in the size of the given formula. Establishing such an impossibility result in full generality is equivalent to proving

NP $\neq$ coNP, and hence this goal currently seems out of reach. Current research in proof complexity instead focuses on studying more limited, concrete methods of reasoning, and on proving lower bounds for these methods.

In this paper we consider **polynomial calculus resolution (PCR)**. This is a more powerful proof system than what is arguably the most well studied system in all of proof complexity, namely **resolution** [12], but is therefore also less well understood. PCR, introduced by Clegg et al. [15] and Alekhnovich and Razborov [1] can be seen as a dynamic version of Hilbert's Nullstellensatz [6]. In order to refute a CNF formula in PCR, the clauses in the formula are translated into multilinear polynomials, and the proof of unsatisfiability, or **refutation**, consists essentially of certifying that the polynomial 1 is a member of the ideal generated by these polynomials.

Two important masures for PCR refutations are the **size** (the number of monomials in the refutation when all polynomials are expanded out as linear combinations of monomials) and the **degree** (the maximal monomial degree in the refutation). Impagliazzo et al. [24] showed a strong connection between these two measures: if a k-CNF formula over n variables admits a PCR refutation of size S then there is also a refutation of degree $k + O(\sqrt{n \log S})$. This result, which is known by [21] to be tight, plays a crucial role in almost all known size lower bounds for PCR. By proving strong enough degree lower bounds one can also obtain size lower bounds, and techniques for establishing lower bounds on degree have been developed in, e.g., [2, 21, 20, 26]. An interesting aspect of [24], however, is that the small-size refutation is not the same as the small-degree one. Instead, the transformation from small size to small degree increases the proof size exponentially. It is natural to ask whether this exponential blow-up is nescessary.

A similar question arises also in the context of the resolution proof system, where the measures of interest are length and width. Building on [24], Ben-Sasson and Wigderson [10] showed that every small-length resolution refutation can be transformed into a small-width refutation with the same parameters as in [24]. Again, this bound is again known to be tight [13], and the conversion increases the length exponentially. For resolution, Thapen [29] proved that such a blow-up cannot be avoided in the worst case. In this work, we show that this holds true for PCR as well.[1] More precisely we prove the following theorem.

▶ **Theorem 1.** *For any $\epsilon > 0$ and c large enough, there is a CNF formula φ with $\theta(c^{1+\epsilon})$ variables, of size $\theta(c^{1+\epsilon})$ and degree $O(\log c)$ such that*

- *φ has a PCR refutation of size polynomial in c and degree $c + O(\log c)$,*
- *φ has a PCR refutation of degree $O(c^{\epsilon})$,*
- *any PCR refutation of degree strictly less than $c - 1$ has size $\exp\big(c^{\Omega(\epsilon)}\big)$.*

In particular, this implies that any PCR refutation of the formula φ in degree $O(\log c) + O(\sqrt{c^{1+o(1)} \log c}) \ll c$, as obtained by the size-degree bound in [24], must have size $\exp\big(c^{\Omega(\epsilon)}\big)$.

1.1 Related work

The study of connections between different complexity measures has received a fair amount of attention in proof complexity. We give a brief overview below, referring the reader to the book [25] or the upcoming survey chapter [14] for more details.

[1] Note that this is more precise than just saying that there is a trade-off between length and width, or between size and degree, so that minimizing the latter measure must lead to an increase in the former. It is easy to show that there are trade-offs between length/size and width/degree as observed in [27], but such trade-offs are very far from being strong enough to show that the blow-ups in [10, 24] is necessary.

We have already mentioned connections between size and degree in PCR [24] and between length and width in resolution [10]. Let us also mention that in [3] a beautiful relation is exhibited between width and **space** in resolution, showing that width provides a lower bound on space, and that a similar, but weaker, result was recently proved in [19] for PCR.

When studying trade-offs, we are asking whether two different complexity measures can be minimized at the same time, or whether optimizing one measure must lead to an increase in the other in the worst case. This question was first raised in the context of proof complexity by Ben-Sasson [8], who proved trade-offs between space and width in resolution. This result was later extended to size-degree trade-offs in PCR [7], and the result for resolution was extended to a wider regime of parameters in [11].

Trade-offs between length/size and space have been shown for resolution in [5, 9] and for PCR in [7]. There are even some size-space trade-offs for stronger proof systems such as **cutting planes** [16] in [18, 22, 23], but since this proof system will not be relevant for the current paper we will not discuss it further.

For length versus width in resolution, or size versus degree in PCR, it is not too hard to show that there are trade-offs [27], but to prove that the length blow-up in [10] is necessary requires the stronger result in [29]. Recently, Razborov [28] established another length-width trade-off showing that low-width *tree-like* resolution refutations of certain k-CNF formulas must have doubly exponential size,

For Nullstellensatz, which as noted above can be viewed as a weaker version of PCR, strong trade-offs between size and degree were shown in [17]. That paper states as an open problem whether similarly strong results could be established for PCR, and this is the problem that we resolve here.

1.2 Organization of the paper

We start with preliminaries in Section 2 where we define the relevant proof systems and discuss some basic properties of ideals in polynomial rings. In Section 3, we present the family of CNF formulas that we will consider, which we call *safe colored polynomial local search* formulas. Section 4 is devoted to a description of the overall proof strategy. Section 5 develops the machinery used to prove PCR degree lower bounds together with the adaptations needed to obtain our results. Section 6, finally, contains some concluding remarks. We refer to the upcoming full-length version of the paper for the details missing in this extended abstract.

2 Preliminaries

Throughout this paper, we let $\mathbb{F}$ denote a fixed but arbitrary field. For a an integer, we denote by $[a]$ the set $\{0, 1, \ldots, a-1\}$. Given a graph H and a vertex $v \in H$, we write $N(v)$ for the set of neighbours of v in the graph H.

2.1 Resolution and Polynomial calculus

We use x to denote boolean variables ranging over $\{0, 1\}$, and write $\bar{x}$ to denote the negation of x. A **literal** is either a variable x or its negation $\bar{x}$. A **clause** C is a disjunction of literals, such as $x \vee \bar{y}$. The **width** of a clause is the number of literals in it. A **CNF formula** F is a conjunction of clauses, and F is a **k-CNF formulas** if all clauses in it have width at most k. Clauses and formulas are considered as sets, so that there are no repetitions and the ordering is immaterial.

The **resolution** proof system has a single derivation rule $\frac{C \vee x \quad D \vee \bar{x}}{C \vee D}$, called the **resolution rule**, for any clauses C, D and any variable x. In this proof system, a **derivation** of a clause D from a CNF formula $F = C_1 \wedge \cdots \wedge C_m$ is a sequence of clauses such that each clause is either an original clause C_i or the conclusion of the resolution rule applied to previously derived clauses. A **resolution refutation** of F is a derivation of the empty clause containing no literals from F. The **length** of a derivation, or refutation, is the length of the sequence of clauses, and the width is the maximum width of any clause appearing in the sequence.

Moving on to polynomial calculus, we let X denote a set of algebraic variables containing x and $\bar{x}$ for all boolean variables, where x and $\bar{x}$ are considered to be distinct, and consider polynomials in the ring $\mathbb{F}[X]$. We note that in the context of algebraic proof systems it is natural to think of 0 as **True** and 1 as **False**, and so we will use this convention in what follows.

The **PCR** proof system contains the following axioms:

- **boolean axioms:** $x^2 - x$ for all variables x;
- **complementary axioms:** $x + \bar{x} - 1$ for all variables x.

It also contains the following derivation rules:

- **linear combination:** $\frac{p \quad q}{\alpha p + \beta q}$ for any $\alpha, \beta \in \mathbb{F}, p, q \in \mathbb{F}[X]$;
- **multiplication:** $\frac{p}{xp}$ for any $p \in \mathbb{F}[X]$ and $x \in X$.

A polynomial f is **derivable** from a set of polynomials $g_1, \ldots, g_k$ (written $g_1, \ldots, g_k \vdash f$) if there is a sequence of polynomials such that each polynomial is either an axiom, an original polynomial g_i, or the conclusion of a derivation rule applied to previously derived polynomials. We refer to such a sequence of polynomials as a **PCR derivation**. A **PCR refutation** of a set of polynomials $g_1, \ldots, g_k$ is a derivation of the polynomial 1 from $g_1, \ldots, g_k$.

▶ **Example 2.** Over the variables x, y, z, if we are given xz and $y\bar{z}$ we can derive xy. This is a simulation, in PCR, of the resolution rule deriving $x \vee y$ from $x \vee z$ and $y \vee \bar{z}$.

$$\dfrac{\dfrac{\dfrac{y\bar{z}}{xy\bar{z}} \quad \dfrac{z + \bar{z} - 1}{xyz + xy\bar{z} - xy}}{xy - xyz} \quad \dfrac{xz}{xyz}}{xy}$$

A set I of polynomials in $\mathbb{F}[X]$ is an **ideal** if I is closed under linear combination and multiplication by any polynomial in $\mathbb{F}[X]$. Given a set of polynomials $S = \{g_1, \ldots, g_k\}$, the **ideal generated by** S is defined as the smallest ideal I_S that contains the set S.

Observe that, by definition, $g_1, \ldots, g_k \vdash f$ is equivalent to saying that f is in the ideal generated by $g_1, \ldots, g_k$ together with all boolean and complementary axioms. Intuitively, a PCR refutation is a certificate that the system of polynomial equations $\{g_1 = 0, \ldots, g_k = 0\}$ has no boolean solutions. If the polynomials have no common boolean solution then there always exists such a certificate, since it can be shown that 1 lies in the ideal generated by the polynomials arising in the system together with the polynomials from the boolean and complementary axioms. In other words, the PCR proof system is *sound* and *complete*.

The **size** of a PCR refutation is the total number of non-zero monomials (counted with repetition) that appear in the derivation when all polynomials are expanded out as linear combinations of monomials. The **degree** of a PCR refutation is the maximal degree of a non-zero monomial that appears in the derivation.

There is a standard **translation** $tr(\cdot)$ from clauses to monomials defined by induction in the following way:

- $tr(x) = x$;
- $tr(\neg x) = \bar{x}$;
- $tr(C \vee D) = tr(C) \cdot tr(D)$.

Using $tr(\cdot)$, any k-CNF formula F with s clauses can be converted into a set of s polynomials of degree at most k by applying the translation to all clauses in F. Denote this set by $tr(F)$. It is not hard to see that F is satisfiable if and only if the set of polynomials $tr(F)$ have a common boolean root. Therefore F is an unsatisfiable CNF formula if and only if there is a PCR refutation of the set of polynomials $tr(F)$. Furthermore, a straightforward generalization of Example 2 shows that a resolution refutation in length ℓ and width w can be converted into a PCR refutation in size $O(\ell w)$ and degree $w+1$.

A **restriction** is a partial assignment of the variables, that is a function $\rho : X \to X \cup \mathbb{F}$ such that the value $\rho(x)$ is either x or a constant from $\mathbb{F}$. For a polynomial p, we denote by $p \restriction \rho$ the polynomial p in which any variable x is replaced by $\rho(x)$.

2.2 Reduction over ideals

Two polynomials p, q are said to be **equivalent modulo an ideal** I, written $p \sim_I q$, if $p - q \in I$. This is an equivalence relation. For any polynomial p we fix a special representative of the equivalence class $[p]$ that we call the **reduction of** p **modulo** I and write as $R_I(p)$. If an ideal I is generated by a set of polynomials S, we abuse notation slightly and write $R_S(p)$ for $R_I(p)$.

To define the representative, we fix any order $\prec$ on the polynomials that respects inclusion as follows:

1. Firstly, for two monomials m_1 and m_2 we have $m_1 \prec m_2$ whenever m_1 is a submonomial of m_2.
2. Secondly, we extend this in an arbitrary way to a total order on monomials.
3. Finally, we order polynomials in the following way. To any polynomial p we associate a sequence s_p consisting of the non-zero monomials of p, sorted in decreasing order with respect to $\prec$. The polynomials are then compared by comparing lexicographically their associated sequences. For example, if $m_1 \prec m_2 \prec m_3$, then the polynomial m_2 is strictly smaller than the polynomial $m_2 + m_1$, which is itself strictly smaller than the polynomial m_3.

The representative $R_I(p)$ is then defined as $\min(\{q \in [p]\})$. We now observe two easy but useful properties of $\prec$.

▶ **Fact 3.** *For any restriction ρ, we have that $p \restriction \rho \preceq p$.*

Proof. To see this, observe that any monomial in $p \restriction \rho$ is either a monomial or a submononial from p; this can only decrease p. ◀

▶ **Fact 4.** *If I_1, I_2 are two ideals such that $I_1 \subseteq I_2$, then*

$$R_{I_2}(p \times R_{I_1}(q)) = R_{I_2}(p \times q).$$

Proof. By definition, $q = R_{I_1}(q) + h$ for some $h \in I_1$. Therefore

$$\begin{aligned} R_{I_2}(p \times q) &= R_{I_2}(p \times (R_{I_1}(q) + h)) \\ &= R_{I_2}(p \times (R_{I_1}(q)) + R_{I_2}(p \times h) \end{aligned}$$

by linearity. To conclude, observe that $p \times h \in I_1$ so is reduced to zero modulo I_1, and hence is reduced to zero modulo I_2 since $I_1 \subseteq I_2$. ◀

3 The formula

In this section, we describe the construction of the family of formulas for which we establish our size-degree trade-off result. We also argue why they have small-size refutations. Our formulas encode a modified version of the **colored polynomial local search (CPLS)** principle, and are very similar to the ones used in [29] to prove trade-offs between length and width in resolution, but there are two main differences. First, we add an *extra color* that we call the *safe color* that plays a role slightly different from the normal ones. The reason for adding this color is purely technical – we were not able to make the PCR machinery works without it. Second, instead of using a grid graph where any two consecutive layers form a complete bipartite graph, we restrict the edges between two consecutive layers to be a well-chosen expander graph. Since this makes the formula more constrained than that in [29], it makes our lower bound slightly stronger.

3.1 The modified CPLS formula

Let a, b, c be positive integers. We work only with graphs $H = (V, E)$ of the following form:

- The set V of vertices are given by $\{(i, x), i \in [a], x \in [b]\}$. We say a vertex (i, x) is the vertex x on layer i.
- Edge appear only between consecutive layers. I.e., if $((i, x), (i', x'))$ is an edge then $i' = i + 1$.

To any such graph H we associate a formula $CPLS^H(a, b, c)$ (or simply $CPLS(a, b, c)$ if H is clear from the context). Intuitively, the formula will give a set of colors (which is a subset of $[c]$) to each node in H according to the following rules.

1. Node $(0, 0)$ gets no color.
2. From every node u there is some special neighbor v on the next layer. If v gets a color then u gets a color. Specifically, there is a **safe color** (corresponding to the first color from $[c]$) and either u gets the safe color or it gets all of v's colors.
3. Every node on the bottom layer gets some color.

Of course, this means there is a path of special neighbors from $(0, 0)$ to the bottom layer. The last node must get some color, so we can trace backward and see $(0, 0)$ gets some color, a contradiction. Hence $CPLS(a, b, c)$ is unsatisfiable. Now we state this formally. We use $\Theta(abc)$ variables:

- $G(u, y)$ for each $u \in V$ and $y \in [c]$. This says whether y is set at u.
- $f(u, v)$ for each $u \in V$ and $v \in N(u)$. This says whether v is a special neighbor of u.
- $h(u)_j$ for each $u = (a - 1, x)$ and $j \in [\log c]$. The function $h(u)$ identifies a color that must be present at u, but is encoded in binary.

Now we can restate the intuitive rules above as formal axioms:

1. for each color $y \in [c]$,
   $\neg G((0, 0), y)$;
2. for each node $u \in V$
   (a) for each neighbor $v \in N(u)$, and color $y \in [c]$,
   $(f(u, v) \wedge G(v, y)) \to G(u, y) \vee G(u, 0)$;
   (b) if u is not on the bottom layer,
   $\bigvee_{v \in N(u)} f(u, v)$;
3. for each node $u \in V$ on the last layer and color $y \in [c]$,
   $(h(u) = y) \to G(u, y)$.

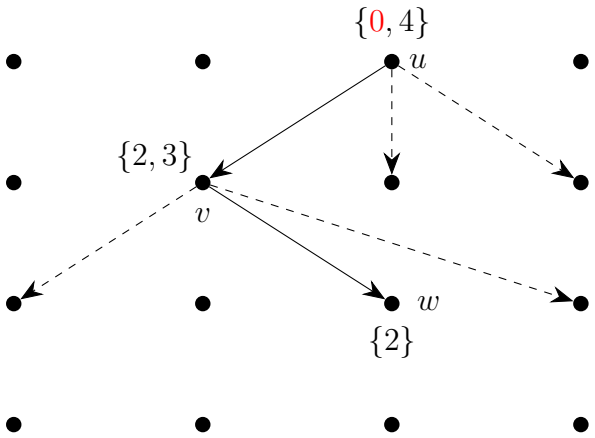


Figure 1 Nodes u and v have three possible neighbors represented by the arrows. The two arrows that are not dotted represent the actual values of f; for these arrows, axioms 2a are respected. Indeed the axioms $P_{v,w,\cdot}$ are satisfied because the colors at v is a superset of the colors at w; the axioms $P_{u,v,\cdot}$ are satisfied because u contains the safe color 0.

See Figure 1 for an illustration of the formula. We refer to instances of axioms 2a, 2b, and 3 as $P_{u,v,y}, Q_u$, and $T_{u,y}$ respectively.

▶ **Definition 5.** *An (s,e)-**expander** is a bipartite graph (V_L, V_R, E) such that for any subset of "left" vertices $S \subseteq V_L$ such that $|S| \leq s$, the following holds: $|N(S)| \geq e|S|$. We recall that $N(S)$ is the neighborhood of S.*

We will use $a = b = c^\epsilon$ in this paper. Let H_i be H restricted to the i^{th} layer. That is $H_i \coloneqq H \cap \{(i,x),(i+1,y) \mid x,y \in [b]\}$. We need H_i to be an $(a^{7/8}, e)$-expander, where any constant $e > 1$ suffices. We also need H_i to have vertex degrees bounded above and below by some constants. For simplicity we assume it has constant degree d.

▶ Remark 6. Note that it is also possible to ask each H_i to be a full bipartite graph. Because the vertex degree then grows with a, we need to change the formula by encoding the neighbors of any vertex in "binary", as in Thapen's paper [29], in order to avoid the width becoming too large (due to type 2b axioms).

3.2 Short and narrow refutations

We give a refutation of $CPLS^H(a,b,c)$ which is of small size. While this refutation can be translated into a refutation of small degree, by a result from [24], we additionally give a much lower degree refutation. The rest of the paper will consist of proving that small size and small degree cannot be obtained simultaneously. The proof of the next two propositions can be found in the appendix of the full version and are very similar to the one from Thapen [29].

▶ **Proposition 7.** *For any graph H, $CPLS^H(a,b,c)$ has a resolution refutation Π of length $O(ab^2c)$ and width $c + \log b + 1$.*

As discussed in Section 2, this gives a PCR refutation of size $O(a^2b^4c^2)$ and degree $c+\log b+2$.

▶ **Proposition 8.** *For any graph H, $CPLS^H(a,b,c)$ has a resolution refutation Π of width $a(\mathrm{d}+1) + \log c$.*

4 Restricting the formula and refutation

The strategy we are going to use is as follows: we suppose for contradiction that we have a refutation Π of the formula $CPLS$ which is both of small size and small degree. Then we will apply a random restriction ρ to both $CPLS$ and Π, which is a partial assignment to the variables. At this point, we have in hand a refutation $\Pi \restriction \rho$ of the restricted formula $CPLS \restriction \rho$. By the assumption that Π is both of small size and small degree, we show that $\Pi \restriction \rho$ enjoys some nice properties; we call such a refutation a **beautiful** refutation. We conclude by showing that $CPLS \restriction \rho$ cannot have any beautiful refutation, yielding the desired contradiction.

We start with the definition of ρ.

4.1 The restriction

Let $p = a^{-3/4}$ and $w = a^{7/8}$. The restriction ρ is randomly set in the following way.

- With probability p, for any vertex u, we set independently for all $y \in [1, c-1]$ the value of $G(u, y)$ to True or False with probability $1/2$; if this happens for a vertex u, we say that **the colors at u are set**. Moreover, if such a vertex is at the bottom layer, then we select one color y that has been set to True during the process and we set $h(u) = y$.
- If we set the colors at u then we also give the safe color to this vertex, i.e., we set $G(u, 0) =$ True. Otherwise we set $G(u, 0) =$ False.
- If we set the colors at u then with probability $\frac{1}{2}$ we also set $f(u, v) =$ True for a uniformly random chosen vertex $v \in N(u)$ and we set all others values $f(u, \cdot)$ to zero.

For the rest of the paper, we write this restriction ρ.

▶ Remark 9. Note that for any u where the colors are set, u gets the safe color. Hence we satisfy all type 2a axioms mentioning $f(u, \cdot)$ and we can treat the vertex u as if it is removed from the graph. For any node v with an edge (v, u) in the graph this reduces the (out-)degree of v by 1. We must maintain a positive degree for vertices outside the last layer. If vertices have degree at least 3 then the chance of reducing any vertex degree to 0 is $o(1)$. Hence our random choice of ρ works with high probability in this case. It is still a $CPLS$ formula (over a smaller graph) and we can now reason over this new graph.

4.2 Beautiful properties after restriction

In this section, we prove that applying the above restriction ρ to a small sized refutation Π gives a refutation $\Pi \restriction \rho$ of $CPLS \restriction \rho$ enjoying useful properties.

▶ **Definition 10.** *A term t **touches** vertex $u \in H$ iff t contains at least one of the following:*

- *$G(u, y)$ for some $y \in [c] \setminus \{0\}$;*
- *$f(u, v)$ for some $v \in N(u)$;*
- *$h(u)_j$ for some $j \in [\log c]$.*

*The **vertex-degree** of a term t is the number of vertices it touches.*

See Figure 2 for an example.

▶ **Definition 11.** *We say that a term t is **almost beautiful** if:*

- *the vertex-degree of t is at most $w + 1$;*
- *it does not contain any $G(\cdot, 0)$ variable;*
- *it touches at most $c - 2$ different colors.*

*If the vertex-degree of t is at most w then we say it is **beautiful**. By extension, a derivation Π is beautiful if every term that appears in Π is beautiful.*

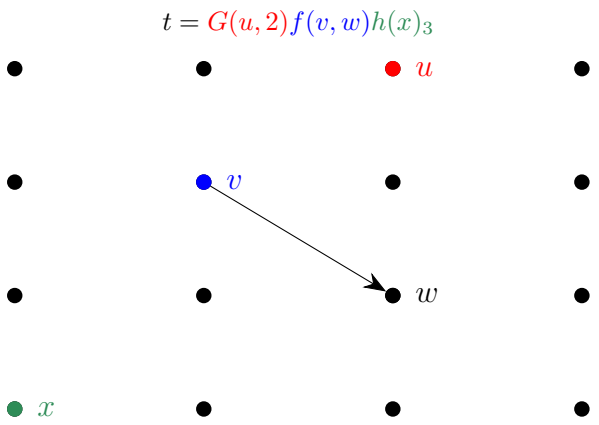


Figure 2 The term t touches three vertices that are the ones colored in the figure.

▶ **Proposition 12.** *There is an ϵ so that if $|\Pi| = s \leq 2^{c^\epsilon}$ then with probability $1 - s \cdot 2^{-c^\epsilon}$ over the choice of ρ, every term in $\Pi \restriction \rho$ is beautiful.*

Proof. Consider a term t and suppose it touches at least w distinct vertices u. We show that except with exponentially small probability ρ sets t to 0.

Suppose we set the colors at u, which happens with probability p. If t contains $G(u,y)$ for $0 < y$ then with probability $\frac{1}{2}$ we set $G(u,y) = 0$. If t contains $h(u)_j$ then because $h(u)$ is uniformly random we set $h(u)_j = 0$ with probability $\frac{1}{2}$.

We also set the arrow at u with probability $p/2$. There are d neighbors of u so we set $f(u,v)$ to True with probability $\frac{1}{d}$ and False with probability $\frac{d-1}{d}$ Hence if t contains $f(u,v)$ then we set t to 0 with probability at least $\frac{p}{2d}$.

Hence, for each u, if t touches u then with probability at least $\frac{p}{2d}$ we set t to 0. For distinct u these events are independent so the probability of not setting t to 0 is at most:

$$(1 - \frac{p}{2d})^w < e^{-\frac{1}{2d}a^{1/8}}.$$

To conclude the proof, observe that $t \restriction \rho$ does not contain any $G(\cdot, 0)$ variable since the restriction sets the value $G(u,0)$ to True of False for any vertex u. ◀

5 PCR machinery

In this section, we ask the order on monomials to have the additional property that it respects the vertex-degree: if a monomial m_1 has a vertex-degree smaller than a monomial m_2, then $m_1 \prec m_2$.

5.1 R operator

We follow a strategy similar in spirit to the technique developed by Alekhnovich and Razborov [2]. The idea is to define a linear map $R : \mathbb{F}[X] \to \mathbb{F}[X]$ that acts as a witness that the polynomial 1 is not reachable from the axioms by any beautiful derivation. Such a linear map was constructed by Alekhnovich and Razborov to separate the polynomial 1 from polynomials reachable by small degree derivations. It has been applied to Tseitin tautologies, among other formulas. Note that Mikša and Nordström [26] gave some sufficient (and quite general) conditions to prove existence of such operators. Unfortunately, we can not use directly these results since they deal with degree and we therefore need to adapt them to handle our notion of beautifulness. More precisely, we aim to prove the following theorem.

▶ **Theorem 13.** *There is a linear map $R : \mathbb{F}[X] \to \mathbb{F}[X]$ such that:*

1. $R(A) = 0$ *for any axiom A;*
2. $R(1) \neq 0$*;*
3. $R(xt) = R(xR(t))$ *whenever xt is a beautiful monomial.*

Theorem 13 is sufficient to prove our main theorem, since any polynomial derived by a beautiful derivation is mapped to zero under the R operator, leaving the polynomial 1 unreachable.

The operator R that we will use is defined on monomials and then extended linearly to general polynomials. To do that, we associate to any monomial t a set of vertices, written $\mathrm{Supp}(t)$, and the value $R(t)$ is defined to be the reduction of t under the ideal generated by axioms associated to $\mathrm{Supp}(t)$. Again, we abuse notation slightly and write this reduced polynomial as $R_{\mathrm{Supp}(t)}(t)$.

More formally, to a given set A of vertices, we identify the following set of axioms:

$$\begin{aligned}
&\{Q_u \mid u \in A\}\\
&\cup \{P_{u,v,y} \mid u \in A, v \in N(u), y \in [c]\}\\
&\cup \{T_{u,y} \mid u \in A, y \in [c]\}\\
&\cup \{\neg G((0,0), y) \mid y \in [c]\}\\
&\cup \text{all boolean axioms}\\
&\cup \text{all complementary axioms.}
\end{aligned}$$

This set contains some axioms that are related to set A as well as the axioms saying that $(0, 0)$ is not colored. $R_A(t)$ is then interpreted as the reduction of t under the ideal generated by the axioms above.

5.2 Closure/support

We first associate to any term t a set of vertices called its *support* and written $\mathrm{Supp}(t)$. The motivation here is that only axioms associated with certain vertices could have been meaningfully used to derive t. Any axiom that mentions a variable of t certainly could be used, but axioms associated with a node u that is in turn associated with a variable in t may be used, and even axioms at some nearby nodes in H may be used.

The support is defined using the following closure operation:

▶ **Definition 14.** *Given a bipartite graph with partition (V_L, V_R), and $S \subseteq V_R$ a subset of "right" vertices. We say that $\mathrm{Cl}(S) \coloneqq \{v \in V_L \mid N(v) \subseteq S\}$ is the* ***closure*** *of S.*

The following property of the closure will be useful.

▶ **Lemma 15.** *Let $G = (V_L, V_R)$ be an (s, e)-expander. If $S, S' \subseteq V_R$ such that $|S|, |S'| < e\frac{s}{2}$ then:*

- $|\mathrm{Cl}(S)| \leq \frac{|S|}{e}$*;*
- $\mathrm{Cl}(S) \cup \mathrm{Cl}(S') \subseteq \mathrm{Cl}(S \cup S')$*;*
- $\forall V \subseteq (V_L \setminus \mathrm{Cl}(S))$ *such that* $|V| \leq \frac{s}{2}$ *we have* $N(V) \not\subseteq S$*.*

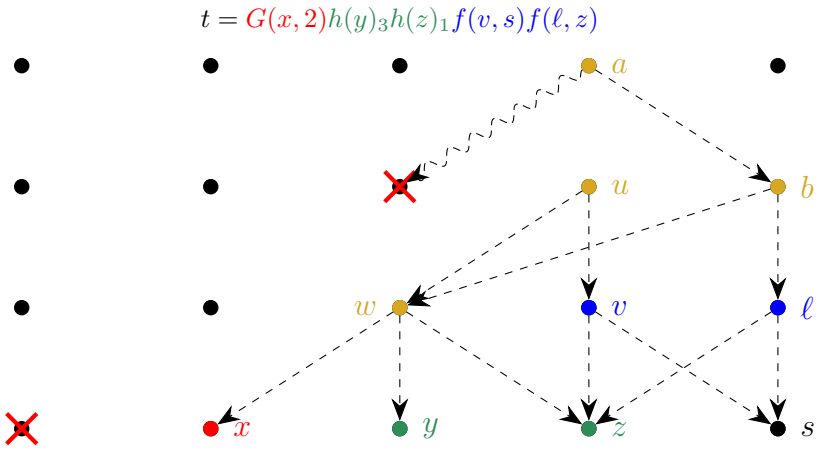


Figure 3 Example of the support of a term t. Color notation:

- nodes for which t contains f variables: $\{v, \ell\}$;
- nodes for which t contains G variables: $\{x\}$;
- nodes for which t contains h variables: $\{y, z\}$;
- nodes that are not touched by t but included in $\mathrm{Supp}(t)$: $\{a, b, u, w\}$;
- nodes that are banished from the graph after application of the restriction from Section 4.

The proof is straightforward and appears in the full version. We can now define the **support** (and **extended support**) associated with a term t. We do this by a downward induction on i over the layers of the graph H as follows:

$$\begin{aligned} b_i &:= \{(i,x) \mid \exists v. f((i,x),v) \text{ or } h((i,x)) \text{ appears in } t\} \\ c_i &:= \{(i,x) \mid \exists y. G((i,x),y) \text{ appears in } t\} \\ d_{i-1} &:= b_{i-1} \cup \mathrm{Cl}(d_i \cup c_i) \\ \mathrm{Supp} &:= \bigcup_{i \in [a]} d_i \\ \mathrm{ExSupp} &:= \bigcup_{i \in [a]} (d_i \cup c_i) \end{aligned}$$

▶ Remark 16. For any terms t, t' it holds that

$$\mathrm{Supp}(t) \cup \mathrm{Supp}(t') \subseteq \mathrm{Supp}(t \cup t').$$

Proof. Follows from Lemma 15, third item. ◀

▶ **Lemma 17.** *If t touches at most ℓ vertices where $\ell \leq 2w$, then $|\mathrm{Supp}(t)| \leq |\mathrm{ExSupp}(t)| = O(\ell)$.*

The proof uses a simple induction and telescoping sum and appears in the full version.

5.3 Proof of Theorem 13

The following propositions are sufficient to prove that the R operator fulfills the desired properties.

▶ **Proposition 18.** *For any variable x, any monomial t such that xt is beautiful, and any $t' \in R(t)$, we have $R_{\mathrm{Supp}(xt')}(xt') = R_{\mathrm{Supp}(xt)}(xt')$.*

▶ **Proposition 19.** *For any almost beautiful monomial t and any $A \subseteq H$ with $|A| \leq O(w)$ we have:*

$$R_{\mathrm{Supp}(t)\cup A}(t) = R_{\mathrm{Supp}(t)}(t).$$

With these propositions in hand, let us now prove Theorem 13

Proof of Theorem 13. We start with the first item, that $R(A) = 0$ for any axiom A. Consider some axiom A and note that it touches a set of vertices $H_A \subseteq H$ of constant size. Let $\sum t_i$ be a polynomial representation of A. By Remark 16, $\bigcup \mathrm{Supp}(t_i) \subseteq \mathrm{Supp}(\bigcup t_i)$ and by Proposition 19, $R(\sum t_i) = \sum R_{\mathrm{Supp}(t_i)}(t_i) = \sum R_{\mathrm{Supp}(\bigcup t_i)\cup H_A}(t_i) = 0$.

For the second item, that $R(1) \neq 0$, observe that since $\mathrm{Supp}(1) = \emptyset$ the set of axioms associated to it is simply:

$$\{\neg G((0,0), y) \mid y \in [c]\}$$
$$\cup \text{ all boolean axioms}$$
$$\cup \text{ all complementary axioms.}$$

This is a satisfiable set of polynomial equations and thus the polynomial 1 is irreducible over the ideal generated by $\mathrm{Supp}(1)$.

Let us now prove the third item, that $R(xt) = R(xR(t))$ whenever xt is beautiful. Consider a beautiful term xt.

$$\begin{aligned}
R(xR(t)) &= \sum_{t' \in R(t)} R(xt') && \text{by linearity} \\
&= \sum_{t' \in R(t)} R_{\mathrm{Supp}(xt')}(xt') && \text{by definition of } R(\cdot) \\
&= \sum_{t' \in R(t)} R_{\mathrm{Supp}(xt)}(xt') && \text{using Proposition 18} \\
&= R_{\mathrm{Supp}(xt)}(xR(t)) && \text{by linearity} \\
&= R_{\mathrm{Supp}(xt)}(xR_{\mathrm{Supp}(t)}(t)) && \text{by definition of } R(\cdot)
\end{aligned}$$

Observe that since $\mathrm{Supp}(t) \subseteq \mathrm{Supp}(xt)$, we can remove the righthand R operator using Fact 4. We then get

$$\begin{aligned}
&= R_{\mathrm{Supp}(xt)}(xt) \\
&= R(xt)
\end{aligned}$$

by definition of $R(\cdot)$. ◀

5.4 Proof of Proposition 19

Proof of Proposition 19. We assume for sake of contradiction that the proposition is false and let A be a minimal witness of this. Without loss of generality $A \cap \mathrm{Supp}(t) = \emptyset$ and A is nonempty. We find A' with $|A'| < |A|$ where the lemma also fails.

Let $r \coloneqq R_{\mathrm{Supp}(t)\cup A}(t)$. We have an equation $t = r + \sum q_i p_i$ where p_i is one of the axiom that correspond to $\mathrm{Supp}(t) \cup A$. By assumption, we also have that $r \prec R_{\mathrm{Supp}(t)}(t)$. We want to construct an assignment α that:

- sets p_i to 0 for at least one $p_i \in A$;
- leaves t unchanged;
- for any p_i either satisfies it or leaves it unchanged.

If such an assignment exists then $t \restriction \alpha = r \restriction \alpha + \sum (q_i \restriction \alpha)(p_i \restriction \alpha)$ and thus $t = r \restriction \alpha + \sum q'_i (p_i \restriction \alpha)$ with $r \restriction \alpha \preceq r$. Since α sets at least one p_i, this shows we can reduce t using only $\mathrm{Supp}(t) \cup (A \setminus \{p_i\})$, as desired.

First observe that if there is a vertex $u \in A$ with neighbor $v \in N(u) \setminus (A \cup \mathrm{ExSupp}(t))$ then we can:

- set the arrow out of u to point to v ($\alpha(f(u,v)) = \mathrm{True}$);
- set all other arrows out of u to False ($\forall v' \in N(u) \setminus v . \alpha(f(u,v')) = \mathrm{False}$);
- set all colors to False at v ($\forall y \in [c] . \alpha(G(v,y)) = \mathrm{False}$).

This satisfies all $P_{u,\cdot,\cdot}$ and Q_u axioms at u and leaves axioms at other vertices in $A \cup \mathrm{Supp}(t)$ untouched. Also $v \notin \mathrm{ExSupp}(t)$ and hence t is also untouched.

The only way we can fail to find such a u and v is if $N(A) \subseteq A \cup \mathrm{ExSupp}(t)$. Note that this implies A contains a vertex in the last layer. Indeed, consider the largest $i \in [a]$ that contains some vertices from A. If $i < a - 1$ then for any vertex $u \in A$ on this layer there is at least one neighbor $v \notin \mathrm{ExSupp}(t)$ because otherwise by definition of closure u should be included in the $\mathrm{Supp}(t)$. Since we picked the largest possible i we also know $v \notin A$.

If A contains a vertex on the last layer then we consider some layer i such that axioms from $\mathrm{ExSupp}(t) \cup A$ do not touch any vertex on layers $i, i+1$ and let

$$B \coloneqq \{(i', j) \mid i' > i, j \in [b]\}.$$

B is the set of all vertices between this layer and the bottom layer. Since $|A| + |\mathrm{Supp}(t)| = O(w)$ we know such a row exists. We pick some color y not mentioned by t, set $G(u,y) = \mathrm{True}$ for all $u \in B$, and set $h((a-1,x)) = y$ for any $(a-1,x) \in B \setminus \mathrm{Supp}(t)$.

Furthermore, for any $u \in B \cap A$ that is not on the last layer we know by the second item of Lemma 15 that u has a neighbor $z \notin \mathrm{ExSupp}(t)$. We set f so it points u to z and set $G(z,y') = \mathrm{False}$ for all other colors. Here we use the fact that $z \notin \mathrm{ExSupp}(t)$ and if we set colors for z the term t remains unchanged. That is, $f(u,v) = \mathrm{False}$ for all $v \in N(u) \setminus \{z\}$ and $f(u,z) = \mathrm{True}$. If there were no such z then u would be included in $\mathrm{Supp}(t)$ and hence would not be in A.

By setting colors in this way we satisfy all axioms for all $u \in B \cap A$. At $u \in B \cap \mathrm{Supp}(t)$ we only set a single color to True and satisfy the corresponding axiom. We can always pick a color y because $|t| < c$ and we touch no variable in t. Finally, we satisfy the Q_u and $T_{u,y}$ axioms anywhere they are touched by setting h and f. ◀

The proof of Proposition 18 uses similar ideas to that of Proposition 19 and appears in the full version.

6 Concluding remarks

In this paper, we have shown that although it is known from [24] that size provides an upper bound on degree in polynomial calculus resolution (PCR) – in the sense that if a set of degree-k polynomials have a PCR refutation in size S, then there is also a refutation in degree $k + O(\sqrt{n \log S})$ – it is not possible in general to construct PCR refutations that get even remotely close to these two upper bounds simultaneously. This extends the analogous trade-off result in [29] for the weaker resolution proof system.

We would like to remark that while our trade-off result is currently stated for formulas of logarithmic width – in order to avoid extra technical complications – it is also possible to obtain the result for constant-width formulas. This involves converting the wide clauses in our formulas to 3-CNF using the standard approach with extension variables. Since the upper bounds for our formulas also work in resolution, this strengthens [29]. We refer to the upcoming full-length version of the paper for the details.

As mentioned in the introduction, in addition to generalizing [29] from resolution to PCR, our main theorem is also a counterpart of the size-degree trade-offs for the weaker algebraic system Nullstellensatz established in [17]. In contrast to our result, however, the Nullstellensatz trade-offs only hold for the version of the proof system without formal variables for negated literals. It would be desirable to obtain more robust trade-off theorems that also applies to Nullstellensatz with separate formal variables for positive and negative literals.

Another, even more interesting, question would be to investigate stronger proof systems. Very recently, Atserias and Hakoniemi [4] showed that if a system of degree-k polynomial constraints over n boolean variables has a **Sums-of-Squares (SOS)** refutation of size S, then it also has a refutation of degree at most $O(k + \sqrt{n \log S})$. They also proved a similar statement for the stronger **Positivstellensatz** proof system, and an analogous result for the weaker system **Sherali-Adams** also follows from the proofs in the paper. However, for these proof systems the question of whether the conversion from small size to small degree can be achieved without blowing up the size remains open, and it is not even known if the upper bound on degree in terms of size in [4] is tight.

References

1. Michael Alekhnovich, Eli Ben-Sasson, Alexander A. Razborov, and Avi Wigderson. Space Complexity in Propositional Calculus. *SIAM J. Comput.*, 31(4):1184–1211, 2002. doi:10.1137/S0097539700366735.
2. Michael Alekhnovich and Alexander A. Razborov. Lower Bounds for Polynomial Calculus: Non-Binomial Case. *Proceedings of the Steklov Institute of Mathematics*, 242:18–35, 2003. Available at http://people.cs.uchicago.edu/~razborov/files/misha.pdf. Preliminary version in *FOCS '01*.
3. Albert Atserias and Víctor Dalmau. A combinatorial characterization of resolution width. *J. Comput. Syst. Sci.*, 74(3):323–334, 2008. doi:10.1016/j.jcss.2007.06.025.
4. Albert Atserias and Tuomas Hakoniemi. Size-Degree Trade-Offs for Sums-of-Squares and Positivstellensatz Proofs. In *34th Computational Complexity Conference, CCC 2019, July 18-20, 2019, New Brunswick, NJ, USA.*, pages 24:1–24:20, 2019. doi:10.4230/LIPIcs.CCC.2019.24.
5. Paul Beame, Chris Beck, and Russell Impagliazzo. Time-Space Tradeoffs in Resolution: Superpolynomial Lower Bounds for Superlinear Space. *SIAM Journal on Computing*, 45(4):1612–1645, August 2016. Preliminary version in *STOC '12*. doi:10.1137/130914085.
6. Paul Beame, Russell Impagliazzo, Jan Krajíček, Toniann Pitassi, and Pavel Pudlák. Lower Bounds on Hilbert's Nullstellensatz and Propositional Proofs. *Proceedings of the London Mathematical Society*, s3-73(1):1–26, 1996. doi:10.1112/plms/s3-73.1.1.

7 Chris Beck, Jakob Nordström, and Bangsheng Tang. Some Trade-off Results for Polynomial Calculus. In *Proceedings of the 45th Annual ACM Symposium on Theory of Computing (STOC '13)*, pages 813–822, May 2013. doi:10.1145/2488608.2488711.

8 Eli Ben-Sasson. Size Space Tradeoffs for Resolution. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing (STOC '02)*, pages 457–464, May 2002. doi:10.1145/509907.509975.

9 Eli Ben-Sasson and Jakob Nordström. Understanding Space in Proof Complexity: Separations and Trade-offs via Substitutions. In *Proceedings of the 2nd Symposium on Innovations in Computer Science (ICS '11)*, pages 401–416, January 2011. arXiv:1008.1789.

10 Eli Ben-Sasson and Avi Wigderson. Short proofs are narrow - resolution made simple. *J. ACM*, 48(2):149–169, 2001. doi:10.1145/375827.375835.

11 Christoph Berkholz and Jakob Nordström. Supercritical Space-Width Trade-offs for Resolution. In *Proceedings of the 43rd International Colloquium on Automata, Languages and Programming (ICALP '16)*, volume 55 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 57:1–57:14, July 2016. doi:10.4230/LIPIcs.ICALP.2016.57.

12 Archie Blake. *Canonical Expressions in Boolean Algebra*. PhD thesis, University of Chicago, 1937. doi:10.2307/2267634.

13 María Luisa Bonet and Nicola Galesi. Optimality of Size-Width Tradeoffs for Resolution. *Computational Complexity*, 10(4):261–276, December 2001. Preliminary version in *FOCS '99*. doi:10.1007/s000370100000.

14 Sam Buss and Jakob Nordström. Proof Complexity and SAT Solving. In Armin Biere, Marijn J. H. Heule, Hans van Maaren, and Toby Walsh, editors, *Handbook of Satisfiability*. IOS Press, 2020. Chapter to appear in the 2nd edition. Draft version available at https://www.math.ucsd.edu/~sbuss/ResearchWeb/ProofComplexitySAT/ProofComplexityChapter.pdf.

15 Matthew Clegg, Jeff Edmonds, and Russell Impagliazzo. Using the Groebner Basis Algorithm to Find Proofs of Unsatisfiability. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, pages 174–183, 1996. doi:10.1145/237814.237860.

16 William Cook, Collette Rene Coullard, and György Turán. On the Complexity of Cutting-Plane Proofs. *Discrete Applied Mathematics*, 18(1):25–38, November 1987. doi:10.1016/0166-218X(87)90039-4.

17 Susanna F. de Rezende, Jakob Nordström, Or Meir, and Robert Robere. Nullstellensatz size-degree trade-offs from reversible pebbling. In *34th Computational Complexity Conference, CCC 2019, July 18-20, 2019, New Brunswick, NJ, USA.*, pages 18:1–18:16, 2019. doi:10.4230/LIPIcs.CCC.2019.18.

18 Susanna F. de Rezende, Jakob Nordström, and Marc Vinyals. How Limited Interaction Hinders Real Communication (and What It Means for Proof and Circuit Complexity). In *Proceedings of the 57th Annual IEEE Symposium on Foundations of Computer Science (FOCS '16)*, pages 295–304, October 2016. doi:10.1109/FOCS.2016.40.

19 Nicola Galesi, Leszek Aleksander Kolodziejczyk, and Neil Thapen. Polynomial calculus space and resolution width. *Electronic Colloquium on Computational Complexity (ECCC)*, 26:52, 2019. URL: https://eccc.weizmann.ac.il/report/2019/052.

20 Nicola Galesi and Massimo Lauria. On the Automatizability of Polynomial Calculus. *Theory of Computing Systems*, 47(2):491–506, August 2010. doi:10.1007/s00224-009-9195-5.

21 Nicola Galesi and Massimo Lauria. Optimality of Size-Degree Trade-offs for Polynomial Calculus. *ACM Transactions on Computational Logic*, 12(1):4:1–4:22, November 2010. doi:10.1.1.703.6976.

22 Mika Göös and Toniann Pitassi. Communication Lower Bounds via Critical Block Sensitivity. *SIAM J. Comput.*, 47(5):1778–1806, 2018. doi:10.1137/16M1082007.

23 Trinh Huynh and Jakob Nordström. On the Virtue of Succinct Proofs: Amplifying Communication Complexity Hardness to Time-Space Trade-offs in Proof Complexity. In *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 - 22, 2012*, pages 233–248, 2012. doi:10.1145/2213977.2214000.

24 Russell Impagliazzo, Pavel Pudlák, and Jirí Sgall. Lower Bounds for the Polynomial Calculus and the Gröbner Basis Algorithm. *Computational Complexity*, 8(2):127–144, 1999. doi:10.1007/s000370050024.

25 Jan Krajíček. *Proof Complexity*, volume 170 of *Encyclopedia of Mathematics and Its Applications*. Cambridge University Press, March 2019. doi:10.1017/9781108242066.

26 Mladen Mikša and Jakob Nordström. A Generalized Method for Proving Polynomial Calculus Degree Lower Bounds. In *30th Conference on Computational Complexity, CCC 2015, June 17-19, 2015, Portland, Oregon, USA*, pages 467–487, 2015. doi:10.4230/LIPIcs.CCC.2015.467.

27 Jakob Nordström. A Simplified Way of Proving Trade-off Results for Resolution. *Information Processing Letters*, 109(18):1030–1035, August 2009. Preliminary version in ECCC report TR07-114, 2007. doi:10.1016/j.ipl.2009.06.006.

28 Alexander A. Razborov. A New Kind of Tradeoffs in Propositional Proof Complexity. *J. ACM*, 63(2):16:1–16:14, 2016. doi:10.1145/2858790.

29 Neil Thapen. A Tradeoff Between Length and Width in Resolution. *Theory of Computing*, 12(1):1–14, 2016. doi:10.4086/toc.2016.v012a005.

Smoothed Efficient Algorithms and Reductions for Network Coordination Games

Shant Boodaghians
University of Illinois at Urbana-Champaign, Urbana, IL, USA
boodagh2@illinois.edu

Rucha Kulkarni
University of Illinois at Urbana-Champaign, Urbana, IL, USA
ruchark2@illinois.edu

Ruta Mehta
University of Illinois at Urbana-Champaign, Urbana, IL, USA
rutamehta@cs.illinois.edu

Abstract

We study the smoothed complexity of finding pure Nash equilibria in Network Coordination Games, a PLS-complete problem in the worst case, even when each player has two strategies. This is a potential game where the *sequential-better-response* algorithm is known to converge to a pure NE, albeit in exponential time. First, we prove polynomial (respectively, quasi-polynomial) smoothed complexity when the underlying game graph is complete (resp. arbitrary), and every player has constantly many strategies. The complete graph assumption is reminiscent of perturbing *all parameters*, a common assumption in most known polynomial smoothed complexity results. We develop techniques to bound the probability that an (adversarial) better-response sequence makes slow improvements to the potential. Our approach combines and generalizes the local-max-cut approaches of Etscheid and Röglin (SODA '14; ACM TALG, '17) and Angel, Bubeck, Peres, and Wei (STOC '17), to handle the multi-strategy case. We believe that the approach and notions developed herein could be of interest in addressing the smoothed complexity of other potential games.

Further, we define a notion of a *smoothness-preserving reduction* among search problems, and obtain reductions from 2-strategy network coordination games to local-max-cut, and from k-strategy games (k arbitrary) to local-max-bisection. The former, with the recent result of Bibak, Chandrasekaran, and Carlson (SODA '18) gives an alternate $O(n^8)$-time smoothed algorithm when $k = 2$. These reductions extend smoothed efficient algorithms from one problem to another.

2012 ACM Subject Classification Theory of computation → Algorithmic game theory

Keywords and phrases Network Coordination Games, Smoothed Analysis

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.73

Related Version A full version of this paper is available at `https://arxiv.org/abs/1809.02280`.

Funding All three authors acknowledge support from NSF grant CCF 1750436.

Acknowledgements We would like to thank Pravesh Kothari for the insightful discussions in the initial stages of this work.

1 Introduction

Coordination games are a widely studied class of games, where players receive equal payoffs, and so are incentivized to coordinate. Network coordination games are a succinctly represented, natural multi-player extension of coordination games. The players simultaneously play multiple two-player coordination games, and receive the sum of their payoffs from these individual games. As a caveat, the players must choose the same strategy to play in all games.

These games naturally arise in various settings like social and biological networks [3, 32, 29], and have been extensively studied in various areas like Game theory and economics, Learning, Networks etc [25, 11, 20, 14, 2].

The natural dynamics in such a game imply that agents will change their strategy choices if this increases their payoff. Because these are coordination games, this also increases the total sum of payoffs. This sum is then a proxy for the progression towards an equilibrium, where no player can improve, hence is a *potential function* for the game, and the game becomes a *potential game*. When no player can benefit by deviating, or equivalently the potential function reaches a *local maximum*, this is a *pure Nash equilibrium*, and the standard search problem for most potential games is to find such an equilibrium.

Finding a pure Nash equilibrium in a network coordination game is complete for the class PLS (Polynomial Local Search) [12]. Although it is widely conjectured that PLS is unlikely to lie in P [6, 9, 38], problems in this class admit local-search algorithms [27], which have been observed to be empirically fast [27, 15, 18], but requiring exponential time in the worst case [40, 39]. To understand this discrepancy, we naturally turn to a beyond worst-case analysis technique called *smoothed analysis*, which "continuously interpolates between the worst-case and average-case analyses of algorithms," [41] (see Section 1.2 for a detailed discussion). Informally, we wish to show that adversarial instances are "scattered" in a probabilistic sense. We say that an algorithm is smoothed-efficient if it is efficient with high probability when all input parameters are perturbed by small random noise. This is one of the strongest performance metrics beyond worst-case performance. We ask the following:

▶ **Question.** *Can we design smoothed efficient algorithms for finding pure Nash equilibria for network coordination games?*

In this paper we answer the question in the affirmative. In particular, we obtain smoothed (quasi-)polynomial time algorithms to find pure Nash equilibria (PNE) in network-coordination games (NetCoordNash) with a constant number of strategies. We also introduce a notion for a *smoothness-preserving reduction*, and show that a special case of NetCoordNash admits such a reduction to local-max-cut, and the general case admits a reduction to local-max-bisection (see Section 2.2 for the problem definitions).

To the best of our knowledge, no smoothed efficient algorithm for a worst-case hard Nash equilibrium problem was known prior to this work, apart from the party affiliation games, the smoothed complexity of which directly follows from local-max-cut [23]. Similarly, to the authors' knowledge, no notion of smoothness-preserving reduction has been shown in the past, and we believe that such reductions are of independent interest.

Local-max-cut is a PLS-complete problem, where the goal is to find a cut in a graph that is maximal up to switching one vertex. In a recent series of results, the smoothed complexity of local-max-cut was shown to be first quasi-polynomial for arbitrary graphs [22], and then polynomial for complete graphs [1]. Both results and a recent (simultaneous) work [8] follow a common high-level framework: using anti-concentration bounds for linear combinations of random variables to argue that bad events are unlikely. Our analysis extends this high-level approach to NetCoordNash. However, local-max-cut is a special case of NetCoordNash where every player has two strategies and the matrix on every edge is off-diagonal (see Figure 1). To handle the extra complexity of NetCoordNash in general, we need to define novel bounds.

1.1 Our Results

A network coordination game is represented by an undirected *game graph* $G = (V, E)$, where the nodes are the players, and each player $v \in V$ simultaneously plays a two-player coordination game with each of its neighbors. If players have k strategies to choose from,

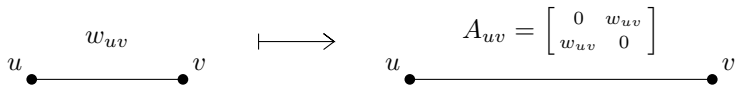


Figure 1 Local-max-cut to 2-strategy network coordination games: mapping of edge (u, v).

the game on each edge uv can be represented by a $k \times k$ payoff matrix. Once every player chooses a strategy, the payoff value for each edge is fixed, and players receive the sum of the payoffs from incident edges. The goal is to find a PNE of this game. We show the natural *better-response algorithm* converges quickly with high probability on perturbed instances.

Smoothed Analysis of NetCoordNash. A strategy profile is at equilibrium if no single player can gain by deviating while others' unilaterally. Better-response dynamics/algorithms are an iterative procedure where any player who benefits by deviating, does so, one at a time, until an equilibrium is reached. This need not converge in general, *e.g.* for a game of rock-paper-scissors, where the players would infinitely cycle through the three strategies. In our setting, however, the sum of payoffs of all players acts as a *potential function*, which increases with every better-response move (Section 2.1). Thus, starting from any initial choice of strategies, *better-response algorithm (BRA)* will converge to a PNE in network coordination games, since it can be shown that the potential function is bounded.

We show that the BRA is an efficient algorithm with probability $1 - 1/poly(n)$ for perturbed instances: when the payoff values are independently sampled from distributions with density bounded by ϕ, the runtime will be polynomial in ϕ and the input size with high probability. One may interpret ϕ as the inverse of the minimum allowed perturbation. The exact relation between perturbation size and running time are as follows:

▶ **Theorem 1.** *Let $G = (V, E)$ be a game graph for an instance of NetCoordNash, with $k \times k$ payoff matrices, whose entries are independently distributed, continuous, random variables, with densities $f_{u,v,i,j} : [-1, 1] \to [0, \phi]$. Let $n := |V|$. If G is a complete graph, then with probability $1 - (nk)^{-3}$, all valid executions of the BRA (even adversarial) will converge to a PNE in at most $(nk\phi)^{O(k)}$ steps, and the expected running time $(nk\phi)^{O(k)}$ as well.*

If G is arbitrary, all valid executions of the BRA, from all starting points, will converge to a PNE in at most $\phi \cdot (nk)^{O(k \log(nk))}$ steps with probability $1 - (nk)^{-2}$ over the payoff entries. Furthermore, the expected running time is also at most $\phi \cdot (nk)^{O(k \log(nk))}$.

The proof of this is discussed in Section 3. The polynomial running time requires the graph to be complete so that *all* parameters can be perturbed. This seems to be unavoidable as all known results on *polynomial* smoothed complexity so far, *e.g.,* linear-programming [41], local-max-cut [1], etc., require this.

The above performance guarantees are only (quasi-)polynomial in the input size for k constant. It is an open problem to improve this to general k. This can be achieved either by showing that local-max-bisection has polynomial smoothed complexity (see below), or by directly tightening the bounds in the proof presented in this paper (Section 3).

Smoothness-Preserving Reductions. Note that standard Karp reductions do not suffice to extend a smoothed efficient algorithm across problems. This is because, among other things, such a reduction needs to ensure that independently perturbed parameters of the original problem produce independent perturbations of all parameters in the reduced problem. In this work, we introduce a notion of a *smoothness-preserving reduction*, which to the knowledge of the authors, has not been studied prior to this work. We obtain two such reductions:

▶ **Theorem 2.** *NetCoordNash with 2×2 payoff matrices admits a weak smoothness-preserving reduction to the local-max-cut problem. Furthermore, NetCoordNash with $k \times k$ matrices for general k admits a weak smoothness-preserving reduction to the local-max-bisection problem. For both results, an instance of NetCoordNash with a general or complete game graph reduces to an instance of local-max-cut/bisection on a general or complete graph, respectively.*

The definition of weak reductions is given in Section 4, and a formal statement of the local-max-cut and -bisection problems is in Section 2.2. In this writeup, we provide an outline of the reductions in Section 4, and a formal description and proofs of correctness are given in the full version of this paper [10]. The first reduction, together with smoothed efficient algorithms for local-max-cut, gives alternate smoothed efficient algorithms for the $k = 2$ instance; namely, the result of [8] implies an $O(n^8)$ algorithm when the game graph is complete. For general network coordination games, the smoothed complexity of local-max-bisection is open, and so any conclusion on the complexity of NetCoordNash is conditional.

1.2 Related Work

The works most closely related to ours are [22] and [1], where the smoothed complexity of local-max-cut was first analyzed, and [8] which refined the analyses. As discussed above, local-max-cut is a special case of NetCoordNash, therefore the techniques do not immediately apply. Independently, [8] also obtained smoothed polynomial algorithms for local-max-3-cut on complete graphs, and quasi-polynomial algorithms in general for local-max-k-cut with constant k. Local-max-k-cut naturally reduces to NetCoordNash with $k \times k$ payoff matrices. However, our result does not subsume theirs as the reduction is not *smoothness preserving*.

Complexity of Equilibrium Computation. There has been extensive work on various potential games, equivalently congestion games (e.g., [35, 31, 36, 23]), capturing routing and traffic situations (e.g., [37, 25]), and resource allocation under strategic agents (e.g., [26, 24]). A potential function ensures that these games always have a pure NE [35]. Finding pure NE is typically PLS-complete in the worst case [23, 12]. Our approach provides tools to prove smoothed analysis results for such games.

For the general games, NE computation is PPAD-complete [17, 13], even (1/poly)-additive approximation. This latter result also implies smoothed complexity does not lie in P unless RP = PPAD [13]. Towards average case analysis, Bárány, Vempala, and Vetta [5] showed that a game picked uniformly at random has a NE with support size 2 for both the players with inverse-logarithmic probability The average case complexity of a random potential game was shown to be polynomial in the number of players and strategies by Durand and Gaujal [19].

Smoothed Analysis. The work of Spielman and Teng [41] introduced the smoothed analysis framework to study the empirical performance of the Simplex method for linear programming. They showed that introducing independent random perturbations to any given (adversarial) LP instance, ensures that Simplex terminates fast with high probability, with polynomial dependence on the inverse of the magnitude of perturbation. Performance on such perturbed instances has since been known as *smoothed complexity* of the problem – one of the strongest guarantees one can hope for *beyond worst-case* analysis. In the past decade and a half, much work has sought to obtain smoothed efficient algorithms when worst-case efficiency seems infeasible [16, 7, 30, 34, 4, 21, 22, 1], including for integer programming, binary search trees, iterative-closest-point (ICP) algorithms, the 2-OPT algorithm for the Traveling Salesman problem (TSP), the knapsack problem, and the local-max-cut problem.

2 Preliminaries: Game Model and Smoothed Analysis

In what follows, the set $\{1, 2, \ldots, k\}$ is denoted as $[k]$, and $\langle \cdot, \cdot \rangle$ denotes inner product.

2.1 Nash Equilibria in Network Coordination Games

A two-player game, where each player has finitely many strategies to choose from is given by two payoff matrices A and B. Assume without loss of generality that both players have k strategies, and thus the matrices are $k \times k$. It is called a *coordination game* if $A = B$.

A *network coordination game* is a multi-player extension of coordination games. The game is specified by an underlying undirected graph $G = (V, E)$, where the nodes are players, and each edge represents a two-player coordination game between its endpoints. It is a *k-network coordination game* if each player has k strategies. For disambiguation, we represent the payoff values as an $|E|k^2$-dimensional vector A, and denote as $A((u, i)(v, j))$ the payoff that players u and v get for the game-edge $uv \in E$, when u chooses strategy i, and v, strategy j. As Nash equilibria are invariant to shifting and scaling of the payoffs, assume without loss of generality that every entry of A is contained in $[-1, 1]$.

Potential Function. Below we show that it suffices to only consider pure strategies, *i.e.* players needn't randomize. Let n be the number of players; a *strategy profile* is a vector $\boldsymbol{\sigma} \in [k]^n$, assigning to each player a strategy in $[k]$. The payoff to player u is given by

$$\text{payoff}_u(\boldsymbol{\sigma}) := \sum_{v:\, uv \in E} A((u, \sigma_u)(v, \sigma_v))$$

Define the *potential function* $\Phi : [k]^n \to \mathbb{R}$ to be the sum of all payoffs. Formally,

$$\Phi(\boldsymbol{\sigma}) := \sum_{(u,v)\in E} A((u, \sigma_u), (v, \sigma_v)) = \tfrac{1}{2} \sum_{u \in V} \text{payoff}_u(\boldsymbol{\sigma}) \tag{1}$$

The potential function is of interest since it captures the possible improvements to all players' payoffs in the following sense [12]: if player u changes their strategy, $\Phi(\boldsymbol{\sigma})$ and $\text{payoff}_u(\boldsymbol{\sigma})$ vary by the same amount. Formally, for all $u \in V$, $\sigma_u, \sigma'_u \in [k]$, and $\boldsymbol{\sigma}_{-u} \in [k]^{n-1}$

$$\Phi(\sigma_u, \boldsymbol{\sigma}_{-u}) - \Phi(\sigma'_u, \boldsymbol{\sigma}_{-u}) \;=\; \text{payoff}_u(\sigma_u, \boldsymbol{\sigma}_{-u}) - \text{payoff}_u(\sigma'_u, \boldsymbol{\sigma}_{-u})$$

where $\boldsymbol{\sigma}_{-u} \in [k]^{n-1}$ denotes the strategy profile on $V \setminus u$. Network coordination games are termed *potential games* because they admit such a potential function. As a consequence, they must admit pure Nash equilibria [35].

Nash Equilibrium and Better-Response Algorithm (BR alg., or BRA). At a Nash equilibrium (NE), no player gains by deviating unilaterally.

$$\textit{NE:} \quad \forall u \in V, \quad \text{payoff}_u(\sigma_u, \boldsymbol{\sigma}_{-u}) \geq \text{payoff}_u(\sigma'_u, \boldsymbol{\sigma}_{-u}), \quad \forall \sigma'_u \in [k]$$

Such a $\boldsymbol{\sigma}$ is called *pure* NE (PNE) as every player is playing a deterministic strategy. By the discussion above, $\boldsymbol{\sigma}$ is a PNE if and only if it is a local maximum for Φ, where $\boldsymbol{\sigma}'$ is in the local neighbourhood of $\boldsymbol{\sigma}$ when they differ in exactly one entry. A deviation for one player is termed a *better-response (BR) move* if their individual payoff strictly increases. Note that if $\boldsymbol{\sigma}'$ is a BR deviation from $\boldsymbol{\sigma}$, differing in a single player, then $\Phi(\boldsymbol{\sigma}') > \Phi(\boldsymbol{\sigma})$. The *better-response algorithm* (BRA) repeatedly makes better-response moves, increasing the Φ value in every step. The terminating point has to be a local maximum of Φ, and thereby a PNE. Since Φ may only take k^n different values, this procedure must terminate at a PNE.

2.2 Smoothed Analysis and Reductions

The notion of smoothed analysis was introduced by Spielman and Teng [41] to bridge the gap between average- and worst-case analysis. The parameters of the problem are perturbed by some small noise, and the performance is measured as a function of the perturbation size. We present first a formal definition of smoothed-efficient algorithms in our setting.

▶ **Definition 3** (Independent distributions with bounded density). *Let X be a random vector in $[-1,1]^d$. We say it is* independently distributed with density bounded by ϕ *if the entries are independently distributed, and the p.d.f. for the i-th entry is a function $f_i : [-1,1] \to [0,\phi]$. Observe that the joint distribution on X has p.d.f. upper-bounded by ϕ^d.*

Intuitively, a bounded-density X is "spread" by at least $1/\phi$. Running-time bounds will be defined as a function of ϕ. We define here polynomial smoothed complexity in our setting:

▶ **Definition 4** (Polynomial Smoothed Complexity). *Let $\mathcal{P}$ be a search problem, whose instances consist of some structural information D – e.g. a graph – and some real-valued information X – e.g. edge weights. We say $\mathcal{A}$ is a* smoothed efficient algorithm *for $\mathcal{P}$ if, $\mathcal{A}(D,X)$ returns a correct solution with probability 1, and there exist constants $c, c' > 0$ such that whenever $X \in \mathbb{R}^d$ is an independently distributed random vector with density bounded by ϕ,*

$$\max_{D} \Pr_{X} \left[\textit{running time of } \mathcal{A} \textit{ on } (D,X) \ \geq (d \cdot |D| \cdot \phi)^c\right] \ \leq (d \cdot |D|)^{-c'} \ .$$

$\mathcal{P}$ is said to have polynomial smoothed complexity *if a smoothed efficient algorithm exists. It has* quasi-poly smoothed complexity *if the above holds for running time $(d \cdot |D| \cdot \phi)^{O(\log(d \cdot |D|))}$.*

An algorithm is smoothed-efficient in expectation if the expected running time over a worst-case choice of ϕ-bounded distributions is (quasi-)polynomial in d, ϕ, and $|D|$; a problem $\mathcal{P}$ is said to have (quasi-)polynomial smoothed complexity in expectation similarly.

Local-max-cut and -bisection. In this paper, we define *smoothness preserving reductions* which allow the extension of smoothed-complexity results, as defined above, from one problem to another. Namely, we obtain reductions to the local-max-cut and -bisection problems. These problems are defined as FLIP and SWAP respectively in [40]. Given a weighted graph $G = (V, E)$, local-max-cut is the problem of finding a cut which is maximal up to flipping one vertex across the cut, and local-max-bisection is the problem of finding a balanced cut of the nodes into two sets of equal size, whose cut value is maximal up to swapping a pair of nodes across the cut. Both problems are shown to be PLS-hard in [40], and the smoothed complexity of local-max-cut has been studied at length, as discussed in the introduction.

3 Smoothed Performance of the BR algorithm

This section is a partial proof of Theorem 1. All missing details may be found in the full version [10]. Recall that Theorem 1 stated that for ϕ-bounded random variables, the BRA converges in $(nk\phi)^{O(k)}$ steps with probability $1 - (nk)^{-3}$ on complete graphs, and $\phi \cdot (nk)^{O(k \log(nk))}$ steps with probability $1 - (nk)^{-2}$ over the payoff entries. Only the payoff values are randomized, the BR moves may be chosen adversarially after sampling.

Recall that a profile $\boldsymbol{\sigma}$ is a PNE if and only if it is a local maximum of the potential Φ. Note that Φ may only take values in the interval $[-n^2, n^2]$, since the payoffs are in $[-1, 1]$. It suffices then to show that with high probability, *every* linear-length sequence of BR moves has significant increase in Φ. We show the following:

▶ **Theorem 5.** *Let $G = (V, E)$ be a game graph, with random payoff vector A, and $\boldsymbol{\sigma}^0 \in [k]^n$ be an arbitrary strategy profile. With probability $1 - (nk)^{-2}$ over the values of A, all BR sequences of length at least $2nk$, initiated at any choice of $\boldsymbol{\sigma}^0$, must have at least one step in which the potential increases by $\epsilon = \phi^{-1}(2n^2k^3)^{-20k\log(nk)}$. If G is a complete graph, then with probability $1 - (nk)^{-3}$, all BR sequences of length at least $2nk$, will have at least one step increasing by $\epsilon' = (20\phi^2n^3k^3)^{-4k-4}$.*

From the above discussion, this implies that the BRA must terminate in $\phi(nk)^{O(k\log(nk))}$ steps with probability $1-(nk)^{-2}$ in general, and $(\phi nk)^{O(k)}$ steps with probability $1-(nk)^{-3}$ on complete game graphs, implying Theorem 1. We proceed with a proof of Theorem 5.

The high level approach of this paper, and also that of [22, 1, 8], is as follows: Express the increase in potential as a linear combination of the payoff values, and conclude Theorem 5 via an anti-concentration inequality and a union bound. Each step of the BRA is some player u, deviating to a new $\sigma \in [k]$, denoted as the (player,strategy) pair (u, σ). Thus, an execution of the BRA is fully specified by a sequence of pairs $S = (u_1, \sigma_1), (u_2, \sigma_2), \ldots$, along with an initial strategy vector $\boldsymbol{\sigma}^0 \in [k]^n$. The strategy profile at time t is given by $\boldsymbol{\sigma}^t := (\sigma_t, \boldsymbol{\sigma}^{t-1}_{-u_t})$.

We wish to control the value of $\Phi(\boldsymbol{\sigma}^t) - \Phi(\boldsymbol{\sigma}^{t-1})$ as a function of the payoff values. To this end, define the *potential-change indicator matrix* for a BR sequence as follows:

▶ **Definition 6.** *For any fixed BR sequence S of length ℓ, let $L(S, \boldsymbol{\sigma}^0) := \{\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \ldots, \boldsymbol{\lambda}_\ell\}$, where $\boldsymbol{\lambda}_t \in \{-1, 0, 1\}^{(|E|\times k^2)}$, for all t. The entries of $\boldsymbol{\lambda}_t$ are indexed by indices of payoff matrix entries, denoted $((v, i)(w, j))$. The values of its entries are chosen as follows:*

$$\boldsymbol{\lambda}_t((v,i)(w,j)) := \begin{cases} 1 & \text{if: } u_t \in \{v,w\} \text{ and } \sigma^t_v = i \text{ and } \sigma^t_w = j. \\ -1 & \text{if: } u_t \in \{v,w\} \text{ and } \sigma^{t-1}_v = i \text{ and } \sigma^{t-1}_w = j. \\ 0 & \text{otherwise.} \end{cases}$$

Each entry denotes whether the payoff values remain in the payoff (0), get added to the total payoff $(+1)$, or removed (-1). This set of vectors – or equivalently the matrix whose rows consist of the $\boldsymbol{\lambda}_t$'s – is termed the potential-change indicator matrix of a sequence.

The arguments $S, \boldsymbol{\sigma}^0$ are omitted if they are clear from context. Observe $\Phi(\boldsymbol{\sigma}^t) - \Phi(\boldsymbol{\sigma}^{t-1}) = \langle \boldsymbol{\lambda}_t, A\rangle$, where A is the vector of payoff values, so the product LA represents the sequence of changes in Φ along an execution of the BRA. Bounding the probability of $LA \notin [0, \epsilon]^\ell$ for all sequences of length $\ell \geq 2nk$ then implies Theorem 5. We use the following.

▶ **Lemma 7** ([33])**.** *Let $X \in \mathbb{R}^d$ be a random vector such that the joint probability on any $a \leq d$ coordinates of X is upper-bounded by ϕ^a at all points. Let M be a rank r matrix in $\eta\mathbb{Z}^{\ell\times d}$, i.e. all entries are multiples of η. Then the joint density of the vector MX is bounded by $(\phi/\eta)^r$, and for any given $b_1, b_2, \ldots, b_\ell \in \mathbb{R}$ and $\epsilon > 0$,*

$$\Pr\Big[MX \in [b_1, b_1+\epsilon] \times \cdots \times [b_\ell, b_\ell + \epsilon]\Big] \leq (\phi\epsilon/\eta)^r \tag{2}$$

Observe that if X is a vector whose entries are independently distributed, and each X_i has probability density bounded by ϕ, then the joint distribution over any ℓ coordinates of X has density bounded by ϕ^ℓ. This statement is a generalization of a lemma from [33], but the proof therein easily extends.

To conclude Theorem 5, we first show that L has large enough rank, then apply the above lemma with $M = L(S, \boldsymbol{\sigma}^0)$ and $X = A$, and finally take a union bound over the choice of S and $\boldsymbol{\sigma}^0$. The following parameters are introduced for clarity of exposition.

▶ **Definition 8** (Active, Inactive, Repeating, and Non-Repeating players.)**.** *Let S be a BR sequence, then player u is said to be* active *if she appears in the sequence, and* inactive *otherwise. An active player u is said to be* repeating *if there exists some strategy i such that (u, i) appears at least twice in S, or if (u, σ_u^0) appears in S at all. An active player who is not repeating is said to be* non-repeating*. We introduce the following notation:*

$p(S)$	*number of active players in S,*	$d(S)$	*number of distinct (u, i) moves in S,*
$p_1(S)$	*num. non-repeating players in S,*	$d_1(S)$	*distinct moves by non-rept. players in S,*
$p_2(S)$	*number of repeating players in S,*	$q_0(S)$	*number of distinct (u, σ_u^0) moves in S.*

Observe that $p = p_1 + p_2$, $k \cdot p \geq d \geq p$, $k \cdot p_1 \geq d_1 \geq p_1$, and $q_0 \leq p_2$. We will often use the quantity $d(S) - q_0(S)$, which is the number of "new" strategies played by the players.

3.1 Inactive Players, Rank Bounds, and Union Bounds

As discussed above, the goal is to show that $L(S)$ has large rank, and apply Lemma 7, taking a union bound over all possible choices of S and $\boldsymbol{\sigma}^0$. Naïvely, there are $k^n(nk)^\ell$ choices for sequences of length ℓ. However, if $p(S) \ll n$, the rank bound cannot exceed $d(S) \leq k \cdot p(S)$ in our model, which does not match the union bound. To circumvent this issue, we modify the matrix L in two different ways, for the cases $p_1(S) \geq p_2(S)$ and $p_2(S) \geq p_1(S)$, respectively. This case analysis is similar to the proofs in [1, 8], however these two papers each use only one of the two constructions, whereas our analyses require both, as the large strategy space allows for more complex interactions between the rows of L, and each construction only allows bounds of one kind.

3.1.1 Control by rounding.

The first modification to L builds on a technique of [1]. While the construction is valid for arbitrary graphs, the rank bounds hold only for complete graphs. Let $V_0 \subset V$ be the set of inactive players, and V_1 the active players. For any $u \in V_1$ and i fixed, all $((u, i)(w, \sigma_w^0))$ rows of L for $w \in V_0$ are identical, up to flipping a row's signs, since w's strategy remains unchanged. Therefore, in the inner product $\langle \boldsymbol{\lambda}_t, A \rangle$, these $((u, i), (w, \sigma_v^0))$ terms are added or subtracted together, and we may simply "guess" this value approximately, and take a union bound on the guesses, rather than guessing strategy choices. This is formalized below.

The quantity of interest is the change in $\Phi(\boldsymbol{\sigma}^t)$ over t, so it is equivalent to instead consider $\Phi(\boldsymbol{\sigma}^t) - \Phi(\boldsymbol{\sigma}^0)$, a constant shift. Let $\widetilde{A}_{uv}^{t-0} = A((u, \boldsymbol{\sigma}_u^t)(v, \boldsymbol{\sigma}_v^t)) - A((u, \boldsymbol{\sigma}_u^0)(v, \boldsymbol{\sigma}_v^0))$.

$$\begin{aligned}\Phi(\boldsymbol{\sigma}^t) - \Phi(\boldsymbol{\sigma}^0) &= \sum_{u,v \in V} A((u, \boldsymbol{\sigma}_u^t)(v, \boldsymbol{\sigma}_v^t)) - A((u, \boldsymbol{\sigma}_u^0)(v, \boldsymbol{\sigma}_v^0)) \\ &= \sum_{u,v \in V_1} \widetilde{A}_{uv}^{t-0} + \sum_{w,w' \in V_0} \widetilde{A}_{ww'}^{t-0} + \sum_{u \in V_1} \sum_{w \in V_0} \widetilde{A}_{uw}^{t-0}\end{aligned}$$

For $w \in V_0$, $\boldsymbol{\sigma}_w^t = \boldsymbol{\sigma}_w^0$, so the "$\widetilde{A}_{ww'}^{t-0}$" terms are 0. Furthermore, the rightmost terms are in fact constants, depending only on σ_u. Let then $C(u, \sigma) := \sum_{w \in V_0} A((u, \sigma)(w, \boldsymbol{\sigma}_w^0)) - A((u, \boldsymbol{\sigma}_u^0)(w, \boldsymbol{\sigma}_w^0))$. Note $C(u, \boldsymbol{\sigma}_u^0) = 0$. The above sum is therefore equivalent to

$$\Phi(\boldsymbol{\sigma}^t) - \Phi(\boldsymbol{\sigma}^0) \;=\; \sum_{u,v \in V_1} \widetilde{A}_{uv}^{t-0} \;+\; 0 \;+\; \sum_{u \in V_1} C(u, \sigma_u^t)$$

For $p(S)$ fixed, we need only take a union bound over the nk^ℓ choices of BR sequence, $k^{p(S)}$ choices of initial strategy, and the $d(S) - q_0(S)$ different C values for each $(u, \sigma \neq \sigma_u^0)$. We will round these to the nearest multiple of ϵ, leaving $2n/\epsilon$ choices for each. With this, we have a union bound of size $k^{p(S)}(nk)^\ell(2n/\epsilon)^{d(S)-q_0(S)}$.

Critical Subsequences and Rank Bounds. It remains to argue that $L(S, \boldsymbol{\sigma}^0)$ has sufficiently large rank bounds. The statements below are presented informally, in the interest of space. See the full version for complete details [10]. To this end, we define *critical subsequences* as follows, based on the definition of critical block from [1]:

▶ **Definition 9** (Critical Subsequence). *Let B be a contiguous subsequence of S. If $\ell(B) \geq 2\big(d(B) - q_0(B)\big)$, but for every $B' \subseteq B$, $\ell(B') < 2\big(d(B') - q_0(B')\big)$, we say that B is* critical. *Note that a return move – i.e. q_0-type move – for a subsequence B which starts at time t_B is a move $(u, \boldsymbol{\sigma}_u^{t_B})$, as opposed to a $(u, \boldsymbol{\sigma}_u^0)$ move.*

Observe that for any BR sequence S of length $2nk$, $\ell(S) \geq 2(d(S) - q_0(S))$, and so a maximal (up-to-inclusion) subsequence satisfying $\ell \leq 2(d - q_0)$ will be critical. Thus, any sequence of length $2nk$ must contain a critical subsequence B, with $\ell(B) = 2(d(B) - q_0(B))$. Critical subsequences are key to the analysis, since their length and number of distinct moves are perfectly correlated. This will be used in conjunction with the notion of *separated blocks*.

▶ **Definition 10** (Separated Blocks). *Fix a BR sequence S. For any active, non-repeating, player u, let T_u be the set of indices where the moving player is u, and $T := \{t_1 < t_2 < \cdots < t_m\}$ be the union of the T_u's. Note that $|T| = d_1(S)$. Let S_i for $0 \leq i \leq m$ be the subsequences of S from time t_i to t_{i+1} excluding boundaries, respectively, where $t_0 = 0$ and $t_{m+1} = |S|$. We say these S_i's are the* separated blocks *of S.*

The separated blocks allow us to isolate large-rank submatrices and combine their ranks. Let w be any inactive player, and (u_i, σ_i) be the non-repeating move which begins separated block i for all i. Then, up to column permutations, the submatrix with rows from T and columns indexed by $((u_i, \sigma_i)(w, \sigma_w))$ is upper-triangular with a non-zero diagonal. The diagonal is necessarily nonzero because the graph is complete. Furthermore, for a separated block S' which begins at move (u_i, σ_i), the submatrix of rows from S' and columns $((v, \sigma_v)(u_i, \sigma_i))$ also form an upper-triangular matrix with non-zero diagonals, where the columns are sorted by the first appearance of v, σ_v in S'. Finally, if we sort the rows of T first, then by chronological order of separated blocks, we can form a block-upper-triangular submatrix with non-zero diagonal terms, of size $d_1(S) + \sum_{S' \text{ separated}}(d(S') - q_0(S'))$. As above, graph completeness is necessary for the diagonal to be nonzero. This submatrix is block-upper-triangular because $((\cdot, \cdot), (u_i, \sigma_i))$ rows are all 0 before u_i first enters σ_i, since u_i is non-repeating. Assuming that the overall sequence S is critical, this gives

$$rank(L) \;\geq\; d_1(S) + \sum_{S' \text{ sep.}} d(S') - q_0(S') \;>\; \tfrac{1}{2}d_1(S) + \tfrac{1}{2}d_1(S) + \sum_{S' \text{ sep.}} \tfrac{1}{2}\ell(S')$$

However, $\ell(S) = d_1(S) + \sum_{S' \text{ sep.}} \ell(S')$, so we have $rank(L) \geq \frac{1}{2}d_1(S) + \frac{1}{2}\ell(S)$. By criticality, $\ell(S) = 2(d(S) - q_0(S))$, which gives a rank bound of $\frac{1}{2}(d_1(S) + \ell(S)) = \frac{1}{2}d_1(S) + d(S) - q_0(S)$.

Combining Rank and Union bounds. Recall that we have shown, for any BR sequence of length $2nk$, it must contain a critical subsequence B. Assuming B has length ℓ, there are $k^{p(B)}(nk)^\ell$ choices for the (modified) matrix $L(B, \boldsymbol{\sigma}^0)$, and $(4n/\epsilon)^{d(B) - q_0(B)}$ choices for the approximate $C(u, \sigma)$ values. The $C(u, \sigma)$'s are rounded to the nearest ϵ, so the true improvements lie in $(0, \epsilon)$ if the approximated improvements lie in $(-\epsilon, 2\epsilon)$, since there can be at most two C terms in each change, and rounding introduces error at most $\epsilon/2$.

We have shown that $L(B, \boldsymbol{\sigma}^0)$ has rank at least $\frac{1}{2}d_1(B) + d(B) - q_0(B)$. Applying Lemma 7, we conclude that the probability that every critical subsequence of a valid BR sequence of length at least $2nk$ has all improvements in $(0, \epsilon)$ is upper-bounded by

$k^{p(B)}(nk)^{\ell}(4n/\epsilon)^{d(B)-q_0(B)}(3\phi\epsilon)^{d_1(B)/2+d(B)-q_0(B)}$. Recall $d_1(B) \geq p_1(B)$, and assume $p_1(B) \geq p_2(B) \implies d_1(B) \geq p_1(B) \geq \frac{1}{2}p(B)$. Also, $d(B) - q_0(B) \leq k \cdot p(B)$, and recall $\ell = 2(d(B) - q_0(B))$. Thus the probability that all improvements are small, for complete graphs, when $p_1(B) \geq p_2(B)$ is at most $\left((20n^3k^3\phi^2)^k\epsilon^{1/4}\right)^{p(B)}$. We later combine this bound with one for the case $p_2(B) \geq p_1(B)$, and take a union bound over the choice of $p(B)$.

3.1.2 Control by cyclic sums

The second rank bound argument is more intricate, and is loosely based on a construction in [22]. The bounds proved here hold for arbitrary graphs. We construct a new matrix Q whose columns lie in the span of L, which cancels the contributions of inactive players, but still captures improvement. Let (u, σ) be some move which appears twice in S, or suppose some (u, σ_u^0) appears in S. Let τ_0 be the index of the first occurrence of (u, σ) in the BR sequence ($\tau_0 = 0$ in the latter case), and let $\tau_1, \tau_2, \ldots$ be all subsequent appearances of $(u, \cdot)$ in the sequence. Suppose τ_m is the second occurrence of (u, σ) in the BR sequence. Then we let $\boldsymbol{q}_{u,\sigma} := \sum_{j=1}^{m} \boldsymbol{\lambda}_{\tau_j}$, noting that the τ_0 is omitted. Let $Q(S, \boldsymbol{\sigma}^0)$ be the matrix whose columns consist of the $\boldsymbol{q}_{u,\sigma}$'s. Observe, for any inactive player w,

$$\boldsymbol{q}_{u,\sigma}\big((u,\sigma')(w,\sigma_w^0)\big) \;=\; \sum_{j=1}^{m} \boldsymbol{\lambda}_{\tau_j}\big((u,\sigma')(w,\sigma_w^0)\big) \;=\; \sum_{j=1}^{m} \mathbb{I}[\sigma_u^{\tau_j} = \sigma'] - \mathbb{I}[\sigma_u^{\tau_j - 1} = \sigma'] \;=\; 0$$

Thus, the $\boldsymbol{q}_{u,\sigma}$'s are all 0 in entries corresponding to edges with inactive players, and therefore the Q matrix does not depend on the initial strategies of the inactive players. Thus, to union bound the possible Q matrices, it suffices to fix the initial strategy of only the active players and the BR sequence. Furthermore, $L \cdot A \in [0, \epsilon]^{\ell} \implies Q \cdot A \in [0, \ell\epsilon]^{d-d_1}$, so it suffices to apply Lemma 7 on the matrix Q, with only polynomial blowup, since $\ell \leq O(nk)$.

Fixing a critical subsequence B, there are at most $(nk)^{\ell(B)}$ choices of the BR sequence, and $k^{p(B)}$ choices of initial strategy profiles for the active players.

Rank Bounds. It remains to show that Q has large rank relative to this smaller union bound. Since we have eliminated inactive players, the high-rank submatrix will consist of interactions between active players. As in [22], construct an auxiliary *directed* graph $G' = (V, E')$, where V is the set of players, and E' is constructed as follows: for any repeating move (u, σ), add the directed edge (u, w) for *every* $w \in V$, such that $\exists \sigma' \in [k] : \boldsymbol{q}_{u,\sigma}\big((u,\sigma)(w,\sigma')\big) \neq 0$. Note that u and w must be connected in the game graph, and we do not need completeness.

Let $P_2 \subseteq V$ be the set of repeating players, and note that they all have non-zero out-degree, since $\boldsymbol{q}_{u,\sigma} \neq \mathbf{0}$ as this would contradict the fact that the sequence is improving. Repeatedly perform the following: choose $r_1 \in P_2$, and let T_1 be the BFS arborescence rooted at r_1 which spans all nodes reachable from r_1 in G'. Delete $V(T_1)$ from G' and repeat, picking an arbitrary root vertex $r_2 \in P_2 \setminus V(T_1)$. Stop when every vertex of P_2 is covered by some arborescence. For all i, let T_i^0 and T_i^1 be a partition of the nodes of T_i which are of even or odd distance from the root, respectively. Let P_i' be the larger of $V(T_i^0) \cap P_2$ and $V(T_i^1) \cap P_2$, and let $P_2' := \bigcup_{i=1}^{\infty} P_i'$, noting that $|P_2'| \geq |P_2|/2 = p_2(S)/2$. We claim that the collection $\mathcal{V} := \{\boldsymbol{q}_{u,\cdot} : u \in P_2'\}$ is linearly independent.

For every $u \in P_2'$, either u was an internal node to the arborescence that covers it, or a leaf. If it is internal, we may associate to u any of its children in the arborescence, which are not contained in P_2', by definition. Let w be the chosen child, then the only $\mathcal{V}$ vectors which have nonzero $((u,\cdot)(w,\cdot))$ entries are the $\boldsymbol{q}_{u,\cdot}$ vectors.

Conversely if u is a leaf, then its out-neighbours must be in previously constructed arborescences. Let w be any such neighbour, then $\boldsymbol{q}_{w,\cdot}$ can not contain a non-zero $((u,\cdot)(w,\cdot))$ entry, as otherwise u would have been in w's arborescence. Therefore, $\boldsymbol{q}_{u,\cdot}$ is the only vector in $\mathcal{V}$ to contain a nonzero $((u,\cdot)(w,\cdot))$ entry. Observe that these matrix entries are necessarily nonzero even if the graph is not complete. The above discussion demonstrates a $|\mathcal{V}| \times |\mathcal{V}|$ diagonal submatrix of $\mathcal{V}$, which implies Q has rank at least $|\mathcal{V}| \geq p_2(S)/2$.

Combining Rank and Union Bounds. This section considers the complete graph case. These bounds will be extended to general graphs in the next section. Section 3.1.1 gives bounds for the case $p_1(B) \geq p_2(B)$, and so we restrict our attention to the case $p_2(B) \geq p_1(B) \implies p_2(B) \geq p(B)/2$.

For any fixed critical subsequence B, we have argued that the probability that $L(B, \boldsymbol{\sigma}^0)$ has all entries in $(0,\epsilon)$ is at most the probability that $Q(B, \boldsymbol{\sigma}^0)$ has all entries in $(0, \ell\epsilon)$. By Lemma 7 and the above rank bounds, this occurs with probability at most $(\ell\phi\epsilon)^{p_2(B)/2}$. Recall that by criticality, $\ell = 2(d(B) - q_0(B)) \leq 2k \cdot p(B)$. Thus, with the above union bound, the probability that all critical subsequences with $p(B)$ active players and $p_2(B) \geq p_1(B)$ have bad improvements is at most $k^{p(B)}(nk)^{2kp(B)}(2k \cdot p(B) \cdot \epsilon\phi)^{p(B)/4} \leq \left(2(nk)^{2k}k^{5/4}(n\phi\epsilon)^{1/4}\right)^{p(S)}$.

3.2 (Quasi-)Polynomial Smoothed Complexity for NetCoordNash

Complete Game Graphs. From Sections, 3.1.1 and 3.1.2, the probability that *all* valid BR sequences of length at least $2nk$ have all improvements in $(0,\epsilon)$ is at most

$$\sum_{p=1}^{n} \left((20n^3k^3\phi^2)^k\epsilon^{1/4}\right)^p + \left(2(nk)^{2k}(nk^5\phi\epsilon)^{1/4}\right)^p$$

by simply taking a union bound over the value of $p(B)$, and whether $p_1(B) \geq p_2(B)$ or the converse hold. Setting $\epsilon = (20\phi^2n^3k^3)^{-4k-4}$, we conclude that this probability is at most $1/(20\phi^2n^3k^4 - 1)$, which concludes the complete-graph portion of Theorem 5 noting that $\phi > \frac{1}{2}$. We omit calculation details and handling edge-cases where all players are active. These are included in the full version [10].

General Game Graphs. For general game graphs, we will use the cyclical-sum construction, as it does not require completeness. Note that it was phrased in terms of critical subsequences, but never used this property. However, the cyclic-sum construction only gives bounds in terms of $p_2(S)$. As in [22], we use the following lemma:

▶ **Lemma 11** ([22], Lemma 3.4). *Any valid BR sequence of length $5nk$ contains a contiguous subsequence S' with $|S'|/(5\log(nk))$ repeating moves (pairs).*

This implies that any $5nk$-length BR sequence S contains a subsequence S' with $p_2(S') \geq |S'|/(5\log(nk))$. Therefore, the probability that all $5nk$-length sequences have improvements in $(0,\epsilon)$ is at most $\sum_{\ell=1}^{5nk} k^n(nk)^\ell(\phi\epsilon)^{\ell/10\log(nk)}$. Setting $\epsilon = \phi^{-1}(2n^2k^3)^{-20k\cdot\log(nk)}$, the probability is at most $1/(nk)^2$. This concludes the general-graph portion of Theorem 5. As above, the details of calculation are omitted.

Performance in Expectation. The performance analysis of BRA in expectation is discussed in the full-version. It is derived as a consequence of the with-high-probability bounds, integrating over the choice of ϵ.

4 Smoothness-Preserving Reduction to Local-Max-Cut and -Bisection

In this section, we refine standard Karp reductions to define *smoothness preserving reductions*, and outline the two reductions of Theorem 2. Recall from Section 2.2 that an algorithm is said to be smoothed-efficient if, on applying independent random perturbations to all inputs of an adversarially chosen instance, the algorithm runs in time polynomial in the input size and inverse perturbation size, with high probability.

▶ **Definition 12** (Strong and Weak Smoothness-Preserving Reductions)**.** *A* weak (randomized) smoothness-preserving reduction *from a search problem $\mathcal{P}$ to problem $\mathcal{Q}$ is defined by poly-time computable functions f_1 and f_2, a full-row-rank, integer, matrix M with polynomially bounded entries, a constant $\eta \geq 1/poly$, and a real probability space $\Omega \subseteq \mathbb{R}^d$; such that the following holds: For any $I = (D, X) \in \mathcal{P}$ and $R \in \Omega$, $J = (f_1(D), \eta M(X \circ R))$ is an instance of $\mathcal{Q}$ whose solutions $\boldsymbol{\sigma}$ map to solutions $f_2(\boldsymbol{\sigma})$ for I. Here, $\circ$ denotes concatenation.*

We require that $|f_1(D)|$, the dimension of R, and the size of M, be polynomial in $|I|$; that the probability density of the entries of R be polynomial in $|I|$ and the density bound on X; and that the entries of R be independently distributed.

If M is a diagonal matrix, then this is a strong *smoothness-preserving reduction.*

The R variables may seem superfluous at first, but are included to ensure that M has full-rank. A key fact to the proof of Lemma 7, the anti-concentration bound at the heart of this paper and previous local-max-cut papers, is the following:

▶ **Proposition 13** ([33])**.** *Let $X \in \mathbb{R}^d$ be a random vector such that the joint probability on any $a \leq d$ coordinates of X is upper-bounded by ϕ^a at all points, and let $M \in \mathbb{R}^{\ell \times d}$ be full-rank, with entries which are multiples of η, for $\ell \leq d$. Then the random variable $Y := MX$ also has bounded joint density $f_Y(y) \leq (\phi/\eta)^\ell$ for all $y \in \mathbb{R}^d$.*

Thus, if the entries of X and R have bounded density, and $|\det(\eta M)| \geq \eta^d$, then the joint distribution on $M(X \circ R)$ has polynomially bounded density. A proof of this statement is provided in the full version of the paper [10].

When M is diagonal, the reduced instance trivially has independent entries, which is sufficient for most smoothed-analysis results to hold. Hence, strong reductions easily extend smoothed efficient algorithms. We conjecture that for many problems, upper-bounding the joint density of the input values suffices for smoothed-efficient algorithms to exist.

▶ **Lemma 14.** *(a) Suppose problem $\mathcal{Q}$ has (quasi-)polynomial smoothed complexity. If $\mathcal{P}$ admits a strong smoothness-preserving reduction to $\mathcal{Q}$, then $\mathcal{P}$ also has (quasi-)polynomial smoothed complexity. (b) If $\mathcal{Q}$ has a smoothed efficient algorithms when the input distribution has joint density bounds as in Proposition 13 and Lemma 7, then if $\mathcal{P}$ admits a weak smoothness-preserving reduction to $\mathcal{Q}$, $\mathcal{P}$ has (quasi-)polynomial smoothed complexity.*

The proof is straightforward modulo technicalities, and is included for completeness in the full version of this paper [10]. It is key that the matrix M has full (row) rank, since this ensures that the joint density on the reduced parameters is sufficiently bounded.

Observe that local-max-cut satisfies the conditions of part (b), since the proofs of [22, 1] simply apply Lemma 7 to the input, similarly to the argument in Section 3. Thus weak reductions to local-max-cut do imply smoothed efficient algorithms. This would not be an interesting notion of reduction if it only held for reductions to max-cut. As mentioned above, we conjecture that many smoothed-analysis results satisfy the conditions for part (b).

As an example, we note that the analyses for the smoothed efficient algorithms for a TSP 2-approximation [21] and for multidimensional bin-packing [28], are robust to this form of input assumption.

The smoothed complexity of local-max-bisection is open, but we believe that the natural local search procedure may admit a similar smoothed analysis to local-max-cut. This would imply a smoothed efficient algorithm for k-NetCoordNash for non-constant k.

4.1 Outline of Reductions

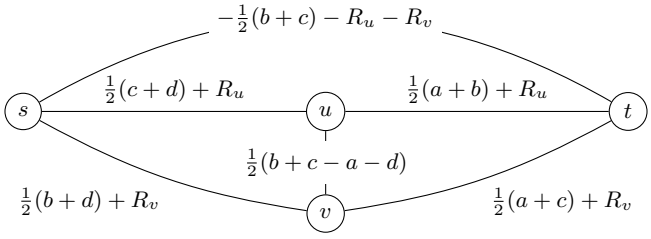
It suffices then, to provide weak smoothness-preserving reductions from NetCoordNash to local-max-cut and -bisection, as stated in Theorem 2. The main technical part involves the construction of M such that its rows are independent, and solutions of the resulting instance map to solutions of the original.

In this writeup, we give only a sketch of the reductions. The formal definitions and rank analyses may be found in the full version of this paper [10]. The reduction from 2-NetCoordNash to local-max-cut will be given first and in more detail, as it is cleaner, and is the basis for the reduction from k-NetCoordNash to local-max-bisection.

2-NetCoordNash reduces to Local-max-cut. Let $G = (V, E)$ be the game graph, with payoff vector A. Construct a weighted cut graph $H = (V', E')$ where $V' = V \cup \{s, t\}$, and $E' = E \cup \bigcup_{u \in V} \{su, ut\}$. We will define edge weights such that (1) every locally maximal cut is an s-t cut, and (2) the value of the cut (S, T) with $s \in S$ and $t \in T$ is equal to the total payoff of the game when $\sigma_u = 1$ if $u \in S$, and 2 if $u \in T$. Thus changing a player's strategy is equivalent to flipping its vertex across the cut. In the reverse direction, local-max-cuts are exactly the local maxima of the game's potential function, and thereby pure NE.

The following figure gives the edge weights for a small 2-player example which achieves the above properties, with the payoff matrix given as follows:

uv game payoffs: $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$



The general construction consists of placing copies of the above gadget on H for every game edge in E, taking the sum of the edge-weights for the su, ut, and st edges.

Observe that the above construction has edge weights which are linear combinations of the payoff values. Furthermore, for all values of R_u and R_v, the cut values are equal to payoff values. The R values are added only to increase the rank of the reduction matrix, and choosing them negative ensures that only s-t cuts are maximal. We show [10] by induction on $|V|$ that the matrix rows are independent, and so the reduction satisfies the necessary conditions. Note that the cut graph is complete if and only if the game graph is.

***k*-NetCoordNash reduces to Local-max-bisection.** The reductions from games with k strategies are not as straightforward. Let $G = (V, E)$ be the game graph with payoff vector A. We construct a weighted cut graph $H = (V', E')$ where $V' = (V \times [k]) \cup \{s_0, s_1, \ldots, s_{n(k-2)}, t\}$, the (player,strategy) pairs, and construct E' as follows: for every node (u, i) and $0 \leq a \leq n(k-2)$, we add an $\{s_a, (u, i)\}$ and $\{(u, i), t\}$ edge; for every $u \in V$

and $i \neq j$, we add a $\{(u,i),(u,j)\}$ edge; and for every $uv \in E$ and $i,j \in [k]$, we add a $\{(u,i),(v,j)\}$ edge. Call a cut (S,T) *valid* if $s_\ell \in S$ for all ℓ, $t \in T$, and S contains exactly one (u,i) node for all $u \in V$. Note that all valid cuts are balanced.

By construction, there is a natural strategy profile associated with each valid cut, namely if node (u,i) is in S then $\sigma_u = i$. We wish to choose edge weights such that (1) all locally maximal bisections are valid, and (2) the cut value is equal to $\Phi(\boldsymbol{\sigma})$. (1) will be achieved by giving low weight to the $\{(u,i),(u,j)\}$ edges, and higher weight to the $\{s_a,(u,i)\}$ edges, using the extra randomness available. This respectively ensures that it is always in our interest to have a small number of $(u,\cdot)$ nodes in S, but not none. As above, we will introduce extra randomness to the edge weights to ensure that M is full-rank. In this case, we will show M is full rank by arguing that it is upper-triangular after basic row operations.

The cut graph is again complete if and only if the game graph is, and thereby we have shown the second part of Theorem 2. The edge weights are not given directly, but are instead found by solving for the total cut values. As it would not be of any value to the reader to be given the values of the edge-weights without the full exposition, the details are left to the full version of this paper [10].

References

1 Omer Angel, Sébastien Bubeck, Yuval Peres, and Fan Wei. Local max-cut in smoothed polynomial time. In *ACM Symposium on Theory of Computing*, pages 429–437, 2017.

2 Elliot Anshelevich, Anirban Dasgupta, Eva Tardos, and Tom Wexler. Near-optimal network design with selfish agents. In *ACM Symposium on Theory of Computing*, pages 511–520, 2003.

3 Radhika Arava. Social Network Analysis Using Coordination Games. *arXiv preprint*, 2017. `arXiv:1708.09570`.

4 D. Arthur and S. Vassilvitskii. Worst-Case and Smoothed Analysis of the ICP Algorithm, with an Application to the k-Means Method. *SIAM J. on Computing*, 39(2):766–782, 2009.

5 Imre Bárány, Santosh Vempala, and Adrian Vetta. Nash equilibria in random games. *Random Struct. Algorithms*, 31(4):391–405, December 2007.

6 Paul Beame, Stephen Cook, Jeff Edmonds, Russell Impagliazzo, and Toniann Pitassi. The Relative Complexity of NP Search Problems. *J. Comput. Sys. Sci.*, 57(1):3–19, 1998.

7 Rene Beier and Berthold Vöcking. Random Knapsack in Expected Polynomial Time. *J. Comput. Syst. Sci.*, 69(3):306–329, November 2004.

8 Ali Bibak, Charles Carlson, and Karthekeyan Chandrasekaran. Improving the smoothed complexity of FLIP for max cut problems. In *ACM-SIAM SODA*, pages 897–916, 2019.

9 N. Bitansky, O. Paneth, and A. Rosen. On the Cryptographic Hardness of Finding a Nash Equilibrium. In *IEEE FOCS*, pages 1480–1498, 2015.

10 Shant Boodaghians, Rucha Kulkarni, and Ruta Mehta. Nash Equilibrium in Smoothed Polynomial Time for Network Coordination Games. *arXiv preprint*, 2018. `arXiv:1809.02280`.

11 Joris Broere, Vincent Buskens, Jeroen Weesie, and Henk Stoof. Network effects on coordination in asymmetric games. *Scientific reports*, 7(1):17016, 2017.

12 Yang Cai and Constantinos Daskalakis. On minmax theorems for multiplayer games. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 217–234, 2011.

13 X. Chen, X. Deng, and S.-H. Teng. Computing Nash Equilibria: Approximation and Smoothed Complexity. In *IEEE Symposium on Foundations of Computer Science*, pages 603–612, 2006.

14 Syngjoo Choi, Douglas Gale, Shachar Kariv, and Thomas Palfrey. Network architecture, salience and coordination. *Games and Economic Behavior*, 73(1):76–90, 2011.

15 Bruno Codenotti, Stefano De Rossi, and Marino Pagan. An Experimental Analysis of Lemke-Howson Algorithm. arXiv preprint, 2008. `arXiv:0811.3247`.

16 Valentina Damerow, Friedhelm Meyer auf der Heide, Harald Räcke, Christian Scheideler, and Christian Sohler. Smoothed Motion Complexity. In *Algorithms – ESA*, pages 161–171, 2003.

17 C. Daskalakis, P. Goldberg, and C. Papadimitriou. The Complexity of Computing a Nash Equilibrium. *SIAM Journal on Computing*, 39(1):195–259, 2009. doi:10.1137/070699652.

18 Argyrios Deligkas, John Fearnley, Tobenna Peter Igwe, and Rahul Savani. An Empirical Study on Computing Equilibria in Polymatrix Games. In *AAMAS*, pages 186–195, 2016.

19 Stéphane Durand and Bruno Gaujal. Complexity and Optimality of the Best Response Algorithm in Random Potential Games. In *Algorithmic Game Theory*, pages 40–51, 2016.

20 Glenn Ellison. Learning, local interaction, and coordination. *Econometrica: Journal of the Econometric Society*, pages 1047–1071, 1993.

21 Matthias Englert, Heiko Röglin, and Berthold Vöcking. Worst Case and Probabilistic Analysis of the 2-Opt Algorithm for the TSP. *Algorithmica*, 68(1):190–264, 2014.

22 Michael Etscheid and Heiko Röglin. Smoothed analysis of local search for the maximum-cut problem. *ACM Transactions on Algorithms (TALG)*, 13(2):25, 2017.

23 Alex Fabrikant, Christos Papadimitriou, and Kunal Talwar. The Complexity of Pure Nash Equilibria. In *ACM Symposium on Theory of Computing*, pages 604–612, 2004.

24 Michal Feldman and Tami Tamir. Conflicting congestion effects in resource allocation games. *Operations research*, 60(3):529–540, 2012.

25 Tobias Harks, Martin Hoefer, Max Klimm, and Alexander Skopalik. Computing pure Nash and strong equilibria in bottleneck congestion games. *Math. Prog.*, 141(1):193–215, 2013.

26 Ramesh Johari and John N Tsitsiklis. Efficiency loss in a network resource allocation game. *Mathematics of Operations Research*, 29(3):407–435, 2004.

27 David S. Johnson, Christos H. Papadimitriou, and Mihalis Yannakakis. How easy is local search? *Journal of Computer and System Sciences*, 37(1):79–100, 1988.

28 David Karger and Krzysztof Onak. Polynomial approximation schemes for smoothed and random instances of multidimensional packing problems. In *ACM-SIAM Symposium on Discrete Algorithms*, volume 7, pages 1207–1216, 2007.

29 Dharshana Kasthurirathna, Mahendra Piraveenan, and Michael Harré. Influence of topology in the evolution of coordination in complex networks under information diffusion constraints. *The European Physical Journal B*, 87(1):3, 2014.

30 Bodo Manthey and Rüdiger Reischuk. Smoothed Analysis of Binary Search Trees. In *Algorithms and Computation*, pages 483–492, 2005.

31 Dov Monderer and Lloyd S Shapley. Potential games. *Games and economic behavior*, 14(1):124–143, 1996.

32 Andrea Montanari and Amin Saberi. The spread of innovations in social networks. *National Academy of Sciences*, 107(47):20196–20201, 2010.

33 Heiko Röglin. The Complexity of Nash Equilibria, Local Optima, and Pareto-Optimal Solutions. *Fakultät für Math., Informatik und Naturw. der RWTH*, 2008.

34 Heiko Röglin and Berthold Vöcking. Smoothed analysis of integer programming. *Mathematical Programming*, 110(1):21–56, June 2007.

35 Robert W. Rosenthal. A class of games possessing pure-strategy Nash equilibria. *International J. of Game Theory*, 2(1):65–67, 1973.

36 T. Roughgarden and É. Tardos. How Bad is Selfish Routing? *J. ACM*, 49(2):236–259, 2002.

37 Tim Roughgarden. Routing Games, 2007.

38 Aviad Rubinstein. Settling the Complexity of Computing Approximate Two-player Nash Equilibria. *SIGecom Exch.*, 15(2):45–49, February 2017.

39 Rahul Savani and Bernhard von Stengel. Exponentially Many Steps for Finding a Nash Equilibrium in a Bimatrix Game. In *IEEE FOCS*, pages 258–267, 2004.

40 Alejandro A. Schäffer and Mihalis Yannakakis. Simple Local Search Problems That Are Hard to Solve. *SIAM J. Comput.*, 20(1):56–87, February 1991.

41 Daniel A. Spielman and Shang-Hua Teng. Smoothed Analysis of Algorithms: Why the Simplex Algorithm Usually Takes Polynomial Time. *J. ACM*, 51(3):385–463, May 2004.

Local-To-Global Agreement Expansion via the Variance Method

Tali Kaufman
Department of Computer Science, Bar-Ilan University, Ramat Gan, Israel
kaufmant@mit.edu

David Mass
Department of Computer Science, Bar-Ilan University, Ramat Gan, Israel
dudimass@gmail.com

Abstract

Agreement expansion is concerned with set systems for which local assignments to the sets with almost perfect pairwise consistency (i.e., most overlapping pairs of sets agree on their intersections) implies the existence of a global assignment to the ground set (from which the sets are defined) that agrees with most of the local assignments.

It is currently known that if a set system forms a *two-sided* or a *partite* high dimensional expander then agreement expansion is implied. However, it was not known whether agreement expansion can be implied for *one-sided* high dimensional expanders.

In this work we show that agreement expansion can be deduced for one-sided high dimensional expanders assuming that all the vertices' links (i.e., the neighborhoods of the vertices) are agreement expanders. Thus, for one-sided high dimensional expander, an agreement expansion of the large complicated complex can be deduced from agreement expansion of its small simple links.

Using our result, we settle the open question whether the well studied Ramanujan complexes are agreement expanders. These complexes are neither partite nor two-sided high dimensional expanders. However, they are one-sided high dimensional expanders for which their links are partite and hence are agreement expanders. Thus, our result implies that Ramanujan complexes are agreement expanders, answering affirmatively the aforementioned open question.

The local-to-global agreement expansion that we prove is based on the *variance method* that we develop. We show that for a high dimensional expander, if we define a function on its top faces and consider its local averages over the links then the variance of these local averages is much smaller than the global variance of the original function. This decreasing in the variance enables us to construct one global agreement function that ties together all local agreement functions.

2012 ACM Subject Classification Theory of computation → Computational complexity and cryptography

Keywords and phrases Agreement testing, High dimensional expanders, Local-to-global, Variance method

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.74

Funding *Tali Kaufman*: Supported by ERC and BSF.
David Mass: Supported by the Adams Fellowship Program of the Israel Academy of Sciences and Humanities.

1 Introduction

Agreement expansion has been extensively studied. It plays an important role in nearly all PCP constructions, and has found various applications in many recent works (see e.g., [8, 5, 1, 9] and [3]). Previous works could only prove agreement expansion for complexes which are two-sided local spectral expanders or partite one-sided local spectral expanders. However, the question for the general case of one-sided local spectral expanders remained open. In this work we show that all one-sided local spectral expanders are agreement expanders, given that

the local views of their vertices are agreement expanders. In particular, this solves an open question regarding the well studied Ramanujan complexes, which are not two-sided local spectral expanders nor partite, but the local views of their vertices are agreement expanders.

Our main theorem is a local-to-global agreement expansion. We show that agreement expansion of the entire complex can be deduced from agreement expansion of its vertices, i.e., if locally the complex is an agreement expander, then it is also globally expanding. In many cases of high dimensional expanders, the entire complex is a complicated object which is hard to understand, but it is composed of many simple local objects. We show that it is enough to argue regarding the simple local views of the complex, which then implies that the whole complicated complex is also an agreement expander.

The main theorem that we prove is based on the *variance method* that we develop. We show that for one-sided local spectral expanders, the variance of any function decreases as we go down the dimensions. Namely, if we define a function on the top faces of the complex and consider its local averages over the links then the variance of these local averages is much smaller than the global variance of the original function. This decreasing in variance enables us to construct a single global function that stitches together all the local agreement functions.

Agreement tests

Let S a collection of subsets over some ground set V. A *local assignment* is a collection of functions $f = \{f_\sigma\}_{\sigma\in S}$, where each f_σ is a function that specifies a $0,1$ value for each $u \in \sigma$. Any f_σ is local in the sense that it gives values only to the elements in σ, independently of the rest of the functions and the ground set. A *global assignment* is a single function $g : V \to \{0,1\}$ which specifices a value for every vertex in the ground set. We say that a local assignment $f = \{f_\sigma\}$ comes from a global assignment g, if for every $\sigma \in S$ it holds that $f_\sigma = g\big|_\sigma$, i.e., f_σ is just the restriction of g to σ.

Denote by $\mathcal{G}$ the set of all global assignments. An agreement test is specified by a distribution $\mathcal{D}$ on tuples $(\tau, \sigma_1, \sigma_2)$ such that $\sigma_1, \sigma_2 \in S$ and $\tau \subseteq \sigma_1 \cap \sigma_2$. The test picks τ, σ_1, σ_2 according to $\mathcal{D}$ and accepts if f_{σ_1} and f_{σ_2} agree on τ, i.e., if for every $u \in \tau$, $f_{\sigma_1}(\tau) = f_{\sigma_2}(\tau)$.[1] We denote the acceptance probability of the test by

$$\text{agree}_{\mathcal{D}}(f) = \Pr_{\tau,\sigma_1,\sigma_2\sim\mathcal{D}}[f_{\sigma_1}\big|_\tau = f_{\sigma_2}\big|_\tau].$$

It is easy to see that if a local assignment comes from a global assignment then the test accepts with probability 1. Denote by $\text{dist}(f, g)$ the fraction of $\sigma \in S$ for which $f_\sigma \neq g\big|_\sigma$. An agreement theorem states that if $\text{agree}_{\mathcal{D}}(f) \geq 1 - \varepsilon$ then f is $1 - O(\varepsilon)$ close to a global assignment, i.e., there exists a global assignment $g \in \mathcal{G}$ such that $\text{dist}(f, g) \leq O(\varepsilon)$. In other words, an agreement theorem guarantees that the agreement test provides a good approximation for the distance of a local assignment from the global assignments.

High dimensional expanders

A d-dimensional simplicial complex X is a $(d+1)$-hypergraph which is closed under containment: For any $(d+1)$-hyperedge σ in X, all of its subsets $\tau \subset \sigma$ also belong to X. A hyperedge σ is also called a face of the complex, and its dimension is $|\sigma| - 1$. The set of all k-dimensional faces of the complex is denoted by $X(k)$.

[1] A weaker notion of an agreement test, which we do not discuss in this paper, is concerned with an *approximate* global consistency, i.e., that f_σ agrees with $g\big|_\sigma$ on most of the elements in σ.

For any face $\sigma \in X$, its *link* is the subcomplex obtained by all the faces in X which contain σ after removing σ from all of them, and denoted by $X_\sigma = \{\tau \setminus \sigma \mid \sigma \subseteq \tau \in X\}$. Note that if X is of dimension d then X_σ is of dimension $d - |\sigma|$.

▶ **Definition 1.1** (Partite complex). *A d-dimensional complex X is called* partite *if its vertices can be partitioned into $d+1$ sets $X(0) = V_1 \cup V_2 \cup \cdots \cup V_{d+1}$ such that any d-dimensional face $\sigma \in X(d)$ contains a vertex from each V_i, i.e., $|\sigma \cap V_i| = 1$.*

In recent years, several distinct notions of high dimensional expansion have been studied. For a detailed survey regarding high dimensional expanders we refer the reader to [10]. In this work we focus on the spectral expansion of the links of the complex.

▶ **Definition 1.2** (Local spectral expansion). *A d-dimensional complex X is called a* λ-one-sided local spectral expander *(or* λ-two-sided local spectral expander*) if for every $-1 \leq k \leq d-2$ and every $\sigma \in X(k)$, the underlying graph[2] of X_σ is a λ-one-sided spectral expander (or λ-two-sided spectral expander, respectively).*

High dimensional expanders and agreement expansion

The work of [4] initiated the relation of high dimensional expanders to agreement tests. We follow their definition of *agreement expansion*.

▶ **Definition 1.3** (Agreement expansion). *A d-dimensional complex X is called a* c-agreement expander for dimension k *if there exists a distribution $\mathcal{D}$ such that for every local assignment $f = \{f_\sigma\}_{\sigma \in X(k)}$ there exists a global agreement function $g \in \mathcal{G}$ such that*

$$\mathrm{dist}(f, g) \leq c \cdot \mathrm{disagree}_{\mathcal{D}}(f),$$

where $\mathrm{disagree}_{\mathcal{D}}(f) = 1 - \mathrm{agree}_{\mathcal{D}}(f)$ is the rejection probability of the test.

The name "agreement expansion" comes from its similarity to other expansion measures, since X is an agreement expander if and only if

$$\min_f \frac{\mathrm{disagree}_{\mathcal{D}}(f)}{\mathrm{dist}(f, \mathcal{G})} \geq \frac{1}{c},$$

where the minimum is taken over all local assignments that do not come from a global assignment.

In [6], the authors prove that there exists a constant $c > 0$ such that the d-dimensional *complete complex*, which is the complex that contains all possible sets of size $\leq d+1$, is a c-agreement expander for its top dimension. Building on [6], [4] show that there exists a constant $c > 0$ such that a d-dimensional two-sided local spectral expander is a c-agreement expander *for dimension* $k = O(\sqrt{d})$. Their proof goes by a reduction to the complete complex, and therefore they could not prove agreement expansion for the top dimension of the complex, but only for some lower dimension.

In a recent work, Dikstein and Dinur [2] prove that there exists a constant $c > 0$ such that a d-dimensional two-sided local spectral expander or a partite one-sided local spectral expander are c-agreement expanders *for their top dimension*. However, the general question for one-sided local spectral expanders remained open.

[2] The graph whose vertices are $X_\sigma(0)$ and its edges are $X_\sigma(1)$.

Our main theorem in this work is that agreement expansion of a complex can be deduced from the agreement expansion of the links of its vertices. In particular, for any constant $c > 0$ there exists a constant $c' > 0$ *which is dependent only on* c such that a d-dimensional one-sided local spectral expander is a c'-agreement expander *for its top dimension*, given that the links of its vertices are c-agreement expanders for their top dimension.

▶ **Theorem 1.4** (Main Theorem, informal)**.** *For any constant $c > 0$ there exists a constant $c' = c'(c)$ such that if the links of the vertices of a one-sided local spectral expander are c-agreement expanders for their top dimension, then the entire complex is a c'-agreement expander for its top dimension. Moreover, the global agreement function that agrees with most of the local functions is defined by majority decoding.*

Our result extends [2] to general one-sided local expanders, which are not necessarily partite. Moreover, the result of [2] ensures that if the agreement test accepts with probability $1-\varepsilon$ then there exists *some global function* that agrees with $1-c\varepsilon$ of the local functions. The global function that [2] construct is not necessarily the majority function, but rather some conditional majority. We show here that the *majority function* agrees with $1-c'\varepsilon$ of the local functions, *regardless* of which functions agree with the local functions on each vertex. Our proof technique can be used for two-sided local spectral expanders and for partite one-sided local spectral expanders as well: As a first step use [2] to construct a local agreement function for each vertex, and then by our work conclude that the majority function agrees with most of these local agreement functions.

▶ **Corollary 1.5.** *There exists a constant $c > 0$ such that any d-dimensional two-sided local spectral expander and any d-dimensional partite one-sided local spectral expander is a c-agreement expander for its top dimension, where the global agreement function is defined by majority decoding.*

Ramanujan complexes

Much of the motivation for the study of high dimensional expanders comes from the existence of Ramanujan complexes, whose properties are optimal in almost every measure. Ramanujan complexes are the high dimensional analogs of the celebrated LPS Ramanujan graphs [11], which arise from number theory. In [12] the authors describe an explicit construction of a family of bounded degree Ramanujan complexes, i.e., every vertex is contained in a bounded number of faces. Thus, the number of d-dimensional faces in these complexes is linear in the number of vertices.

Ramanujan complexes are known to be one-sided local spectral expanders [7], and their links are also partite. However, Ramanujan complexes are not two-sided local spectral expanders nor partite, so previous works could not show that they are agreement expanders. As a corollary of our theorem, we settle this open question.

▶ **Corollary 1.6.** *There exists a constant $c > 0$ such that Ramanujan complexes are c-agreement expanders for their top dimension.*

The variance method

Our local-to-global agreement theorem is based on the *variance method*. The decreasing in variance idea has first appeared in [4], where the authors proved that in high dimensional expanders, the difference of variances of a function on successive dimensions is decreasing. The authors in [4] proved it *only for two-sided* local spectral expanders, and used it just for

the single purpose of showing that random walks on high dimensional expanders have an optimal convergence rate. We extend their work and show that the same holds for *one-sided* local spectral expanders as well. In addition, we show the general statement that the variance of any function on the top faces of a high dimensional expander decreases by a factor of d when considered on the vertices.

▶ **Theorem 1.7** (The variance method, informal). *Let X be a d-dimensional one-sided local spectral expander. For any function $h : X(d) \to \mathbb{R}$,*

$$\operatorname*{Var}_{u\in X(0)} \operatorname*{\mathbb{E}}_{\sigma\in X_u(d-1)} h(u\sigma) \leq O\left(\frac{1}{d}\right) \operatorname*{Var}_{\sigma\in X(d)} h(\sigma).$$

The small variance is important in order to get a global function that agrees with $1 - O(\varepsilon)$ of the local functions. Recall that we are given a local assignment $f = \{f_\sigma\}$ and a probability distribution $\mathcal{D}$ such that $\text{disagree}_{\mathcal{D}}(f) \leq \varepsilon$ and our goal is to construct a global function g such that $\text{dist}(f, g) \leq O(\varepsilon)$. By a simple union bound argument, it is easy to get $\text{dist}(f, g) \leq O(d\varepsilon)$. In order to get an agreement which is not dependent on d, we must bound the probability of the bad events by $O(\varepsilon/d)$ and then by a union bound to get an agreement of $O(\varepsilon)$. Using the variance method, we can show that the variance on the vertices decreases by a factor of d, and thus the bad events happen with a probability bounded by $O(\varepsilon/d)$. We believe that this method will find many more applications in the future.

Organization

In Section 2 we provide some required preliminaries regarding expander graphs and high dimensional expanders. In Section 3 we introduce the variance method. In Section 4 we prove our main theorem.

2 Preliminaries

2.1 Expander graphs

Let $G = (V, E)$ be a graph with positive weights on the edges $w : E \to \mathbb{R}_{>0}$. Assume without loss of generality that the sum of the weights is 1 (otherwise just normalize), i.e., w is a probability distribution on the edges. These weights induce a probability distribution on the vertices as well, where the probability of a vertex $u \in V$ is given by $\frac{1}{2}\sum_{e\ni u} w(e)$. All the following probabilities are taken according to these distributions.

For any two functions $h, h' : E \to \mathbb{R}$, we define their inner product by

$$\langle h, h'\rangle = \sum_{e\in E} w(e)h(e)h'(e) = \operatorname*{\mathbb{E}}_{e\in E} h(e)h'(e).$$

Similarly, for any two functions $g, g' : V \to \mathbb{R}$, their inner product is defined by

$$\langle g, g'\rangle = \operatorname*{\mathbb{E}}_{u\in V} g(u)g'(u).$$

The adjacency operator $A = A(G) : \mathbb{R}^V \to \mathbb{R}^V$ is defined by $Ag(u) = \mathbb{E}_{uv|u}\, g(v)$, where $uv|u$ denotes the event that the edge $\{u, v\}$ was chosen given that the vertex u was chosen.

Denote the eigenvalues of the adjacency operator of G by $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_{|V|}$. It is easy to see that $\lambda_1 = 1$ (with corresponding eigenfunction $\mathbf{1}$) and that $\lambda_{|V|} \geq -1$. Let $0 < \lambda < 1$ be a positive constant. The graph G is said to be a λ-*one-sided spectral expander* if $\lambda_2 \leq \lambda$, and G is said to be a λ-*two-sided spectral expander* if both $\lambda_2 \leq \lambda$ and $\lambda_{|V|} \geq -\lambda$.

2.2 High dimensional expanders

Recall that a d-dimensional simplicial complex X is a $(d+1)$-hypergraph which is closed under containment. Throughout the paper we refer to the probability of choosing a face of the complex, which is given by the following probabilistic process. Consider the random sequence of faces $\sigma_d \supset \sigma_{d-1} \supset \cdots \supset \sigma_0 \supset \emptyset$, where $\sigma_d \in X(d)$ is chosen uniformly at random, and then for each $k = d-1, \cdots, 0$, σ_k is chosen by removing a uniformly random vertex from σ_{k+1}. For any $\sigma \in X(k)$, we denote the probability of choosing σ by

$$\Pr[\sigma] = \Pr[\sigma_d \supset \sigma_{d-1} \supset \cdots \supset \sigma_0 \supset \emptyset \mid \sigma_k = \sigma].$$

For any $-1 \leq k < d$ and a face $\sigma \in X(k)$, its *link* is the $(d-k-1)$-dimensional complex defined by $X_\sigma = \{\tau \setminus \sigma \mid \sigma \subseteq \tau \in X\}$. The probability of choosing a face $\tau \in X_\sigma$ is given by $\Pr[\tau \in X_\sigma] = \Pr[\tau \cup \sigma \mid \sigma_k = \sigma]$.

For the agreement test, we use the $\mathcal{D}_{d,\ell}$ distribution as defined in [2].

▶ **Definition 2.1** (The $\mathcal{D}_{d,\ell}$ distribution)**.** *Let X be a d-dimensional simplicial complex and $\ell < d$ be a positive integer. The $\mathcal{D}_{d,\ell}$ distribution is defined by the following random process:*

1. *Choose $\tau \in X(\ell)$ at random.*
2. *Choose $\sigma_1, \sigma_2 \in X_\tau(d-\ell-1)$ independently at random.*

The tuple that is then returned is $(\tau, \tau \cup \sigma_1, \tau \cup \sigma_2)$.

3 The Variance Method

We show in this section that for one-sided local spectral expanders, the variance of any function decreases as we go down the dimensions of the complex. We first bound the variance in one-sided spectral expander graphs, and then show how to use this bound in order to bound the variance on functions of a complex.

3.1 Bounding the variance in expander graphs

Let $G = (V, E)$ be a weighted graph. Recall that the adjacency operator $A : \mathbb{R}^V \to \mathbb{R}^V$ is defined by $Ag(u) = \mathbb{E}_{uv|u}\, g(v)$. We define the following two additional averaging operators:

- $A^\downarrow : \mathbb{R}^E \to \mathbb{R}^V$ defined by $A^\downarrow h(u) = \mathbb{E}_{e|u}\, h(e)$.
- $A^\uparrow : \mathbb{R}^V \to \mathbb{R}^E$ defined by $A^\uparrow g(e) = \mathbb{E}_{u|e}\, g(u)$.

The following properties are pretty standard.

(1) $A^\downarrow$ and $A^\uparrow$ are adjoint to each other, i.e., for any two vectors $h \in \mathbb{R}^E, g \in \mathbb{R}^V$,

$$\langle A^\downarrow h, g\rangle = \mathop{\mathbb{E}}_{u} \mathop{\mathbb{E}}_{e|u} h(e)g(u) = \mathop{\mathbb{E}}_{e} \mathop{\mathbb{E}}_{u|e} h(e)g(u) = \langle h, A^\uparrow g\rangle.$$

(2) Denote by $\lambda_2(A^\downarrow A^\uparrow)$ and $\lambda_2(A)$ the second largest eigenvalues of $A^\downarrow A^\uparrow$ and A correspondingly. It is easy to check that $A^\downarrow A^\uparrow = (A + I)/2$, where I is the identity operator. Thus,

$$\lambda_2(A^\downarrow A^\uparrow) = \frac{1 + \lambda_2(A)}{2}.$$

(3) $A^\uparrow A^\downarrow$ and $A^\downarrow A^\uparrow$ have the same non-zero eigenvalues. In particular, $\lambda_2(A^\uparrow A^\downarrow) = \lambda_2(A^\downarrow A^\uparrow)$.

(4) By Rayleigh quotient

$$\lambda_2(A^{\uparrow}A^{\downarrow}) = \max_{\substack{h:E\to\mathbb{R}\\ h\perp\mathbf{1}}} \frac{\langle A^{\uparrow}A^{\downarrow}h, h\rangle}{\|h\|^2}.$$

We will use the following lemma, which is immediate from the aforementioned properties.

▶ **Lemma 3.1.** *Let $G=(V,E)$ be a λ-one-sided spectral expander graph. For any function $h: E\to\mathbb{R}$ such that $h\perp\mathbf{1}$ it holds that*

$$\left\|A^{\downarrow}h\right\|^2 \le \frac{1+\lambda}{2}\|h\|^2.$$

Proof. It follows immediately from the properties mentioned above, since

$$\left\|A^{\downarrow}h\right\|^2 = \langle A^{\downarrow}h, A^{\downarrow}h\rangle = \langle A^{\uparrow}A^{\downarrow}h, h\rangle \le \lambda_2(A^{\uparrow}A^{\downarrow})\|h\|^2 = \lambda_2(A^{\downarrow}A^{\uparrow})\|h\|^2 = \frac{1+\lambda_2(A)}{2}\|h\|^2,$$

where we used properties (1), (4), (3) and (2) in that order. ◀

We can now bound the variance of any function on the edges of the graph.

▶ **Lemma 3.2.** *Let $G=(V,E)$ be a λ-one-sided spectral expander graph. For any function $h: E\to\mathbb{R}$ it holds that*

$$\mathop{\mathbb{E}}_{u}\mathop{\mathrm{Var}}_{e|u} h(e) \ge \frac{1-\lambda}{1+\lambda}\mathop{\mathrm{Var}}_{u}\mathop{\mathbb{E}}_{e|u} h(e).$$

Proof. Assume without loss of generality that $\mathbb{E}_e h(e)=0$ (otherwise, define $h'=h-\mathbb{E}_e h(e)$ and continue with h'). Now, note that

$$\mathop{\mathrm{Var}}_{u}\mathop{\mathbb{E}}_{e|u} h(e) = \mathop{\mathbb{E}}_{u}\left(\mathop{\mathbb{E}}_{e|u} h(e)\right)^2 = \left\|A^{\downarrow}h\right\|^2. \tag{1}$$

Note also that

$$\mathop{\mathbb{E}}_{u}\mathop{\mathrm{Var}}_{e|u} h(e) = \mathop{\mathbb{E}}_{u}\left(\mathop{\mathbb{E}}_{e|u} h(e)^2 - \left(\mathop{\mathbb{E}}_{e|u} h(e)\right)^2\right) = \mathop{\mathbb{E}}_{e} h(e)^2 - \mathop{\mathbb{E}}_{u}\left(\mathop{\mathbb{E}}_{e|u} h(e)\right)^2 = \|h\|^2 - \left\|A^{\downarrow}h\right\|^2. \tag{2}$$

Since $\mathbb{E}_e h(e)=0$, by lemma 3.1 we have that

$$\|h\|^2 \ge \frac{2}{1+\lambda}\left\|A^{\downarrow}h\right\|^2. \tag{3}$$

Combining (1), (2) in (3) finishes the proof. ◀

3.2 Bounding the variance in one-sided local spectral expanders

The following lemma follows immediately from lemma 3.2.

▶ **Lemma 3.3.** *Let X be a d-dimensional λ-one-sided local spectral expander. For any $1\le k\le d$ and a function $h: X(k)\to\mathbb{R}$ it holds that*

$$\mathop{\mathbb{E}}_{\sigma\in X(k-1)}\mathop{\mathrm{Var}}_{u\in X_\sigma(0)} h(\sigma u) \ge \frac{1-\lambda}{1+\lambda}\mathop{\mathbb{E}}_{\sigma\in X(k-2)}\mathop{\mathrm{Var}}_{u\in X_\sigma(0)}\mathop{\mathbb{E}}_{v\in X_{\sigma u}(0)} h(\sigma uv).$$

Proof. By definition, for every $\sigma \in X(k-2)$, the underlying graph of X_σ is a λ-one-sided spectral expander. Thus, by lemma 3.2,

$$\mathop{\mathbb{E}}_{u\in X_\sigma(0)} \mathop{\mathrm{Var}}_{v\in X_{\sigma u}(0)} h(\sigma uv) \geq \frac{1-\lambda}{1+\lambda} \mathop{\mathrm{Var}}_{u\in X_\sigma(0)} \mathop{\mathbb{E}}_{v\in X_{\sigma u}(0)} h(\sigma uv).$$

Averaging over all $\sigma \in X(k-2)$ finishes the proof. ◀

We can now prove the following two lemmas for one-sided local spectral expanders.

▶ **Lemma 3.4.** *Let X be a $(d-1)$-dimensional λ-one-sided local spectral expander, where $d \geq k\cdot\ell$ for $k \geq 2$, $\ell \geq 1$. If $\lambda \leq 1/2d$ then for any function $h : X(d-1) \to \mathbb{R}$,*

$$\mathop{\mathrm{Var}}_{\sigma\in X(d-1)} h(\sigma) \leq \frac{k+2}{k-1} \mathop{\mathbb{E}}_{\tau\in X(\ell-1)} \mathop{\mathrm{Var}}_{\sigma\in X_\tau(d-\ell-1)} h(\tau\sigma).$$

Proof. Note that

$$\begin{aligned}
\mathop{\mathrm{Var}}_{\sigma\in X(d-1)} h(\sigma) &= \mathop{\mathbb{E}}_{u_1}\cdots\mathop{\mathbb{E}}_{u_d} h(u_1\cdots u_d)^2 - \Big(\mathop{\mathbb{E}}_{u_1}\cdots\mathop{\mathbb{E}}_{u_d} h(u_1\cdots u_d)\Big)^2 \\
&= \mathop{\mathbb{E}}_{u_1}\cdots\mathop{\mathbb{E}}_{u_d} h(u_1\cdots u_d)^2 - \mathop{\mathbb{E}}_{u_1}\cdots\mathop{\mathbb{E}}_{u_{d-1}}\Big(\mathop{\mathbb{E}}_{u_d} h(u_1\cdots u_d)\Big)^2 + \\
&\quad \mathop{\mathbb{E}}_{u_1}\cdots\mathop{\mathbb{E}}_{u_{d-1}}\Big(\mathop{\mathbb{E}}_{u_d} h(u_1\cdots u_d)\Big)^2 - \mathop{\mathbb{E}}_{u_1}\cdots\mathop{\mathbb{E}}_{u_{d-2}}\Big(\mathop{\mathbb{E}}_{u_{d-1}}\mathop{\mathbb{E}}_{u_d} h(u_1\cdots u_d)\Big)^2 + \cdots \\
&\quad \cdots + \mathop{\mathbb{E}}_{u_1}\Big(\mathop{\mathbb{E}}_{u_2}\cdots\mathop{\mathbb{E}}_{u_d} h(u_1\cdots u_d)\Big)^2 - \Big(\mathop{\mathbb{E}}_{u_1}\cdots\mathop{\mathbb{E}}_{u_d} h(u_1\cdots u_d)\Big)^2 \qquad (4) \\
&= \mathop{\mathbb{E}}_{u_1}\cdots\mathop{\mathbb{E}}_{u_{d-1}}\mathop{\mathrm{Var}}_{u_d} h(u_1\cdots u_d) + \\
&\quad \mathop{\mathbb{E}}_{u_1}\cdots\mathop{\mathbb{E}}_{u_{d-2}}\mathop{\mathrm{Var}}_{u_{d-1}}\mathop{\mathbb{E}}_{u_d} h(u_1\cdots u_d) + \cdots \\
&\quad \cdots + \mathop{\mathrm{Var}}_{u_1}\mathop{\mathbb{E}}_{u_2}\cdots\mathop{\mathbb{E}}_{u_d} h(u_1\cdots u_d),
\end{aligned}$$

where the second equality follows by a telescoping sum argument. Similarly,

$$\begin{aligned}
\mathop{\mathbb{E}}_{\tau\in X(\ell-1)}\mathop{\mathrm{Var}}_{\sigma\in X_\tau(d-\ell-1)} h(\tau\sigma) &= \mathop{\mathbb{E}}_{u_1}\cdots\mathop{\mathbb{E}}_{u_{d-1}}\mathop{\mathrm{Var}}_{u_d} h(u_1\cdots u_d) + \\
&\quad \mathop{\mathbb{E}}_{u_1}\cdots\mathop{\mathbb{E}}_{u_{d-2}}\mathop{\mathrm{Var}}_{u_{d-1}}\mathop{\mathbb{E}}_{u_d} h(u_1\cdots u_d) + \cdots \\
&\quad \cdots + \mathop{\mathbb{E}}_{u_1}\cdots\mathop{\mathbb{E}}_{u_\ell}\mathop{\mathrm{Var}}_{u_{\ell+1}}\mathop{\mathbb{E}}_{u_{\ell+2}}\cdots\mathop{\mathbb{E}}_{u_d} h(u_1\cdots u_d).
\end{aligned}$$

For any $1 \leq i \leq k-1$, invoking lemma 3.3 $i\cdot\ell$ times on each of the last ℓ summands of (4) yields

$$\begin{aligned}
&\mathop{\mathbb{E}}_{u_1}\cdots\mathop{\mathbb{E}}_{u_{\ell-1}}\mathop{\mathrm{Var}}_{u_\ell}\mathop{\mathbb{E}}_{u_{\ell+1}}\cdots\mathop{\mathbb{E}}_{u_d} h(u_1\cdots u_d) + \cdots + \mathop{\mathrm{Var}}_{u_1}\mathop{\mathbb{E}}_{u_2}\cdots\mathop{\mathbb{E}}_{u_d} h(u_1\cdots u_d) \\
&\quad \leq \Big(\frac{1+\lambda}{1-\lambda}\Big)^{i\ell}\Bigg(\mathop{\mathbb{E}}_{u_1}\cdots\mathop{\mathbb{E}}_{u_{(i+1)\ell-1}}\mathop{\mathrm{Var}}_{u_{(i+1)\ell}}\mathop{\mathbb{E}}_{u_{(i+1)\ell+1}}\cdots\mathop{\mathbb{E}}_{u_d} h(u_1\cdots u_d) + \cdots \\
&\qquad\qquad \cdots + \mathop{\mathbb{E}}_{u_1}\cdots\mathop{\mathbb{E}}_{u_{i\ell}}\mathop{\mathrm{Var}}_{u_{i\ell+1}}\mathop{\mathbb{E}}_{u_{i\ell+2}}\cdots\mathop{\mathbb{E}}_{u_d} h(u_1\cdots u_d)\Bigg) \\
&\quad \leq e\Bigg(\mathop{\mathbb{E}}_{u_1}\cdots\mathop{\mathbb{E}}_{u_{(i+1)\ell-1}}\mathop{\mathrm{Var}}_{u_{(i+1)\ell}}\mathop{\mathbb{E}}_{u_{(i+1)\ell+1}}\cdots\mathop{\mathbb{E}}_{u_d} h(u_1\cdots u_d) + \cdots \\
&\qquad\qquad \cdots + \mathop{\mathbb{E}}_{u_1}\cdots\mathop{\mathbb{E}}_{u_{i\ell}}\mathop{\mathrm{Var}}_{u_{i\ell+1}}\mathop{\mathbb{E}}_{u_{i\ell+2}}\cdots\mathop{\mathbb{E}}_{u_d} h(u_1\cdots u_d)\Bigg),
\end{aligned}$$

where the last inequality follows since $i \cdot \ell \leq d-1$ and $\lambda \leq 1/2d$. Thus, by invoking lemma 3.3 $i \cdot \ell$ times on the last ℓ summands of (4) for $i = 1, \ldots, k-1$ we get

$$(k-1) \operatorname*{Var}_{\sigma \in X(d-1)} h(\sigma) \leq (k-1+e) \operatorname*{\mathbb{E}}_{\tau \in X(\ell-1)} \operatorname*{Var}_{\sigma \in X_\tau(d-\ell-1)} h(\tau\sigma),$$

which finishes the proof. ◀

▶ **Lemma 3.5.** *Let X be a d-dimensional λ-one-sided local spectral expander. If $\lambda \leq 1/(2d+1)$ then for any function $h : X(\ell) \to \mathbb{R}$, $\ell \leq d$,*

$$\operatorname*{Var}_{u \in X(0)} \operatorname*{\mathbb{E}}_{\sigma \in X_u(\ell-1)} h(u\sigma) \leq \frac{8}{5\ell} \operatorname*{Var}_{\sigma \in X(\ell)} h(\sigma).$$

Proof. Note that

$$\begin{aligned} \operatorname*{Var}_{\sigma \in X(\ell)} h(\sigma) = \operatorname*{\mathbb{E}}_{u_1} \cdots \operatorname*{\mathbb{E}}_{u_\ell} \operatorname*{Var}_{u_{\ell+1}} h(u_1 \cdots u_{\ell+1}) + \\ \operatorname*{\mathbb{E}}_{u_1} \cdots \operatorname*{\mathbb{E}}_{u_{\ell-1}} \operatorname*{Var}_{u_\ell} \operatorname*{\mathbb{E}}_{u_{\ell+1}} h(u_1 \cdots u_{\ell+1}) + \cdots \\ \cdots + \operatorname*{Var}_{u_1} \operatorname*{\mathbb{E}}_{u_2} \cdots \operatorname*{\mathbb{E}}_{u_{\ell+1}} h(u_1 \cdots u_{\ell+1}). \end{aligned} \tag{5}$$

By invoking lemma 3.3 on each summand of (5), where on the ith summand we invoke it $i-1$ times, we get

$$\operatorname*{Var}_{\sigma \in X(\ell)} h(\sigma) \geq \sum_{i=0}^{\ell} \left(\frac{1-\lambda}{1+\lambda}\right)^i \operatorname*{Var}_{u \in X(0)} \operatorname*{\mathbb{E}}_{\sigma \in X_u(\ell-1)} h(u\sigma) \geq \ell(1-e^{-1}) \operatorname*{Var}_{u \in X(0)} \operatorname*{\mathbb{E}}_{\sigma \in X_u(\ell-1)} h(u\sigma),$$

where the last inequality follows since $\lambda \leq 1/(2d+1)$. ◀

▶ **Corollary 3.6.** *By the assumptions of lemma 3.5, if the range of h is $[0,1]$ then*

$$\operatorname*{Var}_{u \in X(0)} \operatorname*{\mathbb{E}}_{\sigma \in X_u(\ell-1)} h(u\sigma) \leq \frac{8}{5\ell} \operatorname*{\mathbb{E}}_{\sigma \in X(\ell)} h(\sigma).$$

Proof. It follows immediately since for h with range in $[0,1]$, $\operatorname{Var}_{\sigma \in X(\ell)} h(\sigma) \leq \mathbb{E}_{\sigma \in X(\ell)} h(\sigma)$. ◀

4 Local-to-Global Agreement Expansion

In this section we state and prove our main theorem.

▶ **Theorem 4.1** (Main Theorem). *For any constant $c > 0$ and two natural numbers $d > \ell \geq 2$ such that $\ell = \Theta(d)$, there exists a constant $c' = c'(c, d/\ell)$ such that the following holds. Let X be a d-dimensional λ-one-sided local spectral expander, $\lambda \leq 1/(2d+1)$. If for every vertex $v \in X(0)$, the link X_v is a c-agreement expander with regard to the $\mathcal{D}_{d-1,\ell-1}$ distribution, then X is a c'-agreement expander with regard to the $\mathcal{D}_{d,\ell}$ distribution. Moreover, the global agreement function is defined by majority decoding.*

The general idea will be to decompose the global agreement probability to local agreements in the links, and to show that with high probability the local functions agree with the global majority function.

Let $f = \{f_\sigma\}_{\sigma \in X(d)}$. Define the majority function $\mathrm{maj} : X(0) \to \{0,1\}$ by

$$\mathrm{maj}(v) = \operatorname*{arg\,max}_{\alpha \in \{0,1\}} \left\{ \Pr_{\sigma \in X_v(d-1)} [f_{v\sigma}(v) = \alpha] \right\}.$$

Denote by $\varepsilon = \mathrm{disagree}(f)$. The proof will follow from the following claims.

▷ Claim 4.2. $\mathop{\mathbb{E}}_{v\in X(0)}\Pr_{\sigma\in X_v(d-1)}[f_{v\sigma}(v)\neq \mathrm{maj}(v)] \le 2\Big(1+\frac{3}{d/\ell-1}\Big)\varepsilon.$

▷ Claim 4.3. $\mathop{\mathbb{E}}_{v\in X(0)}\Pr_{\sigma\in X_v(d-1)}\big[g_v\big|_\sigma \neq \mathrm{maj}\big|_\sigma\big] \le 120\Big(\frac{2d}{\ell}c^2+2c+1+\frac{3}{d/\ell-1}\Big)\varepsilon,$

where $g_v : X_v(0)\to\{0,1\}$ is the function promised by the c-agreement expansion of X_v.

Proof of Main Theorem. Consider a vertex $v\in X(0)$ and a top face in its link $\sigma\in X_v(d-1)$. Note that if the following three events happen then $f_{v\sigma}$ agrees with $\mathrm{maj}\big|_{v\sigma}$:
(1) $f_{v\sigma}(v)=\mathrm{maj}(v)$,
(2) $f_{v\sigma}\big|_\sigma = g_v\big|_\sigma$,
(3) $g_v\big|_\sigma = \mathrm{maj}\big|_\sigma$.
Therefore,

$$\begin{aligned}
\Pr_{\sigma\in X(d)}[f_\sigma\neq\mathrm{maj}\big|_\sigma] &= \mathop{\mathbb{E}}_{v\in X(0)}\Pr_{\sigma\in X_v(d-1)}[f_{v\sigma}\big|_{v\sigma}\neq\mathrm{maj}\big|_{v\sigma}]\\
&\le \mathop{\mathbb{E}}_{v\in X(0)}\Big(\Pr_{\sigma\in X_v(d-1)}[f_{v\sigma}(v)\neq\mathrm{maj}(v)]+\\
&\qquad\Pr_{\sigma\in X_v(d-1)}[f_{v\sigma}\big|_\sigma\neq g_v\big|_\sigma]+\\
&\qquad\Pr_{\sigma\in X_v(d-1)}[g_v\big|_\sigma\neq\mathrm{maj}\big|_\sigma]\Big)\\
&\le 2\Big(1+\frac{3}{d/\ell-1}\Big)\varepsilon + c\varepsilon + 120\Big(\frac{2d}{\ell}c^2+2c+1+\frac{3}{d/\ell-1}\Big)\varepsilon\\
&= \Big(\frac{240d}{\ell}c^2+241c+122+\frac{366}{d/\ell-1}\Big)\varepsilon,
\end{aligned}$$

where the second inequality follows by claims 4.2 and 4.3, and by the c-agreement expansion of the link of every vertex in the complex. Finally, the theorem follows since $d/l=\Theta(1)$. ◀

4.1 Proofs of Claims 4.2 and 4.3

Proof of Claim 4.2. The idea of the proof is by the variance method, as follows. For any vertex $v\in X(0)$ we consider the indicator function on the d-faces whether a local function f_σ agrees with the majority on v. We note that the global variance of this function indicates the disagreement probability of two d-faces on v, and the average over $\tau\in X_v(\ell-1)$ of local variances indicates the disagreement probability of two d-faces on v *with intersection of size at least* ℓ. Since $\ell=\Theta(d)$ we conclude by the variance method that these two distributions are approximately equal. Details follow.

Let $v\in X(0)$. Define the indicator function $h_v : X_v(d-1)\to\{0,1\}$ by

$$h_v(\sigma)=\begin{cases}1 & f_{v\sigma}(v)\neq\mathrm{maj}(v),\\ 0 & f_{v\sigma}(v)=\mathrm{maj}(v).\end{cases}$$

By definition, $\mathbb{E}_{\sigma\in X_v(d-1)}h_v(\sigma)\le 1/2$. Thus,

$$\mathop{\mathbb{E}}_{\sigma\in X_v(d-1)}h_v(\sigma) \le 2\mathop{\mathbb{E}}_{\sigma\in X_v(d-1)}h_v(\sigma)\left(1-\mathop{\mathbb{E}}_{\sigma\in X_v(d-1)}h_v(\sigma)\right). \tag{6}$$

Since h_v is an indicator function

$$\mathop{\mathbb{E}}_{\sigma\in X_v(d-1)}h_v(\sigma)\left(1-\mathop{\mathbb{E}}_{\sigma\in X_v(d-1)}h_v(\sigma)\right) = \mathop{\mathrm{Var}}_{\sigma\in X_v(d-1)}h_v(\sigma). \tag{7}$$

Since X is a λ-one-sided local spectral expander, by lemma 3.4

$$\operatorname*{Var}_{\sigma\in X_v(d-1)} h_v(\sigma) \le \frac{d+2\ell}{d-\ell} \operatorname*{\mathbb{E}}_{\tau\in X_v(\ell-1)} \operatorname*{Var}_{\sigma\in X_{v\tau}(d-\ell-1)} h_v(\tau\sigma). \tag{8}$$

Again, since h_v is an indicator function

$$\operatorname*{Var}_{\sigma\in X_{v\tau}(d-\ell-1)} h_v(\tau\sigma) = \operatorname*{\mathbb{E}}_{\sigma\in X_{v\tau}(d-\ell-1)} h_v(\tau\sigma)\left(1 - \operatorname*{\mathbb{E}}_{\sigma\in X_{v\tau}(d-\ell-1)} h_v(\tau\sigma)\right). \tag{9}$$

Combining (6), (7), (8) and (9) yields

$$\operatorname*{Pr}_{\sigma\in X_v(d-1)}[f_{v\sigma}(v)\neq \mathrm{maj}(v)] \le$$
$$2\frac{d+2\ell}{d-\ell} \operatorname*{\mathbb{E}}_{\tau\in X_v(\ell-1)} \operatorname*{Pr}_{\sigma_1,\sigma_2\in X_{v\tau}(d-\ell-1)}[f_{v\tau\sigma_1}(v)\neq \mathrm{maj}(v)\wedge f_{v\tau\sigma_2}(v)=\mathrm{maj}(v)].$$

Finally, since $f_{v\tau\sigma_1}(v)\neq \mathrm{maj}(v)\wedge f_{v\tau\sigma_2}(v)=\mathrm{maj}(v)$ implies that $f_{v\tau\sigma_1}\big|_{v\tau} \neq f_{v\tau\sigma_2}\big|_{v\tau}$, we can conclude that

$$\begin{aligned}\operatorname*{\mathbb{E}}_{v\in X(0)} \operatorname*{Pr}_{\sigma\in X_v(d-1)}[f_{v\sigma}(v)\neq \mathrm{maj}(v)] &\le 2\frac{d+2\ell}{d-\ell} \operatorname*{\mathbb{E}}_{\tau\in X(\ell)} \operatorname*{Pr}_{\sigma_1,\sigma_2\in X_\tau(d-\ell-1)}[f_{\tau\sigma_1}\big|_\tau \neq f_{\tau\sigma_2}\big|_\tau]\\ &\le 2\Big(1+\frac{3}{d/\ell-1}\Big)\varepsilon.\end{aligned}$$

◁

Before proving Claim 4.3, let us define the following set of "bad neighbors". For any vertex $u\in X(0)$, by the agreement expansion of X_u, there exists a function $g_u : X_u(0)\to\{0,1\}$ that agrees with most of the top faces on u. We say that $v\in X_u(0)$ is a bad neighbor of u if g_u disagrees with many top faces that contain v:

$$B_u = \left\{ v\in X_u(0) \;\middle|\; \operatorname*{Pr}_{\sigma\in X_{uv}(d-2)}[f_{uv\sigma}\big|_{v\sigma}\neq g_u\big|_{v\sigma}] > \frac{3}{10}\right\}.$$

We will use the following two claims, which guarantee that on average almost all of the neighbors are not bad.

▷ Claim 4.4. $\operatorname*{\mathbb{E}}_{u\in X(0)} \operatorname*{Pr}_{v\in X_u(0)}[v\in B_u] \le 160\Big(\frac{d}{\ell}c^2+c\Big)\frac{\varepsilon}{d-1}$.

▷ Claim 4.5. $\operatorname*{\mathbb{E}}_{v\in X(0)} \operatorname*{Pr}_{u\in X_v(0)}[g_u(v)\neq \mathrm{maj}(v) \;\wedge\; v\notin B_u] \le 80\Big(1+\frac{3}{d/\ell-1}\Big)\frac{\varepsilon}{d-1}$.

Proof of Claim 4.3. Note that

$$\begin{aligned}\operatorname*{\mathbb{E}}_{u\in X(0)} \operatorname*{Pr}_{\sigma\in X_u(d-1)}[g_u\big|_\sigma \neq \mathrm{maj}\big|_\sigma] &= \operatorname*{\mathbb{E}}_{u\in X(0)} \operatorname*{Pr}_{\sigma\in X_u(d-1)}[\exists v\in\sigma \text{ s.t. } g_u(v)\neq \mathrm{maj}(v)]\\ &\le d \operatorname*{\mathbb{E}}_{u\in X(0)} \operatorname*{Pr}_{v\in X_u(0)}[g_u(v)\neq \mathrm{maj}(v)].\end{aligned}$$

By the law of total probability

$$\begin{aligned}\operatorname*{\mathbb{E}}_{u\in X(0)} \operatorname*{Pr}_{v\in X_u(0)}[g_u(v)\neq \mathrm{maj}(v)] &= \operatorname*{\mathbb{E}}_{u\in X(0)} \operatorname*{Pr}_{v\in X_u(0)}[g_u(v)\neq \mathrm{maj}(v) \;\wedge\; v\in B_u] +\\ &\quad \operatorname*{\mathbb{E}}_{v\in X(0)} \operatorname*{Pr}_{u\in X_u(0)}[g_u(v)\neq \mathrm{maj}(v) \;\wedge\; v\notin B_u]\\ &\le 160\Big(\frac{d}{\ell}c^2+c\Big)\frac{\varepsilon}{d-1} + 80\Big(1+\frac{3}{d/\ell-1}\Big)\frac{\varepsilon}{d-1}\\ &= 80\Big(\frac{2d}{\ell}c^2+2c+1+\frac{3}{d/\ell-1}\Big)\frac{\varepsilon}{d-1},\end{aligned}$$

where the inequality follows by claims 4.4 and 4.5. Therefore,

$$\mathop{\mathbb{E}}_{u\in X(0)} \Pr_{\sigma\in X_u(d-1)}[g_u\big|_\sigma \neq \mathrm{maj}\big|_\sigma] \leq 120\Big(\frac{2d}{\ell}c^2 + 2c + 1 + \frac{3}{d/\ell - 1}\Big)\varepsilon,$$

where the inequality follows by the assumption that $d \geq 3$. ◁

4.2 Proofs of Claims 4.4 and 4.5

For any $u \in X(0)$, denote by ε_u the disagreement probability conditioned on u:

$$\varepsilon_u = \mathop{\mathbb{E}}_{\tau\in X_u(\ell-1)} \Pr_{\sigma_1,\sigma_2\in X_{u\tau}(d-\ell-1)}[f_{u\tau\sigma_1}\big|_{u\tau} \neq f_{u\tau\sigma_2}\big|_{u\tau}],$$

and define the following set of vertices:

$$S = \left\{ u \in X(0) \;\middle|\; \varepsilon_u \leq \frac{1}{5c} \right\}.$$

The proofs will follow from the following claims.

▷ Claim 4.6. $\Pr_{u\in X(0)}[u \notin S] \leq 160c^2\frac{\varepsilon}{\ell}$.

▷ Claim 4.7. For any vertex $u \in S$,

$$\Pr_{v\in X_u(0)}[v \in B_u \mid u \in S] \leq 160c\frac{\varepsilon_u}{d-1}.$$

Proof of Claim 4.4. Assuming claims 4.6 and 4.7 the proof follows immediately since

$$\begin{aligned}
\mathop{\mathbb{E}}_{u\in X(0)} \Pr_{v\in X_u(0)}[v \in B_u] &\leq \Pr_{u\in X(0)}[u \notin S] + \mathop{\mathbb{E}}_{u\in S} \Pr_{v\in X_u(0)}[v \in B_u] \\
&\leq 160c^2\frac{\varepsilon}{\ell} + 160c\frac{1}{d-1}\mathop{\mathbb{E}}_{u\in S}\varepsilon_u \\
&\leq 160\Big(\frac{d}{\ell}c^2 + c\Big)\frac{\varepsilon}{d-1},
\end{aligned}$$

where the last inequality follows since $\mathbb{E}_{u\in X(0)}\varepsilon_u \leq \varepsilon$. ◁

Proof of Claim 4.5. Consider a vertex $v \in X(0)$ and the sets of top faces which disagree with the majority on v. By the variance method we know that almost all the vertices in the link of v see the right amount of top faces that disagree with the majority on v. Thus, if for a vertex $u \in X_v(0)$, where v is not bad neighbor of u, the function g_u disagrees with the majority on v, it must see a lot of top faces that disagree with the majority. By the variance method we know that this happens with a small probability. Details follows.

For any vertex $v \in X(0)$ define the function $h_v : X_v(d-1) \to \{0,1\}$ by

$$h_v(\sigma) = \begin{cases} 1 & f_{v\sigma}(v) \neq \mathrm{maj}(v), \\ 0 & f_{v\sigma}(v) = \mathrm{maj}(v). \end{cases}$$

Note that $\mathbb{E}_{\sigma\in X_v(d-1)} h_v(\sigma) \leq 1/2$. Note also that for any vertex $u \in X_v(0)$,

$$g_u(v) \neq \mathrm{maj}(v) \quad \wedge \quad v \notin B_u \quad \Rightarrow \quad \mathop{\mathbb{E}}_{\sigma\in X_{uv}(d-2)} h_v(u\sigma) > \frac{7}{10}.$$

Therefore,

$$\Pr_{u\in X_v(0)}[g_u(v) \neq \mathrm{maj}(v) \ \wedge \ v \notin B_u] \leq \Pr_{u\in X_v(0)}\left[\mathop{\mathbb{E}}_{\sigma\in X_{uv}(d-2)} h_v(u\sigma) - \mathop{\mathbb{E}}_{\sigma\in X_v(d-1)} h_v(\sigma) > \frac{1}{5}\right].$$

By corollary 3.6

$$\mathop{\mathrm{Var}}_{u\in X_v(0)} \mathop{\mathbb{E}}_{\sigma\in X_{uv}(d-2)} h_v(u\sigma) \leq \frac{8}{5(d-1)} \mathop{\mathbb{E}}_{\sigma\in X_v(d-1)} h_v(\sigma).$$

Thus, by Chebyshev inequality

$$\Pr_{u\in X_v(0)}\left[\mathop{\mathbb{E}}_{\sigma\in X_{uv}(d-2)} h_v(u\sigma) - \mathop{\mathbb{E}}_{\sigma\in X_v(d-1)} h_v(\sigma) > \frac{1}{5}\right] \leq \frac{40}{d-1} \mathop{\mathbb{E}}_{\sigma\in X_v(d-1)} h_v(\sigma).$$

Averaging over all $v \in X(0)$ yields

$$\begin{aligned}\mathop{\mathbb{E}}_{v\in X(0)} \Pr_{u\in X_v(0)}[g_u(v) \neq \mathrm{maj}(v) \ \wedge \ v \notin B_u] &\leq \frac{40}{d-1} \mathop{\mathbb{E}}_{v\in X(0)} \mathop{\mathbb{E}}_{\sigma\in X_v(d-1)} h_v(\sigma) \\ &\leq 80\Big(1 + \frac{3}{d/\ell - 1}\Big)\frac{\varepsilon}{d-1},\end{aligned}$$

where the last inequality follows by claim 4.2. ◁

4.3 Proofs of Claims 4.6 and 4.7

Both of these claims follow immediately by the variance method. In claim 4.6 we use the small variance of the function that measures the disagreement probability on a face $\tau \in X(\ell)$. Since the variance of the average of this function on the vertices is small, almost all the vertices have disagreement close to the expectation.

In claim 4.7 we use the small variance of the indicator function in the link of a vertex $u \in X(0)$, which indicates whether a top face disagrees with the function g_u. Since the variance of the average of this function on the vertices is small, almost all the vertices in the link of u see a small amount of top faces which disagree with g_u, i.e., almost all the vertices in the link of u are not bad neighbors. Details follow.

Proof of Claim 4.6. Note that $\mathbb{E}_{u\in X(0)} \varepsilon_u \leq \varepsilon$ and assume that $\varepsilon \leq 1/10c$. Thus, for a vertex $u \in X(0)$ to not be in S, it has to deviate from its mean by more than $1/10c$. By corollary 3.6

$$\mathop{\mathrm{Var}}_{u\in X(0)} \varepsilon_u \leq \frac{8}{5\ell} \mathop{\mathbb{E}}_{\tau\in X(\ell)} \Pr_{\sigma_1,\sigma_2\in X_\tau(d-\ell-1)}[f_{\tau\sigma_1}|_\tau \neq f_{\tau\sigma_2}|_\tau] \leq \frac{8\varepsilon}{5\ell}.$$

Thus, by Chebyshev inequality

$$\Pr_{u\in X(0)}[u \notin S] \leq \Pr_{u\in X(0)}\left[\varepsilon_u - \mathop{\mathbb{E}}_{u'\in X(0)} \varepsilon_{u'} > \frac{1}{10c}\right] \leq \frac{160c^2\varepsilon}{\ell}.$$ ◁

Proof of Claim 4.7. Consider a vertex $u \in S$. Since X_u is a c-agreement expander, there exists a local function $g_u : X_u(0) \to \{0,1\}$ such that

$$\Pr_{\sigma\in X_u(d-1)}[f_{u\sigma}|_\sigma \neq g_u|_\sigma] \leq c\varepsilon_u.$$

Define the indicator function $h_u : X_u(d-1) \to \{0,1\}$ by

$$h_u(\sigma) = \begin{cases} 1 & f_{u\sigma}|_\sigma \neq g_u|_\sigma, \\ 0 & f_{u\sigma}|_\sigma = g_u|_\sigma. \end{cases}$$

By the definition of S, $\mathbb{E}_{\sigma\in X_u(d-1)} h_u(\sigma) \le c\varepsilon_u \le 1/5$. Thus, in order for a vertex $v \in X_u(0)$ to be in B_u, it has to deviate from its mean by more than $1/10$. By corollary 3.6

$$\operatorname*{Var}_{v\in X_u(0)} \mathop{\mathbb{E}}_{\sigma\in X_{uv}(d-2)} h_u(v\sigma) \le \frac{8}{5(d-1)} \mathop{\mathbb{E}}_{\sigma\in X_u(d-1)} h_u(\sigma) \le \frac{8c\varepsilon_u}{5(d-1)}.$$

Thus, by Chebyshev inequality

$$\Pr_{v\in X_u(0)}[v \in B_u \mid u \in S] \le \Pr_{v\in X_u(0)}\left[\mathop{\mathbb{E}}_{\sigma\in X_{uv}(d-2)} h_u(v\sigma) - \mathop{\mathbb{E}}_{\sigma\in X_u(d-1)} h_u(\sigma) > \frac{1}{10}\right] \le \frac{160c\varepsilon_u}{d-1}.$$

◁

References

1. Boaz Barak, Pravesh K. Kothari, and David Steurer. Small-Set Expansion in Shortcode Graph and the 2-to-2 Conjecture. In *10th Innovations in Theoretical Computer Science Conference, ITCS 2019, January 10-12, 2019, San Diego, California, USA*, pages 9:1–9:12, 2019.
2. Yotam Dikstein and Irit Dinur. Agreement testing theorems on layered set systems. In *Proceedings of the 60th Annual Symposium on Foundations of Computer Science (FOCS)*, 2019.
3. Irit Dinur, Yuval Filmus, and Prahladh Harsha. Analyzing Boolean functions on the biased hypercube via higher-dimensional agreement tests. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2124–2133, 2019.
4. Irit Dinur and Tali Kaufman. High dimensional expanders imply agreement expanders. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 974–985, 2017.
5. Irit Dinur, Subhash Khot, Guy Kindler, Dor Minzer, and Muli Safra. Towards a proof of the 2-to-1 games conjecture? In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 376–389, 2018.
6. Irit Dinur and David Steurer. Direct product testing. In *2014 IEEE 29th Conference on Computational Complexity (CCC)*, pages 188–196, 2014.
7. S. Evra and T. Kaufman. Bounded degree cosystolic expanders of every dimension. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 36–48, 2016.
8. Subhash Khot, Dor Minzer, and Muli Safra. On independent sets, 2-to-2 games, and Grassmann graphs. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 576–589, 2017.
9. Subhash Khot, Dor Minzer, and Muli Safra. Pseudorandom sets in grassmann graph have near-perfect expansion. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 592–601, 2018.
10. A. Lubotzky. High Dimensional Expanders. *arXiv*, 2017. `arXiv:1712.02526`.
11. A. Lubotzky, R. Phillips, and P. Sarnak. Ramanujan graphs. *Combinatorica*, 8(3):261–277, 1988.
12. A. Lubotzky, B. Samuels, and U. Vishne. Explicit constructions of Ramanujan complexes of type $\widetilde{A}_d$. *European Journal of Combinatorics*, 26(6):965–993, 2005.

MPC for MPC: Secure Computation on a Massively Parallel Computing Architecture

T-H. Hubert Chan
The University of Hong Kong, Hong Kong
hubert@cs.hku.hk

Kai-Min Chung
Academia Sinica, Taipei City, Taiwan
kmchung@iis.sinica.edu.tw

Wei-Kai Lin
Cornell University, Ithaca, NY, USA
wklin@cs.cornell.edu

Elaine Shi
Cornell University, Ithaca, NY, USA
elaine@cs.cornell.edu

Abstract

Massively Parallel Computation (MPC) is a model of computation widely believed to best capture realistic parallel computing architectures such as large-scale MapReduce and Hadoop clusters. Motivated by the fact that many data analytics tasks performed on these platforms involve sensitive user data, we initiate the theoretical exploration of how to leverage MPC architectures to enable efficient, privacy-preserving computation over massive data. Clearly if a computation task does not lend itself to an efficient implementation on MPC even without security, then we cannot hope to compute it efficiently on MPC with security. We show, on the other hand, that *any task that can be efficiently computed on MPC can also be securely computed with comparable efficiency*. Specifically, we show the following results:

- any MPC algorithm can be compiled to a *communication-oblivious* counterpart while asymptotically *preserving its round and space complexity*, where communication-obliviousness ensures that any network intermediary observing the communication patterns learn no information about the secret inputs;
- assuming the existence of Fully Homomorphic Encryption with a suitable notion of compactness and other standard cryptographic assumptions, any MPC algorithm can be compiled to a secure counterpart that defends against an adversary who controls not only intermediate network routers but additionally up to $1/3 - \eta$ fraction of machines (for an arbitrarily small constant η) – moreover, this compilation preserves the round complexity tightly, and preserves the space complexity upto a multiplicative security parameter related blowup.

As an initial exploration of this important direction, our work suggests new definitions and proposes novel protocols that blend algorithmic and cryptographic techniques.

2012 ACM Subject Classification Theory of computation → Cryptographic protocols

Keywords and phrases massively parallel computation, secure multi-party computation

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.75

Funding *T-H. Hubert Chan*: T-H. Hubert Chan was partially supported by the Hong Kong RGC under the grant 17200418.
Kai-Min Chung: This research is partially supported by the Academia Sinica Career Development Award under Grant no. 23-17 and Ministry of Science and Technology, Taiwan, under Grant no. MOST 106-2628-E-001-002-MY3.
Wei-Kai Lin: This work is supported by the DARPA Brandeis award.
Elaine Shi: This work is supported in part by NSF CNS-1453634, an ONR YIP award, a Packard Fellowship, and an IARPA HECTOR grant under a subcontract from IBM.

Acknowledgements We gratefully thank Xiaorui Sun for his patient and detailed explanations about the Massively Parallel Computation (MPC) model, for answering many of our technical questions, and for suggesting an idea to transform an MPC protocol without the s-sender-constraint to one that has.

1 Introduction

In the past decade, parallel computation has been widely adopted to manipulate and analyze large-scale data-sets, and numerous programming paradigms such as MapReduce, Hadoop, and Spark have been popularized to help program large computing clusters. This has partly served as a driving force for the algorithms community to better understand the power and limitations of such parallel computation models. The first theoretic model capturing modern parallel computation frameworks was proposed by Karloff, Suri, and Vassilvitskii [79]. Since then, a flurry of results have appeared proposing refinements to the model as well as novel algorithms with better asymptotical and practical efficiency [4, 40, 76, 79, 80, 83, 89, 99].

With these amazing efforts, the community has converged on a model called *Massively Parallel Computation* (MPC), which is believed to best capture large computing clusters (e.g., those operated by companies like Google and Facebook) consisting of a network of Random-Access Machines (RAMs), each with a somewhat considerable amount of local memory and processing power – and yet each individual machine is not powerful enough to store the massive amount of data available. In the MPC model of computation, we typically assume a total of N data records where N is rather large (in practice, the data size can range from tens of terabytes to a petabyte). Each machine can locally store only $s = N^{\varepsilon}$ amount of data for some constant $\varepsilon \in (0, 1)$; and the number of machines $m \geq N^{1-\varepsilon}$ such that all machines can jointly store the entire dataset. In many MPC algorithms it is also desirable if $m \cdot s = \widetilde{O}(N)$ or $m \cdot s \leq N^{1+\theta}$ for some small constant $\theta \in (0, 1)$, i.e., the total space consumed should not be too much larger than the dataset itself [2, 5, 79, 83].

In the standard algorithms literature on MPC, a key focus has been the design of algorithms that minimize the *round complexity*, partly by harnessing the reasonably large local memory that is available on each processing unit. Using round complexity as a primary metric, a rich set of computational tasks have been investigated in the MPC model, including graph problems [2, 4–6, 10–13, 15, 19, 20, 29, 36, 40, 52, 60, 63, 81, 83, 93, 97], clustering [16, 17, 46, 61, 107] and submodular function optimization [41, 47, 80, 90]. Interestingly, it is also known that a number of tasks (such as sorting, parity, minimum spanning tree) that either suffered from an almost logarithmic depth lower bound on a classical Parallel Random-Access Machine (PRAM) now can be accomplished in $O(1)$ or sublogarithmic number of rounds on an MPC framework [40, 67, 79, 93]. Note that a PRAM assumes that each processing unit has $O(1)$ local storage and thus PRAM is not the best fit for capturing modern parallel computing clusters.

1.1 Privacy-Preserving Data Analytics on MPC Frameworks

In this paper, we are the first to ask the question, how can we leverage an MPC cluster to facilitate *privacy-preserving*, large-scale data analytics? This question is of increasing importance because numerous data analytics tasks we want to perform on these frameworks involve sensitive user data, e.g., users' behavior history on websites and/or social networks, medical records, or genomic data. We consider two primary scenarios:

Scenario 1: MPC with secure end-points

We may consider an MPC framework where the end-points[1] are secured by trusted processors such as Intel SGX. Without loss of generality, we may assume that data is encrypted in memory or in transit such that all the secure processors can decrypt them inside a hardware-enabled sandbox where computation will take place (to achieve this the secure processors may employ standard cryptographic techniques to perform a secure handshake and establish a shared encryption key). Finally, when the computation result is ready, the secure processors can send encrypt results to an authorized analyst who has a corresponding key to decrypt the results.

In this scenario, we consider a network adversary (e.g., compromised operating systems, intermediate network routers, or system administrators) that can observe the communication patterns between the end-points, and we would like to make sure that the MPC algorithm's *communication pattern leak no information* about the secret data.

Note that in Scenario 1, we make a simplifying assumption that the adversary cannot observe the memory access patterns on end-points: since known works on efficient Oblivious RAM (ORAM) [64, 66, 100, 102] have in principle solved this problem; not only so, in recent work the first secure processor with ORAM support has been taped out [50, 51, 98].

Scenario 2: MPC with insecure end-points

In the second scenario, imagine that the adversary controls not only the intermediate network routers but also some of the end-points. For example, the end-points may be legacy machines without trusted hardware support, and they may have a comprised operating system. The adversary may also be a rogue system administrator who has access to a subset of the machines. We assume, however, that the adversary controls only a small subset of the end-points – such an assumption is reasonable if the end-points are hosted by various organizations, or by the same organization but in different data centers, or if they have diverse hardware/software configurations such that they are unlikely to be all hit by the same virus.

In this scenario, we would like to obtain a guarantee similar to that of cryptographic Secure Multi-Party Computation (SMPC)[2], i.e., an adversary controlling a relatively small subset of the machines cannot learn more information beyond what is implied by the union of the corrupt machine's outputs. Note that in this scenario, all machines' outputs can also be in encrypted format such that only an authorized data analyst can decrypt the result; or alternatively, secret shared such that only the authorized data analyst can reconstruct – in this case, the adversary should not be able to learn anything at all from the computation.

With the aforementioned scenarios in mind, we ask, what computation tasks can be securely and efficiently computed on an MPC architecture? Clearly, if a computation task does not lend itself to efficient computation on MPC *even without security*, we cannot hope to attain an efficient and secure solution on the same architecture. Therefore, the best we can hope for is the following: for computational tasks that indeed have efficient MPC algorithms, we now want to compute the same task securely on MPC while preserving the efficiency of the original insecure algorithm. In other words, we ask the following important question:

> Can we *securely* evaluate a function f on an MPC framework, while paying not too much more overhead than evaluating f *insecurely* on MPC?

[1] By end-points, we mean the machines, as opposed to the communication/network.
[2] In this paper, to avoid confusion, we use SMPC to mean cryptographic Secure Multi-Party Computation; and we use MPC to mean Massively Parallel Computation.

1.2 Our Results and Contributions

Conceptual and definitional contributions

We initiate the exploration of how to leverage Massively Parallel Computation (MPC) to secure large-scale computation. The widespread empirical success of MPC frameworks in practice, as well as the typically sensitive nature of the data involved in the analytics provide strong motivations for our exploration. We hope that the formal definitions and theoretical feasibility results in our work will encourage future work along this direction, and hopefully leading to practical solutions for privacy-preserving large-scale data analytics.

In comparison, although earlier works originating from the cryptography community have explored secure computation on parallel architectures, most known results [3, 24, 25, 32, 34, 35, 37, 38, 87, 92] adopt PRAM as the model of computation. As discussed later in Section 2, known results specialized for PRAMs do not directly lead to the type of results in this paper due to the discrepancy both in model and in metrics. As mentioned, the PRAM model is arguably a mismatch for the parallel computing architectures encountered in most practical scenarios. This is exactly why in the past decade, the algorithms community have focused more on the modern MPC model which better captures the massively parallel computing clusters deployed by companies like Google and Facebook. Therefore, we hope that our work will bring the MPC model to the attention of the cryptography community for exploring parallel secure computation.

We proceed to present a summary of our major results.

Communication-oblivious MPC

To securely compute a function in Scenario 1 and as a stepping stone towards enabling Scenario 2, we first define a notion of *communication obliviousness* for MPC algorithms. Informally speaking, we want that the adversary learns no information about the secret inputs after observing an MPC algorithm's communication patterns. In this paper we require a very strong notion of communication obliviousness, where we simply want that the MPC algorithm's communication patterns be deterministic and input independent[3].

We prove that any MPC algorithm Π can be compiled to a communication-oblivious counterpart while asymptotically preserving its round complexity and space consumption.

▶ **Theorem 1** (Communication-oblivious MPC algorithms). *Suppose that $s = N^\varepsilon$ and that m is upper bounded by a fixed polynomial in N. Given any MPC algorithm Π that completes in R rounds where each of the m machines has s local space, there is a communication-oblivious MPC algorithm $\widetilde{\Pi}$ that computes the same function as Π except with $\exp(-\Omega(\sqrt{s}))$ probability, and moreover $\widetilde{\Pi}$ completes in $O(R)$ rounds, and consuming $O(s)$ space on each of the m machines. Furthermore, only $O(m \cdot s)$ amount of data are communicated in each round in an execution of $\widetilde{\Pi}$.*

Note that numerous interesting MPC algorithms known thus far have total communication at least $\Omega(R \cdot m \cdot s)$ where R denotes the protocol's round complexity (ignoring polylogarithmic factors) [6,59,62,67,77], and for this class of MPC algorithms, our compilation also introduces very little asymptotical communication overhead.

[3] We stress that the algorithm itself can be randomized, we just want its communication patterns to be deterministic and fixed a-priori.

Secure multi-party computation for MPC

We now turn to Scenario 2. In this setting security means that a relatively small corrupt coalition cannot learn anything more beyond the coalition's joint outputs. We now ask the following natural question:

> Can we compile any MPC protocol to a *secure* counterpart (where security is in the above sense), allowing only $O(1)$ blowup in round complexity and security parameter related blowup in the total space[4]?

We answer this question affirmatively assuming that the adversary controls only $\frac{1}{3} - \eta$ fraction of machines for any arbitrarily small constant η. Note that $\frac{1}{3}$ is necessary since the MPC model assumes a point-to-point channel without broadcast, and in this model it is known that secure computation cannot be attained in the presence of $\frac{1}{3}$ or more corruptions [48, 82].

To achieve this result, we need to assume the existence of a common random string and appropriate cryptographic hardness assumptions, including the Learning With Errors (LWE) assumption, enhanced trapdoor permutations, as well as the existence of a Fully Homomorphic Encryption (FHE) scheme with an appropriate notion of *compactness* [55, 58]. It is well-known that such compact FHE schemes are implied by a suitable circularly secure variant of the LWE assumption [58], although our compiler can work in general given any such compact FHE scheme (not necessarily based on LWE). Our result is summarized in the following theorem:

▶ **Theorem 2** (Secure computation for MPC). *Assume the existence of a common random string, the Learning With Errors (LWE) assumption, enhanced trapdoor permutations, as well as the existence of an FHE scheme with a suitable notion of compactness (see Appendix A.1 for a formal definition of compactness). Suppose that $s = N^{\varepsilon}$ and that m is upper bounded by a fixed polynomial in N. Let κ denote a security parameter, and assume that $s \geq \kappa$. Given any MPC algorithm Π that completes in R rounds where each of the m machines has s local space, there is an MPC algorithm $\widetilde{\Pi}$ that securely realizes the same function computed by Π in the presence of an adversary that statically corrupts at most $\frac{1}{3} - \eta$ fraction of the machines for an arbitrarily small constant η. Moreover, $\widetilde{\Pi}$ completes in $O(R)$ rounds, consumes at most $O(s) \cdot \mathsf{poly}(\kappa)$ space per-machine, and incurs $O(m \cdot s) \cdot \mathsf{poly}(\kappa)$ total communication per round.*

Now, one interesting question is whether the cryptographic assumptions we rely on in the above theorem can be avoided. We show that if one can indeed achieve the same result with *statistical* security, then it would imply (at least partial) solutions to long-standing open questions in the cryptography literature. Specifically, in Appendix B, we show that if we could construct such a compiler, it would immediately imply a constant-round Secure Multi-Party Computation protocol for a broad class of circuits that can be computed in small space, achieving *total* communication complexity that is (significantly) sublinear in the circuit size, regardless of the number of parties. As noted in numerous works [26, 39, 44, 45], the existence of such constant-round, sublinear-communication multi-party computation (for circuits) with *statistical* security has been a long-standing open problem, even for special (but nonetheless broad) classes of circuits.

[4] Since many well-known MPC algorithms [2, 4–6, 10–13, 15–17, 19, 20, 29, 36, 40, 41, 46, 47, 52, 60, 61, 63, 80, 81, 83, 90, 93, 97, 107] incur only constant to sub-logarithmic rounds, we would like to preserve the round complexity tightly; and thus we do not allow a security parameter related blowup for round complexity.

Interestingly, we note that barring strong assumptions such as Indistinguishable Obfuscation [53], the only known approach to construct constant-round, sublinear-communication *multi*-party computation for circuits of unbounded polynomial size is also through compact FHE [55, 58]. We stress, however, that even with a compact FHE scheme, constructing our "MPC to SMPC-for-MPC" compiler is non-trivial and require the careful combination of several techniques.

2 Technical Roadmap

We now present a succinct and informal technical roadmap to capture our main ideas and new techniques.

Recall that in the MPC model of computation, there are m machines each with s local space. All machines will jointly compute a function over a large input containing N words. We assume that $s = N^{\varepsilon}$ for some constant $\varepsilon \in (0, 1)$, and that $m \in [N^{1-\varepsilon}, \mathsf{poly}(N)]$. Note that although our results will hold as long as m is upper bounded by some polynomial function in N, in known MPC algorithms typically we desire that $m \cdot s$ is not too much greater than N. At the beginning of the first round, every machine receives an input whose size is bounded by s. In every other round, each machine begins by receiving incoming messages from the network, and it is guaranteed that no more than s words will be received such that the machine can write them down in its local memory – henceforth this is referred to as the *s-receiver-constraint*. After receiving the inputs or the network messages, all machines perform local computation, and then send messages to other machines. These messages will then be received at the beginning of the next round.

As explained earlier, in the algorithms literature on MPC, the primary metric of performance is the algorithm's *round complexity* [2, 4–6, 10–13, 15–17, 19, 20, 29, 36, 40, 41, 46, 47, 52, 60, 61, 63, 80, 81, 83, 90, 93, 97, 107].

2.1 Achieving Communication Obliviousness: Oblivious Routing

Many known MPC algorithms are not communication oblivious, and thus the communication patterns of these algorithms can leak information about the secret inputs. For example, many graph algorithms for MPC have communication patterns that will leak partial information about the structure and properties of the graph, such as the degrees of nodes or the connectivity between vertices [6, 40, 62, 81, 93].

Our goal is to compile an MPC algorithm to a communication-oblivious counterpart while preserving its round and space complexity. To achieve this, we will compile each communication round of the original MPC to a constant-round, oblivious protocol that accomplishes the same message routing. Interestingly, the compiled protocol will respect communication-obliviousness in a very strong sense: *its communication patterns are deterministic and independent of the input.*

Sender and receiver constraints

In the MPC model of computation [6, 40, 93], we typically have that in each round, each machine sends at most s words and receives at most s words – henceforth these are referred to as the s-sender-constraint and the s-receiver-constraint respectively. Note that if a sender wants to send the same word to two machines, it is counted twice.

Oblivious routing

Oblivious routing basically aims to obliviously realize the message routing as long as both the s-sender- and s-receiver-constraints are satisfied.

More specifically, suppose that each machine receives at most s send-instructions as input, where each send-instruction contains an outgoing word to be sent and a destination machine for the outgoing word. The joint inputs of all machines guarantee that each machine will receive no more than s words. How can we design a constant-round, communication-oblivious protocol that routes the outgoing words to the appropriate destinations?

▶ Remark 3. It seems that some MPC works in the algorithms literature respect only the s-receiver constraint but not the s-sender constraint. We think most likely, the folklore understanding is that as long as we assume the s-receiver constraint, whether or not there is an s-sender constraint do not really affect the expressive power of the computation model. For completeness, in Appendix C, we describe a round- and space-preserving transformation that compiles any MPC protocol that satisfies only the s-receiver-constraint to one that satisfies both $O(s)$-receiver- and $O(s)$-sender-constraints. This means that all of our results would be applicable to MPC algorithms that satisfy only the s-receiver-constraint but not the s-sender-constraint.

Background

Our approach is partly inspired by algorithmic techniques from the recent Oblivious RAM and oblivious sorting line of work [7, 33, 49, 96, 101]. Specifically, these works propose a RAM algorithm with a fixed memory access pattern that routes elements to random buckets and succeeds with $1 - \exp(-\Omega(Z))$ probability where Z denotes each bucket's capacity (and assuming that the total number of elements is polynomially bounded). To accomplish this, imagine that initially all N elements are divided into $2N/Z$ buckets each of capacity Z such that each bucket is at most half-loaded. Every element is assigned a random label declaring which bucket it wants to go to. Now, these prior works rely on a *logarithmic-depth*, butterfly network of buckets and move the elements along this butterfly network based on their respective labels, until eventually every element falls into its desired bucket – this is guaranteed to succeed with $1 - \exp(-\Omega(Z))$ probability where a failure can only occur if in the middle some bucket in the butterfly network exceeds its capacity – henceforth this is said to be an overflow event.

In summary, the insight we can gain from this elegant construction is the following: roughly speaking, a butterfly network of super-logarithmically sized buckets can obliviously route elements to their desired buckets in the final layer with a deterministic communication pattern (determined by interconnections in the butterfly network); but to ensure correctness, i.e., to ensure that overflow does not happen except with negligible probability, the elements should have random destination labels to achieve good load-balancing properties.

A first attempt

Our idea is to use such a butterfly-network to route the words to their destinations – specifically, one can imagine that each bucket is relatively small such that every machine holds $\Theta(\frac{s}{Z})$ of the resulting buckets; and moreover, the buckets are numbered $1, 2, \ldots, O(m \cdot s/Z)$ respectively. For convenience, we will use the term "element" to mean an outgoing word to be sent. Since every sender already knows the destination machines for each of its input elements, it can basically assign the element to a random bucket within the destination

machine. Henceforth, by "destination label", we mean the index of the destination bucket (as opposed to the destination machine). We are, however, faced with two challenges which prevent us from adopting the known approach in its current form:

- *Load balancing challenge:* First, in our setting, the destination labels of the input elements are not completely random; and thus the load-balancing properties that hold in earlier works [7, 33, 49] for randomly assigned labels no longer hold in our case;
- *Round complexity challenge:* Second, the natural way to adopt the butterfly network is to assign to each machine an appropriate subset of $\Theta(\frac{s}{Z})$ buckets in each layer of the network. However, if $\frac{s}{Z} = \Theta(1)$, then we will incur logarithmic number $\Omega(\log m)$ of rounds (corresponding exactly to the depth of the butterfly network). Later, we shall set Z to be small enough (e.g., $Z = O(\sqrt{s})$ in size) to reduce the number of rounds.

Overcoming the load balancing challenge

To overcome the load balancing challenge, our idea is to run this butterfly network twice: the first time we use it to route every element a *random* destination bucket just like in the earlier works; and the second time we use it to route every element to a random destination bucket within the destination machine they originally wanted to go to. At the end of the second phase, every element will be routed to the machine it wants to go to.

For the first phase, we can rely on the same load balancing arguments as in the previous works [7, 33, 49] to prove that overflow events happen only with negligible probability. For the second phase, we will prove a new stochastic bound showing that the same load-balancing properties hold with a different starting condition (see Section 4.4): *i)* the starting condition must satisfy the s-receiver-constraint; and *ii)* initially the elements are assigned to random input buckets, which is guaranteed by phase 1.

It remains to overcome the round complexity challenge which we explain below.

Overcoming the round complexity challenge

The earlier works rely on a 2-way butterfly network where in each layer i, a local 2-way routing decision is made for every element based on the i-th bit of its destination label. In our new construction, we will compress r layers of work in the original butterfly to a single round, exploiting the fact that each machine local space to store roughly 2^r buckets, where $2^r = \Theta(\frac{s}{Z})$.[5] In this way our new approach requires $O((\log \frac{N}{Z})/r) = O(1/\varepsilon)$ rounds for $Z = O(\sqrt{s})$ and $s = N^{\varepsilon}$. Effectively, in each round i, each machine would be looking at the i-th r-bit-group of an element's label to make a 2^r-way routing decision for the element.

To make this idea work, the crux is to show that at the end of every round (corresponding to r layers in the old 2-way butterfly), there is a communication-efficient way for the machines to exchange messages and rearrange their buckets such that the interconnections in the graph are "localized" in the next round too. In this way, within each round, each machine can perform 2^r-way routing on its local elements, simulating r layers of the old 2-way butterfly, without communicating with any other machine. Fortunately, we can accomplish this by exploiting the structure of the butterfly network. We defer the algorithmic details and proofs to Section 4.

[5] Our techniques remotely reminiscient of the line of work on external-memory ORAM constructions with large, N^{ϵ} CPU private cache [33, 68, 69, 96, 101] – however, all previous works consider a *sequential* setting.

2.2 SMPC for MPC

2.2.1 Informal Problem Statement

We now turn our attention to Scenario 2 where the adversary controls not just the intermediate network routers but also a subset of the machines involved in large-scale computation. As before, given a computation task f that has an efficient but insecure MPC algorithm, we now would like to *securely* realize f also using the MPC model while preserving its efficiency. Here, our security goal is to guarantee that the adversary learns nothing more than what is already implied by the joint outputs of corrupt machines. Such a notion of security can be formally defined using a standard simulation-based paradigm [30], requiring that the real-world protocol must "securely emulate" an ideal-world protocol where all machines simply forward their inputs to a trusted ideal functionality who performs the desired computation task and forwards the outputs to each machine. Intuitively, we require that for any real-world attack that can be performed by a polynomially bounded adversary, there is an ideal-world adversary that can essentially implement the same attack in the ideal world. We refer the reader to Section 5 for a formal definition. Note that our definition follows the same style as the Universal Composition framework [30]. Henceforth, a secure multi-party computation protocol satisfying the aforementioned security notion is said to be an "SMPC-for-MPC" protocol.

Before we describe how to solve this problem, we need to clarify a few points about the model of execution as we marry SMPC and MPC. Since we now have corrupt machines and corrupt machines are allowed to send arbitrary messages to any machine, we can no longer guarantee that each honest machine receive at most s words. Instead, we require that at the end of the every round $r-1$, every machine can write down in its local memory a receiving schedule for round r of the form $\{(f_j, w_j)\}_j$, where $f_j \in [m]$ denotes the index of a sender to anticipate a message from in round r and w_j denotes the number of words expected from f_j. In this way, an honest machine will only save the first w_j words received from every anticipated sender f_j contained in this receiving schedule; all other received messages are discarded immediately.

▶ Remark 4. Recall that since we showed how to compile any MPC protocol to one that has a fixed communication schedule while asymptotically preserving round and space complexity (Section 2.1), requiring that receivers be able to anticipate their receiving schedule does not reduce the expressive power of the underlying computational model. However, somewhat more subtlely, we do not insist that the communication patterns be deterministic (i.e., fully fixed a-priori) for our SMPC-for-MPC protocols in order not to rule out interesting SMPC-for-MPC protocols – it suffices for an honest machine to anticipate its receiving schedule of some round right before the round starts.

2.2.2 MPC to "SMPC-for-MPC" Compiler

Recall that we would like the compiled SMPC-for-MPC protocol to tightly preserve the original MPC program's round complexity, and for per-machine space we only allow a security parameter related blowup. Although in the cryptography literature, secure multi-party computation (SMPC) techniques have been explored for various models of computation including circuits [21, 55, 65, 105, 106], RAM [1, 54, 56, 57, 71, 85, 86, 104], and PRAM [24, 25, 32, 34, 35, 37, 38, 87, 92], none of the existing approaches can satisfy our efficiency requirements due to a discrepancy in both model and metric of performance.

First, any approach whose round complexity depends at least on the depth of the circuit [9, 22, 65, 78] or on the parallel runtime of a PRAM [24] can be immediately ruled out, since MPC protocols can have high circuit/PRAM depth (e.g., if each machine's local

computation is of a sequential nature). We thus turn our attention to known approaches that achieve small round complexity. For example, a line of works [18, 42, 84] showed a technique where multiple machines jointly garble a circuit/PRAM in constant number of rounds, then each individual machine simply evaluates the garbled circuit/PRAM locally. Although such a protocol would indeed complete in a small number of rounds, the garbled circuit/PRAM's size would be at least linear in the total computation time of all parties (unless we make strong assumptions such as Indistinguishable Obfuscation [53]). This means that the per-machine space blowup will be proportional to the number of machines m. Similarly, other constant-round approaches, including those relying on Threshold Fully Homomorphic Encryption [8] or Multi-Key Fully Homomorphic Encryption [14, 28, 72, 75, 91], also do not directly work due to a linear-in-m blowup in the per-machine space, e.g., due to the need to store all machines' encrypted inputs (although later in our construction we will indeed leverage FHE-based MPC techniques as a building block, although this MPC will only be run among small committees).

Our approach

Henceforth we assume that the adversary controls only $\frac{1}{3} - \eta$ fraction of the machines where η may be an arbitrarily small constant. As explained earlier, this is nearly optimal tolerance in a point-to-point network since an honest fraction of at least $\frac{2}{3}$ is necessary for realizing broadcast in a point-to-point network. Without loss of generality, we may also assume that the original MPC has already been compiled to a communication-oblivious counterpart whose communication patterns are deterministic and input independent.

Our idea is to directly emulate the original (communication-oblivious) MPC round by round, by having a randomly elected small committee emulate each machine's actions in the original MPC. Henceforth the committee acting on behalf of machine i in the original MPC is called the i-th committee. As long as the committee size is polylogarithmic in the security parameter, except with negligible probability each committee must have at least a $\frac{2}{3}$-fraction of honest machines. Therefore, we may employ an SMPC that tolerates less than $\frac{1}{3}$ corruption (and satisfies suitable efficiency requirements to be described below) to securely emulate the following actions – note that in all cases this SMPC protocol will be executed among a small set of polylogarithmically many machines; and thus we call this SMPC building block CommitteeSMPC:

1. *Share input*: initially each machine i secret shares its input with the i-th committee; the secret sharing scheme must be *robust* such that correct reconstruction can be achieved in polynomial time even when corrupt machines provide false shares. Note that to achieve this we may run the aforementioned CommitteeSMPC protocol among machine i and the i-th committee;
2. *Local compute*: in every round's computation step, the i-th committee employ CommitteeSMPC to jointly evaluate what machine i's is supposed to locally compute in the original MPC, and the result is again robustly secret-shared among the committee;
3. *Communicate*: whenever machine i wants to send a message to machine j in the original MPC, now the i-th committee and the j-th committee employ CommitteeSMPC to accomplish this. At the end of this small protocol, every member of the j-th committee will obtain a *fresh* robust secret share of the message.

Instantiating CommitteeSMPC with suitable efficiency requirements

For our compilation to satisfy the stated efficiency goals, the CommitteeSMPC employed must meet certain efficiency requirements:

1. *Constant round:* the protocol must complete in constant number of rounds; and
2. *Weakly space efficient:* the space consumed by each machine in the CommitteeSMPC protocol asypmtotically matches the RAM-space-complexity of the function being evaluated, allowing only a security-parameter-related blowup[6]. In this paper, we assume that the RAM-space-complexity accounts for the space for writing down the RAM program's description and all inputs and outputs.

▶ Remark 5. Note that since the RAM-space-complexity accounts for writing down all machines' inputs and outputs, by definition the RAM-space-complexity of the function being evaluated incur a linear blowup in the number of machines. However, since CommitteeSMPC will only be run among at most 2 committees, the number of machines participating is small in our case. In fact, by the weak space efficiency condition, each machine's space complexity may sometimes need to be sublinear in the circuit size, runtime, or even depth of the function being evaluated (depending on the function being evaluated).

The only known approach for achieving these guarantees simultaneously is through Threshold Fully Homomorphic Encryption (TFHE) [8] or Multi-Key Threshold Fully Homomorphic Encryption (MTFHE) [14, 72, 75] with a suitable notion of *compactness* (to be explained shortly after). Roughly speaking, to perform an SMPC, the following takes place where the encryption scheme employed is (M)TFHE:

(a) possibly after a setup phase, each machine encrypts their local input using the (M)TFHE scheme and computes a zero-knowledge proof attesting to the well-formedness of the ciphertext; now the machine broadcasts the ciphertext as well as the proof;
(b) now each machine homomorphically evaluates an encryption of all machines' outputs;
(c) for every machine involved, compute the partial decryption share for that machine's encrypted output, along with a zero-knowledge proof attesting to the correctness of decryption; now send the partial decryption share and the proof to the corresponding machine.
(d) a machine finally reconstructs the output after collecting enough decryption shares.

If the (M)TFHE scheme employed is *compact* in the sense that the public key, secret key, and ciphertext sizes depend only on the security parameter κ but not the size or the depth of the circuit being evaluated, then we can show that in the above steps each machine needs only $O(m' \cdot s) \cdot \mathsf{poly}(\kappa)$ space where m' denotes the number of machines involved in the CommitteeSMPC protocol and s denotes the RAM-space-complexity of the function being evaluated. Specifically, recall that any RAM machine with space complexity s can be converted to a layered circuit with width s (and moreover the circuit can be generated and written down layer by layer consuming space proportional to the RAM's next-instruction circuit size). Thus in the above, Step (b) can be accomplished using $O(m' \cdot s) \cdot \mathsf{poly}(\kappa)$ space; and it is easy to verify that all other steps indeed consume at most $O(m' \cdot s) \cdot \mathsf{poly}(\kappa)$ too.

In existing (M)TFHE schemes [8, 14, 72], however, the key and ciphertext sizes are dependent on the depth of the circuit being evaluated and thus they do not satisfy the aforementioned compactness requirement. To make these schemes compact, we need to rely on the bootstrapping technique described in the original Gentry work on FHE [55]; and it is

[6] We call this notion weakly space efficient, since one can imagine a stronger notion requiring that the *total* space consumed by *all* parties asyptotically matches the RAM-space-complexity of the function being evaluted.

well-known that to get provable security with bootstrapping, we need to assume that the (M)TFHE scheme employed satisfies *circular security* – informally speaking, ciphertexts must nonetheless remain semantically secure even when we use the encryption scheme to encrypt the secret decryption key.

Since there are relatively few known (M)TFHE constructions [14,72], rather than assuming that the existing constructions are circularly secure, we would like to further hedge our bet. Later in our technical sections, we further relax the assumption and base our scheme instead on LWE and the existence of *any* compact FHE. Note that compact FHE is known to exist assuming circularly secure variants of LWE; but for our purpose, we can work with any compact FHE scheme including ones that depend on different algebraic assumptions.

2.3 Related Work

As mentioned earlier, the cryptography literature has extensively considered secure computation on an parallel architecture but most existing works focus on the PRAM model [3, 24, 25, 32, 34, 35, 37, 38, 87, 92]. Since most real-world large-scale parallel computation is now done on an MPC architecture, we hope that our work will bring the MPC computation model (which has been extensively studied in the algorithms literature) to the attention of the cryptography community. Besides the PRAM model, Parter and Yogev have considered secure computation on graphs in the so-called CONGEST model of computation [94, 95].

3 Preliminaries

3.1 Massively Parallel Computation Model

We now describe the *Massively Parallel Computation* (MPC) model. Let N be the input size in words where each word consists of $w = \Omega(\log N)$ bits, and $\varepsilon \in (0, 1)$ be a constant. The MPC model consists of m parallel machines, where $m \in [N^{1-\varepsilon}, \mathsf{poly}(N)]$ and each machine has a local space of $s = N^{\varepsilon}$ words. Hence, the total space of all machines is $m \cdot s \geq N$ words. Often in the design of MPC algorithms we also want that the total space is not too much larger than N, and thus many works [2, 5, 79, 83] assume that $m \cdot s = \widetilde{O}(N)$, or $m \cdot s = O(N^{1+\theta})$ for some small constant $\theta \in (0, 1)$. Henceforth the m machines are numbered $1, 2, \ldots, m$ respectively. The m machines are pairwise connected and every machine can send messages to every other machine.

In this paper we are interested in *protocols* (also called *algorithms*) in the MPC model. In this model, the computation proceeds in *rounds*. At the beginning of each round, if this is the first round then each machine receives N/m words as input; else each machine receives incoming messages from the network, and a well-formed MPC algorithm must guarantee that each machine receives at most s words since there is no additional space to store more messages. After receiving the incoming messages or inputs, every machine now performs local computation; and we may assume that the local computation is bounded by $\mathsf{poly}(s)$ and the choice of the polynomial poly is fixed once the parameters s and m are fixed. After completing the local computation, every machine may send messages to some other machines through a pairwise channel, and then all messages are received at the beginning of the next round. When the algorithm terminates, the result of computation is written down jointly by all machines, i.e., by concatenating the outputs of all machines. Every machine's output is also constrained to at most s words. An MPC algorithm may be *randomized*, in which case every machine has a sequential-access random tape and can read random coins from the random tape. The size of this random tape is not charged to the machine's space consumption.

In the standard literature on MPC, we are most concerned about the *round complexity* of an MPC algorithm. Specifically, this model has been of interest to the algorithms community since numerous tasks that are known to have logarithmic depth lower bounds on the classical Parallel Random-Access Machine (PRAM) model are known to have sublogarithmic- or even constant-round algorithms in the MPC model [40, 67, 79, 93].

Useful notations and conventions

We introduce a couple useful notations and conventions:

- Given an MPC algorithm Π, we use the notation $(y_1, y_2, \ldots, y_m) \leftarrow \Pi(x_1, x_2, \ldots, x_m)$ to denote a possibly randomized execution of the algorithm where each machine i's input is x_i and its output is y_i for $i \in [m]$.
- When we say the input to an MPC algorithm, we mean the concatenation of all machines' inputs. When we say that an input array I is *evenly spread* across the m machines, we mean that every machine but the last one obtains $\lfloor |I|/m \rfloor$ elements and the last machine obtains $|I| \mod m$ elements where $|I|$ denotes the total number of elements in I.
- We use the term *s-receiver-constraint* to refer to the requirement that a well-formed MPC algorithm must ensure that each machine receives no more than s words in each round. One may also consider, symmetrically, an *s-sender-constraint*, that is, in each round every machine can send at most s words (where sending the same word to two machines counts twice). Many MPC algorithms in the literature respect both constraints. However, it seems that some other works in the MPC literature require only the *s-receiver-constraint* but not the *s-sender-constraint*.

  It turns out that the two modeling approaches are equivalent in terms of expressive power as we show in Appendix C. Therefore, in the main body of the paper, we simply assume that both constraints must be respected, but our results also extend to MPC algorithms that respect only the s-receiver-constraint.
- Like in the standard algorithms literature on MPC, we are concerned about *asymptotical* complexity. For this reason, whenever convenient, we shall assume that each machine is allowed $O(s)$ local space rather than s. Similarly, the s-receiver-constraint is sometimes interpreted as each machine receiving no more than $O(s)$ data per-round.
- We assume that the original MPC protocol to be compiled runs in a *fixed* number of rounds. If not, we can always pad its round complexity to be the worst case. This ensures that no information will be leaked through the number of rounds.

3.2 Communication-Oblivious MPC Algorithms

Communication-oblivious MPC algorithms

To compile an MPC algorithm to a secure counterpart, we go through an important stepping stone where we first compile the original MPC algorithm to a communication-oblivious counterpart. As mentioned earlier in Section 1, communication-oblivious MPC algorithms are also interesting in their own right, e.g., for MPC clusters where the end points are secured with secure processors such as Intel SGX, such that the adversary can observe only the communication patterns.

Intuitively, an MPC algorithm is said to be communication-oblivious, iff an adversary who can observe the communication patterns of the machines learn nothing about the secret input. When we execute an MPC algorithm on some input I, the *communication pattern* incurred in the execution is the concatenation of the following:

1. For each round r, a matrix $P_r \in [s]^{m \times m}$ where $P_r[i,j] \in [s]$ indicates how many words machine i sends to machine j in round r;
2. An ordered list containing the number of words each machine outputs at the end of the algorithm.

In this paper, we define a very strong notion of communication obliviousness: we say that an MPC algorithm is *communication-oblivious*, iff the communication pattern of the algorithm is deterministic, input independent, and known a-priori. Note that this also implies that the algorithm must run for a deterministic number of rounds.

Defining correctness of MPC algorithms

We will define a notion of δ-correctness for an MPC algorithm. Let $(y_1, y_2, \ldots, y_m) \leftarrow \mathcal{F}(x_1, x_2, \ldots, x_m)$ be a (possibly randomized) ideal functionality which, upon receiving inputs $x_1, x_2, \ldots, x_m$, outputs $y_1, y_2, \ldots, y_m$. Here x_i represents machine i's input and y_i represents machine i's output for $i \in [m]$; without loss of generality we may assume that each x_i contains exactly s words (if not we can always pad it with dummies to exactly s words). We say that an MPC algorithm denoted Π δ-correctly realizes the ideal functionality $\mathcal{F}$ iff for any input $I = (x_1, \ldots, x_m)$ the statistical distance between $\Pi(I)$ and $\mathcal{F}(I)$ is at most δ.

▶ **Definition 6** (δ-oblivious realization of an ideal functionality)**.** *We say that an MPC algorithm Π δ-obliviously realizes an ideal functionality $\mathcal{F}$, iff Π is communication-oblivious and moreover Π δ-correctly realizes $\mathcal{F}$.*

▶ Remark 7. One can alternatively consider a weaker notion for an MPC algorithm Π to "δ-oblivious realize" the ideal functionality $\mathcal{F}$, that is, we require that there exists a simulator Sim, such that for any input I of appropriate length, the following distributions must have statistical distance at most δ:

- $\mathbf{Real}^{\Pi}(1^m, 1^s, I)$: outputs the outcome of the MPC algorithm Π on input I and its communication patterns;
- $\mathbf{Ideal}^{\mathcal{F}}(1^m, 1^s, I)$: outputs $\mathcal{F}(I)$ and $\mathsf{Sim}(1^m, 1^s)$. Notice that the simulator Sim is not given the input I.

It is not hard to see that our notion (i.e., Definition 6) implies this weaker notion. This weakened definition would permit the communication patterns to be randomized and also not necessarily known a-priori. In this paper we focus on the stronger notion since we can achieve even the stronger notion in an efficiency-preserving manner.

4 Oblivious Routing and Communication-Oblivious Compiler

4.1 Problem Definition

Recall that our first goal is to obtain an MPC protocol communication oblivious, and the crux of the problem is to realize an oblivious form of routing. Imagine that in the original MPC protocol, in some round, every machine has a list of at most s words that they want to send, and each outgoing word has an intended destination machine. It is also guaranteed that every machine will receive no more than s words. The question is how to route these messages obliviously such that the communication patterns leak no information.

More formally, the $\mathsf{Routing}$ problem has the following syntax:

- **Input.** Every machine $i \in [m]$ has *at most s send instructions* where each send instruction consists of a word to be sent, and the index of the recipient machine. All machines' send instructions must jointly satisfy the s-receiver-constraint, i.e., every machine receives no more than s words.

- **Output.** Every machine outputs the list of words it is supposed to receive as uniquely defined by the joint input (i.e., all machines' send instructions); and moreover, these received words are sorted based on a lexicographically increasing order. If fewer than s words are received, the machine pads the output array with dummies to a length of exactly s.

The above abstraction defines a most natural ideal functionality $\mathcal{F}_{\text{Routing}}$ which implements the routing task correctly.

In the remainder of this section, we will devise an oblivious MPC protocol that accomplishes routing. In the process we will need two intermediate building blocks called "Bucket Route" and "Random Bucket Assignment" respectively.

Notational convention: a global Overflow indicator

Throughout this section, we shall assume that each machine maintains a global variable denoted Overflow. Initially the Overflow bit is set to 0. During the algorithm's execution, a machine may set the Overflow bit to 1. If at the end of the algorithm, all machines' Overflow indicators remain unset, we say that the algorithm is successful; otherwise the algorithm is said to have failed.

4.2 Building Block: Bucket Route

4.2.1 Syntax

The goal is to classify elements into buckets based on each element's label indicating its desired destination bucket. The algorithm is not always guaranteed to succeed; however, should it succeed, the final assignment should be correct. Recall that an algorithm is said to be successful if no machine has set their Overflow indicator by the end of the algorithm. We consider both the input and output configuration as a list of buckets spread evenly across the machines, where each bucket contains either real or dummy elements.

More formally, a $\mathsf{BucketRoute}^Z$ algorithm, parametrized by a bucket size Z, satisfies the following syntax – recall that there are in total m machines each of which has local space $O(s)$; without loss of generality, we may assume that m and s are both powers of 2:

- **Input.** In the beginning, each machine stores 2^r number of buckets each of capacity Z where $2^r \cdot Z = 2s$. Each bucket contains Z elements, some of which are real and others are dummy. Each real element carries an ℓ-bit label where $2^\ell = m \cdot 2^r$ – henceforth if a real element's label is $k \in \{0,1\}^\ell$, we say that the element wants to go to the k-th bucket out of a total of $2^\ell = m \cdot 2^r$ buckets.
- **Output.** The machines jointly output a list of $2^\ell = m \cdot 2^r$ buckets, each of capacity Z. We require that if the algorithm is successful (i.e., no machine's Overflow indicator is set), then every bucket must contain all the real elements wanting to go there plus an appropriate number of dummy elements.

4.2.2 Protocol

We now describe a $\mathsf{BucketRoute}^Z$ algorithm that tries to move every real element to the desired destination bucket but will possibly fail in the middle due to overflow events. Later in Section 4.4, we will show that if the input configuration satisfies certain nice conditions, then the overflow probability can be made negligibly small. Roughly speaking, by "nice conditions", we want that all input buckets are at most half-full; and moreover the input elements' labels are randomly chosen. In this section, we first focus on presenting the algorithm and we will worry about the probability analysis in Section 4.4.

Algorithm 1 Bucket Route.

Input: Let $2^r \cdot Z = 2s$ and let $\ell = \log_2 m + r$. The input consists of 2^ℓ buckets denoted $\mathcal{L} := \{\mathsf{B}_i : i \in [2^\ell]\}$ each with capacity Z, where each real element in a bucket contains an ℓ-bit label. There are $m = 2^{\ell - r}$ parallel machines, each of which has enough memory to store and process 2^r buckets.

1: **procedure** $\mathsf{BucketRoute}^Z(\mathcal{L} := \{\mathsf{B}_i : i \in [2^\ell]\})$ ▷ The input is a list of 2^ℓ buckets.
2: Each bucket in $\mathcal{L}$ receives an empty bit string as its label.
3: **for** $\lfloor \frac{\ell}{r} \rfloor$ sequential iterations **do**
4: Partition the buckets in $\mathcal{L}$ into m groups of equal size 2^r, where the buckets in each group has the same label; a machine is assigned to each group.

▷ *Step 4 requires the machines to exchange their buckets according to a predetermined fashion.*

5: **for** each of m machines in parallel **do**
6: Every machine calls 2^r-way $\mathsf{LocalClassify}^{r,Z}$ on its group of buckets to produce its modified group of buckets (whose labels have lengths increased by r).
7: Update the list $\mathcal{L}$ to be the union of all machines' modified groups of buckets.
8: **if** $t := \ell \mod r \neq 0$ **then**
9: In the last iteration, each machine receives 2^{r-t} sub-groups of buckets, where each sub-group contains 2^t buckets with the same label.
10: **for** each of m machines in parallel **do**
11: Every machine calls 2^t-way $\mathsf{LocalClassify}^{t,Z}$ on every of its sub-groups of buckets.
12: Let $\mathcal{L}$ be the union of all machines' updated buckets.
13: **return** the list $\mathcal{L}$ of 2^r buckets, each of which receives a unique label in $\{0,1\}^r$. Recall that if any instance of multi-way $\mathsf{LocalClassify}$ encounters $\mathsf{Overflow}$, the failure-indicator $\mathsf{Overflow}$ will be set to 1 but the algorithm will continue after truncating the excessive elements in the overflowing bucket(s).

At a very high level, our $\mathsf{BucketRoute}^Z$ algorithm will proceed in iterations such that at the end of the i-th iteration, elements will be sorted based on the first $i \cdot r$ bits of their respective labels. During each iteration i, every machine will obtain a list of buckets containing elements whose labels all share the same $((i-1) \cdot r)$-bit prefix, and the machine will locally further classify these elements into buckets based on the next r bits of their respective labels. This subroutine is henceforth called $\mathsf{LocalClassify}$ which is described more formally below. We will then describe the full $\mathsf{BucketRoute}^Z$ algorithm after specifying this subroutine.

A multi-way LocalClassify subroutine

We introduce a subroutine called a 2^t-way $\mathsf{LocalClassify}^{t,Z}$. The 2^t-way $\mathsf{LocalClassify}^{t,Z}$ subroutine is always performed by a single machine locally which operates on a list of 2^t buckets:

- Imagine that a machine has as input a list of 2^t buckets each of capacity Z where $2^t \cdot Z \leq 2s$. All of the 2^t buckets must have the same bucket-label $x \in \{0,1\}^{|x|}$, where $|x|$ is the bit-length of x. Moreover, every real element contained in the buckets has an ℓ-bit label of the form $x||y$, i.e., the element's label must be prefixed by x, followed by a suffix denoted y (that has length at least r). It is guaranteed that $|x| + r \leq \ell$.
- Now, the machine locally rearranges the input to form 2^t output buckets as follows: for each $i \in \{0,1\}^t$, the i-th output bucket has the bucket-label $x||i$, and contains all the real elements from the input whose label is prefixed with $x||i$, padded with dummies to a capacity of Z. Moreover, in each output bucket the real elements must appear before the dummies.
  If the above rearrange fails due to the overflow of some output bucket, then set the indicator Overflow to 1; moreover, truncate the bucket to a load of Z and continue with the algorithm. Note that if Overflow is set, then some elements can be lost due to the truncation.

The BucketRoute algorithm

We describe the $\mathsf{BucketRoute}^Z$ algorithm in Algorithm 1 which calls LocalClassify as a subroutine. Assuming that no Overflow is encountered, then the algorithm proceeds in the following way: first, using the 2^r-way LocalClassify subroutine, all machines rearrange their local 2^r buckets of elements based on the first r bits of the elements' labels, such that all elements whose labels have the same r-bit prefix fall into the same bucket, and this common r-bit prefix also become the bucket's label. At the end of this iteration, for each r-bit bucket label, there will be $2^{\ell-r}$ buckets with the same bucket label. We will then assign $2^{\ell-2r}$ machines to work on buckets sharing each r-bit bucket label, and each machine now locally calls LocalClassify to further classify the elements based on the next r-bits of their input labels; and this goes on. In general, after the end of the i-th iteration, buckets will acquire labels of length $i \cdot r$ bits, and there will be $2^{\ell-i\cdot r}$ buckets sharing each unique $(i \cdot r)$-bit label; we now assign all buckets with the same label to $2^{\ell-(i+1)\cdot r}$ machines which further classifies the elements based on the next r bits in the input labels. The algorithm will have ended by iteration i if $i \cdot r \geq \ell$, i.e., after $O(\ell/r)$ iterations – at the end, all buckets across all machines will have a unique ℓ-bit label. Algorithm 1 formalizes the above idea and in particular, treats the last iteration more formally for the case when ℓ is not divisible by r.

▶ **Fact 8.** *Using the parameters in Algorithm 1, the number of rounds is* $O(\frac{\ell}{r}) = O(\frac{\log m}{r})$.

▶ **Fact 9** (Communication pattern)**.** *The communication pattern of Algorithm 1 is deterministic, and depends only on the parameters m, ℓ, r and Z. Moreover, in every round, the total communication is upper bounded by $O(m \cdot s)$.*

Proof. Step 4 of Algorithm 1 is where the communication happens, and it is not hard to check that this fact holds. ◀

4.3 Building Block: Oblivious Random Bucket Assignment

Based on the $\mathsf{BucketRoute}^Z$ primitive described above, we realize another intermediate building block called "random bucket assignment". The problem, henceforth denoted $\mathsf{RandBucket}^Z$, is parametrized by an appropriate bucket size Z and is described below:

- **Input.** The input is an array of $m \cdot s$ elements spread evenly across the machines where each machine receives exactly s elements; each element can either be real or dummy. We assume that each element can be stored in $O(1)$ words.

- **Output.** Each machine receives $2^r = 2s/Z$ output buckets each of capacity Z. The contents of the buckets are determined below:
  - Every real element in the input array is assigned to a random bucket out of the $2^r \cdot m$ buckets;
  - If a bucket receives more than Z real elements, choose the Z lexicographically smallest elements to populate the bucket (and the remaining elements are lost);
  - Else if a bucket receives Z or fewer real elements, then the bucket should contain all of these elements, in a lexicographically increasing order, and padded at the end with an appropriate number of dummies to a capacity of Z.

Note that the above abstraction also defines the most natural ideal functionality $\mathcal{F}^Z_{\text{RandBucket}}$ which correctly implements the above task.

Protocol

Using $\mathsf{BucketRoute}^Z$, we can obliviously realize random bucket assignment using the following simple algorithm where we assume that m and s are powers of 2 without loss of generality (if not, we can pad them to the nearest power of 2):

1. Choose r, ℓ based on Z, m, and s, such that $2^r = 2s/Z$ and $2^\ell = m \cdot 2^r$. Henceforth 2^r denotes the number of buckets on each machine, and 2^ℓ denotes the total number of buckets across all machines.
2. Each machine places $Z/2$ elements from the input array to each of its 2^r buckets, and pads each bucket with an additional $Z/2$ dummies to a capacity of Z – note that each bucket is at most half full.
3. Every machine assigns a random ℓ-bit label to each real element in its buckets, indicating which bucket the element wants to go to.
4. Call $\mathsf{BucketRoute}^Z$ (Algorithm 1) to produce the list of output buckets, and for each resulting bucket, sort the elements in it based on a lexicographically increasing order, placing all dummies at the end.

Recall that in the $\mathsf{BucketRoute}^Z$, a machine may encounter a bucket overflow exception which will cause the $\mathsf{Overflow}$ indicator to be set. Below we bound the probability of seeing $\mathsf{Overflow}$ using the fact that the initial labels are randomly chosen and that the initial buckets are at most half full – this allows us to prove good load-balancing properties. We use the standard Chernoff Bound to analyze overflow probability.

▶ **Fact 10** (Chernoff Bound). *Suppose X is a sum of independent $\{0,1\}$-random variables. Then, for any $\beta \geq 2$, $\Pr[X \geq \beta E[X]] \leq \exp(-\frac{\beta E[X]}{6})$.*

▶ **Lemma 11** (Overflow probability for random bucket assignment algorithm). *In the above algorithm, the probability that some machine sets the $\mathsf{Overflow}$ indicator is at most $O(\frac{\ell}{r}) \cdot 2^\ell \cdot e^{-\frac{Z}{6}}$.*

Proof. Observe that there are $O(\frac{\ell}{r})$ iterations in Algorithm 1. We fix some iteration and some bucket B, and analyze the overflow event of that bucket after that iteration.

Observe that if we assume that all buckets from previous iterations have infinite capacity, then no element will be lost in previous iterations. Hence, the resulting number of elements ending up at bucket B after this iteration in this alternate world will stochastically dominate that in the actual world. The property of no overflow in previous iterations is used to ensure that different elements do not influence one another so that we can apply Chernoff Bound. By stochastic dominance, it suffices to analyze the overflow probability of bucket B after this iteration under this additional assumption.

Suppose after this iteration, this bucket has received a label with k bits. This means that the elements in this bucket could only have come from 2^k possible input buckets, each of which contains at most $\frac{Z}{2}$ elements.

Observe that each such element receives a random ℓ-bit label, whose k-bit prefix coincides with this bucket's label with probability $\frac{1}{2^k}$. Hence, the expected number of elements entering this bucket after this iteration is at most $\frac{Z}{2}$.

Therefore, using Chernoff Bound (Fact 10), after each iteration, each bucket overflows with probability at most $e^{-\frac{Z}{6}}$.

Finally, the union bound over all iterations and all buckets gives the result. ◀

▶ **Lemma 12** (Correctness of random bucket assignment)**.** *The above algorithm, parametrized with Z, δ-correctly realizes $\mathcal{F}^Z_{\text{RandBucket}}$ where $\delta = O(\frac{\ell}{r}) \cdot 2^\ell \cdot e^{-\frac{Z}{6}}$ (note that the parameters ℓ and r are determined once Z is determined).*

Proof. Follows from Lemma 11 and the fact for every random string ρ assigning real elements to destination buckets, if ρ does not cause Overflow in the algorithm, then the algorithm's output must correctly match that output by the ideal functionality $\mathcal{F}^Z_{\text{RandBucket}}$ consuming the same randomness ρ. ◀

▶ **Fact 13.** *The communication patterns of the algorithm is deterministic and depends only on the parameters m, ℓ, r and Z.*

Proof. The communication pattern of the algorithm stems from that of $\mathsf{BucketRoute}^Z$ since all other steps are performed locally on each machine and incur no communication. The fact now follows directly from Fact 9. ◀

4.4 Putting it Together: Oblivious Routing

Using the building blocks $\mathsf{BucketRoute}^Z$ and $\mathsf{RandBucket}^Z$, we can realize oblivious routing as follows:

(a) Without loss of generality, we may assume that both m and s are powers of 2 and s is a perfect square. Choose the parameter $Z = 2\sqrt{s}$ and $2^r = 2s/Z$, and $\ell := \log_2 m + r$.

(b) Every machine does the following: let X be an input array containing all the words the machine wants to send (also called outgoing words), padded with dummies to a length of s. Each real outgoing word in the input array has a label of the form $x||\rho$ where $x \in \{0,1\}^{\log_2 m}$ encodes the identifier of the recipient machine, and $\rho \xleftarrow{\$} \{0,1\}^{\ell - \log_2 m}$ is chosen at random and indicates the index of the destination bucket within machine x.

(c) Invoke an oblivious $\mathsf{RandBucket}^Z$ algorithm to assign each outgoing word to a random bucket among a total of 2^ℓ buckets, and recall that each machine stores 2^r of these buckets.

We emphasize that the labels selected earlier in Step (b) are not used as destination labels for the $\mathsf{RandBucket}^Z$ algorithm since the $\mathsf{RandBucket}^Z$ algorithm itself internally assigns a random bucket to each element. However, these labels selected earlier will be used in the next step.

(d) Invoke an instance of $\mathsf{BucketRoute}^Z$ to route each outgoing word to the destination bucket encoded in the label chosen in Step (b). Now, every machine looks at the resulting 2^r buckets it stores and outputs all received (real) words in a lexicographically increasing order, padded with an appropriate number of dummies to a length of s.

▶ **Fact 14.** *In the above routing algorithm, the communication pattern is deterministic and depends only on the parameters Z, m, r, ℓ.*

Proof. The communication pattern of the above algorithm is determined by the communication pattern of the $\mathsf{RandBucket}^Z$ algorithm and the $\mathsf{BucketRoute}^Z$ algorithm. The fact now follows directly from Fact 9 and Fact 13. ◀

▶ **Lemma 15** (Correctness of routing). *Suppose that $\mathsf{RandBucket}^Z$ δ-correctly realizes $\mathcal{F}^Z_{\text{randbucket}}$. Then, the above algorithm δ'-correctly realizes $\mathcal{F}_{\text{Routing}}$ for some $\delta' \leq \delta + O(\frac{\ell}{r}) \cdot 2^\ell \cdot e^{-\frac{Z}{6}}$.*

Proof. We consider a sequence of hybrids.

$\mathbf{Real}^Z(I)$. We will use $\mathbf{Real}^Z(I)$ to denote the outcome of the real-world algorithm.

$\mathbf{Hyb}_1^Z(I)$. We now consider a hybrid execution denoted $\mathbf{Hyb}_1^Z$, which is defined almost the same as the real-world execution, except that we now replace the $\mathsf{RandBucket}^Z$ algorithm with $\mathcal{F}^Z_{\text{randbucket}}$. We define $\mathbf{Hyb}_1^Z(I)$ to output the outcome of this hybrid execution upon the input I.

Since $\mathsf{RandBucket}^Z$ δ-correctly realizes $\mathcal{F}^Z_{\text{randbucket}}$, it holds that for any I, $\mathbf{Hyb}_1^Z(I)$ has at most δ statistical distance from $\mathsf{Real}^Z(I)$.

▶ **Lemma 16.** *For any I, in the execution defined by $\boldsymbol{Hyb}_1^Z(I)$, the probability that some machine sets the $\mathsf{Overflow}$ indicator is bounded by $O(\frac{\ell}{r}) \cdot 2^\ell \cdot e^{-\frac{Z}{6}}$.*

Proof. To prove this, we instead consider an execution $\mathbf{Hyb}_1^\infty$ where the parameters Z, ℓ, and r are chosen as before; however, we do not impose a Z-capacity limit on any of the buckets, including the buckets in $\mathcal{F}_{\text{randbucket}}$ or any bucket in the hybrid execution $\mathbf{Hyb}_1^Z$. It is not hard to that the probability of seeing $\mathsf{Overflow}$ in $\mathbf{Hyb}_1^Z$ is upper bounded by the probability that there is some bucket storing more than Z real elements in $\mathbf{Hyb}_1^\infty$. This can be formally proven through a standard stochastic domination argument.

Therefore, it suffices for us to prove that in $\mathbf{Hyb}_1^\infty$, the probability that some bucket needs to store more than Z real elements is upper bounded by $O(\frac{\ell}{r}) \cdot 2^\ell \cdot e^{-\frac{Z}{6}}$, and below we prove this statement.

The analysis is similar to the proof of Lemma 11. We fix some iteration in Algorithm 1 and some bucket B. Let $X_{i,B}$ denote an indicator random variable indicating the contribution of the i-th outgoing word in the input to the load of the bucket B. Our goal is to prove a tail bound for $\sum_i X_{i,B}$ which denotes bucket B's total load. Note that $X_{i,B}$ depends only on the i-th outgoing word's destination bucket and the initial bucket placement chosen by $\mathcal{F}_{\text{RandBucket}}$ for the i-th outgoing word. Therefore, all of the random variables $\{X_{i,B}\}_i$ are independent which will allow us to apply the standard Chernoff bound. To do so, we will first show that the expected number of real elements the bucket contains after this iteration is at most $\frac{Z}{2}$. Hence, in the execution $\mathbf{Hyb}_1^\infty$, we analyze the expected number of elements for this bucket B after this iteration based on the number k of bits of the label received by this bucket after this iteration. Let $\mathcal{L}$ be the list of buckets obtained after $\mathcal{F}_{\text{randbucket}}$.

1. Suppose $k \leq \log_2 m$. The k-bit label indicate a subset of 2^{m-k} destination machines. Then, only elements going to these 2^{m-k} machines can end up in this bucket. There are totally at most $2^{m-k} \cdot s$ such elements.
   Moreover, only elements from 2^k buckets in $\mathcal{L}$ can end up in bucket B after this iteration. Due to $\mathcal{F}_{\text{randbucket}}$, an element is in one of those 2^k buckets independently with probability $\frac{2^k}{2^\ell}$.
   Hence, the expected number of elements in bucket B after this iteration is at most $2^{m-k} \cdot s \cdot \frac{2^k}{2^\ell} = \frac{s}{2^r} = \frac{Z}{2}$.

2. Suppose $k > \log_2 m$. The k-bit label then identifies $2^{\ell-k}$ buckets within a certain destination machine M.
   Recall that at most s elements are supposed to go to machine M. In order for an element to end up in bucket B after this iteration, it has to satisfy the following independent events:
   a. The element is in one of 2^k buckets in $\mathcal{L}$ whose elements can end up in B after this iteration. Due to $\mathcal{F}_{\text{randbucket}}$, this happens with probability $\frac{2^k}{2^\ell}$.
   b. The element has chosen a destination bucket that is among those $2^{\ell-k}$ buckets whose identity agrees with the k-bit label of the bucket B. This happens with probability $\frac{2^{\ell-k}}{2^r}$.
   Hence, the expected of elements that enter this bucket B after this iteration is at most $s \cdot \frac{2^k}{2^\ell} \cdot \frac{2^{\ell-k}}{2^r} = \frac{s}{2^r} = \frac{Z}{2}$, as required.

Finally, we apply a standard Chernoff bound and then a union bound over all iterations and all buckets just like in Lemma 11, which leads to the lemma statement. ◀

Ideal(I). The experiment $\mathbf{Ideal}(I)$ outputs the result of $\mathcal{F}_{\text{Routing}}$ upon the input I

We now show that for any I, $\mathbf{Ideal}(I)$ and $\mathbf{Hyb}_1^Z$ have statistical distance at most $O(\frac{\ell}{r}) \cdot 2^\ell \cdot e^{-\frac{Z}{6}}$. Note that conditioned on 1) seeing no Overflow in $\mathbf{Hyb}_1^Z$ and 2) that $\mathcal{F}_{\text{randbucket}}$ does not pick bad randomness such that some bucket initially exceeds Z load, then the outcome obtained in $\mathbf{Hyb}_1^Z$ would be the same as the output of $\mathbf{Ideal}(I)$. In Lemma 16, we have shown that the probability of some machine setting the Overflow indicator in $\mathbf{Hyb}_1^Z$ $O(\frac{\ell}{r}) \cdot 2^\ell \cdot e^{-\frac{Z}{6}}$, therefore it suffices to show that the probability that $\mathcal{F}_{\text{randbucket}}$ internally exceeds Z load is upper bounded by $O(\frac{\ell}{r}) \cdot 2^\ell \cdot e^{-\frac{Z}{6}}$. Due to a simple application of the Chernoff bound, we can show that the probability that some fixed bucket exceeds load Z is at most $e^{-\frac{Z}{6}}$. Now, applying a union bound over all 2^ℓ buckets, the conclusion follows.

In summary, applying a standard hybrid argument, we can complete the proof of this lemma. ◀

▶ **Corollary 17.** *The above algorithm δ-obliviously realizes $\mathcal{F}_{\text{Routing}}$ by Definition 6 for $\delta = \exp(-\Omega(\sqrt{s}))$.*

Proof. Straightforward by Fact 14, Lemma 15 and Lemma 12 and the parameters chosen in the algorithm. ◀

4.5 Oblivious Sorting

Given our oblivious routing primitive, one immediate and interesting application is to construct a constant-round MPC algorithm that obliviously realizes sorting.

To achieve this, recall that Goodrich [67] constructed a non-oblivious MPC algorithm that accomplishes sorting in $O(1)$ rounds; and moreover, his algorithm satisfies both the s-receiver-constraint and the s-sender-constraint.

▶ **Lemma 18** (Constant-round sorting, Theorem 3.5 of Goodrich [67]). *Let $m = N^{1-\varepsilon}$ and $s = O(N^\varepsilon)$ for any constant $\varepsilon \in (0, 1)$. Suppose that each item to be sorted can be stored in $O(1)$ words. There exists a deterministic, comparison-based sorting algorithm that can correctly sort N items stored on m machines each with s local space consuming an a-priori fixed constant number of rounds; and moreover, the algorithm satisfies both the s-sender-constraint and the s-receiver-constraint.*

The idea is to apply our Routing algorithm to realize each round of communication in Goodrich's algorithm. In this way, we obtain the following theorem:

▶ **Theorem 19** (Constant-round oblivious sorting on MPC). *Let $m = N^{1-\varepsilon}$ and $s = O(N^{\varepsilon})$ for any constant $\varepsilon \in (0, 1)$, and suppose that each items to be sorted can be represented in $O(1)$ words. There exists an MPC algorithm that obliviously sorts N items stored on s machines in $O(1)$ number of rounds and the result is correct with $1 - \exp(-\Omega(\sqrt{s}))$ probability.*

Proof. Note that Goodrich's algorithm runs for an apriori-fixed constant number of rounds. Thus, if we apply our oblivious Routing algorithm to realize each round of communication in Goodrich's algorithm, the resulting algorithm has a deterministic communication pattern; and moreover, due to Corollary 17, the resulting algorithm correctly sort the input with $1 - \exp(-\Omega(\sqrt{s}))$ probability. ◀

Although our secure multi-party computation compiler later will not directly rely on oblivious sorting as a building block, we state this result explicitly nonetheless since sorting is such an important and fundamental algorithmic building block.

4.6 Communication-Oblivious Compiler

We can now arrive at Theorem 1 which we restate below for the reader's convenience.

▶ **Theorem 20** (Communication-oblivious MPC algorithms: Restatement of Theorem 1). *Suppose that $s = N^{\varepsilon}$ and that m is upper bounded by a fixed polynomial in N. Given any MPC algorithm Π that completes in R rounds where each of the m machines has s local space, there is a communication-oblivious MPC algorithm $\widetilde{\Pi}$ that computes the same function as Π except with $\exp(-\Omega(\sqrt{s}))$ probability, and moreover $\widetilde{\Pi}$ completes in $O(R)$ rounds, and consuming $O(s)$ space on each of the m machines. Furthermore, only $O(m \cdot s)$ amount of data are communicated in each round in an execution of $\widetilde{\Pi}$.*

Proof. We apply the oblivious Routing algorithm developed earlier in this section to realize every communication round of the original MPC protocol Π. The theorem now follows directly from Corollary 17. ◀

5 Secure Multi-Party Computation for Massively Parallel Computing

In this section, we consider how to perform secure computation on a Massively Parallel Computing (MPC) architecture. As before, we consider a set of m machines each of which is s-space-bounded. Without loss of generality, we consider a setting where each machine receives some input denoted $x_1, x_2, \ldots, x_m$ respectively where each input contains at most s words. Now, these machines would like to jointly evaluate a function $(y_1, y_2, \ldots, y_m) \leftarrow f(x_1, x_2, \ldots, x_m)$ such that at the end, the i-th machine obtains the s-bounded output y_i for $i \in [m]$. We would like to ensure that an adversary controlling a relatively small subset of the machines will not learn anything beyond the outputs of the machines in its control.

The question we are concerned about is the following: suppose that there is an efficient, insecure MPC protocol Π to evaluate the function f. Can we now securely evaluate the function f while preserving the efficiency relative to the insecure protocol Π?

5.1 Execution Model: SMPC for MPC

We consider an MPC protocol Π executing on m machines each with maximum space s. To define Secure Multi-Party Computation (SMPC), we augment the protocol execution model used so-far in the paper as follows to capture a polynomial-time adversary who can corrupt a

subset of the machines. Most of the definitions below directly follow the standard approach in the cryptography literature – there is, however, one subtlety that needs to be clarified when we marry SMPC and MPC: since corrupt machines can send arbitrary messages to honest machines, we must now redefine the s-receiver-constraint (see details in the "communication model" paragraph).

- We now parametrize the protocol's execution with a security parameter denoted κ. Therefore we may write Π as $\Pi(1^\kappa, 1^m, 1^s)$, i.e., it is parametrized by the security parameter κ and the MPC framework's parameters 1^m and 1^s.
- A subset of the machines which are said to be *corrupt* are controlled by an adversary $\mathcal{A}(1^\kappa)$. We assume that corruption is static, i.e., $\mathcal{A}$ decides which machines to corrupt before the protocol execution starts. All protocol messages received by corrupt machines are visible to $\mathcal{A}$, and $\mathcal{A}$ fully controls what messages corrupt machines send.

  Machines that are not in $\mathcal{A}$'s control are said to be honest, and honest machines faithfully follow the prescribed protocol.
- All machines' inputs are chosen by some environment denoted $\mathcal{Z}(1^\kappa)$; and at the end of the protocol, all honest machines send their respective output to $\mathcal{Z}$.
- During the execution $\mathcal{A}$ and $\mathcal{Z}$ may communicate freely.

Communication model

The protocol proceeds in rounds and machines communicate with each other through a pairwise point-to-point network. At the beginning of each round, honest machines receive messages from the network; and afterwards they perform computation and send messages. If an honest machine sends a message in round r, then an honest recipient will receive the message at the beginning of the next round. We assume that honest machines communicate through a pairwise *secure channel* such that the adversary can observe who is communicating with whom, the length of each honest message sent, but not the contents of the message – note that since we can realize secure channels from authenticated channels with key exchange and authenticated encryption, assuming secure channels is without loss of generality.

Recall that in this new setting, some machines can be corrupt and corrupt machines send arbitrary, unwanted messages to honest machines. We cannot guarantee that the s-receiver-constraint is respected in the presence of corrupt nodes; instead, we require the following:

> at the end of the previous round, an honest machine must have written down in a designated location in its memory a *receiving schedule* for the next round, i.e., a list of $\{(f_j, w_j)\}_j$ pairs where $f_j \in [m]$ denotes the index of a sender and $w_j \in [s]$ denotes the number of words the machine is expecting to receiver from machine f_j. Additionally, it must hold[7] that $\sum_j w_j \leq s$.

Every sender whose index appears in this receiving schedule is called an anticipated sender. Now, a machine will save only the first w_j words received from each anticipated sender f_j; it will ignore all words received from unanticipated senders; and also ignore all excessive words received from anticipated senders.

[7] Later on, in our compiled protocol, s will actually be substituted with $O(s) \cdot \mathsf{poly}(\kappa)$ where s is the per-machine space complexity of the original MPC to be compiled.

5.2 Security Definition

Security is defined through (computational) observational equivalence of the environment $\mathcal{Z}$ in an ideal-world and a real-world execution. In the real-world execution, machines run the real protocol Π. In the ideal-world execution, the computation task is performed by an ideal functionality $\mathcal{F}^f$ which computes the intended function f and distributes the results to everyone.

Formally, we define a real-world and an ideal-world execution formally as below:

- $\mathbf{Real}^{\mathcal{A},\mathcal{Z},\Pi}(1^\kappa, 1^m, 1^s)$: $\mathcal{Z}(1^\kappa)$ chooses and provides inputs $x_1, x_2, \ldots, x_m$ for each of the m machines. Now the m machines engage in a protocol execution as explained above, where honest machines will faithfully execute the prescribed protocol Π using the input they obtained from $\mathcal{Z}$ but corrupt machines that are controlled by $\mathcal{A}$ can behave arbitrarily. At the end of the protocol, the honest machines send their output to $\mathcal{Z}$.
- $\mathbf{Ideal}^{\mathcal{S},\mathcal{Z},\mathcal{F}^f}(1^\kappa, 1^m, 1^s)$: The ideal-world execution involves the environment $\mathcal{Z}$ and an ideal-world adversary denoted $\mathcal{S}$. $\mathcal{Z}$ and $\mathcal{S}$ can communicate arbitrarily during the ideal-world execution. Now the execution is defined as below where $\mathsf{Honest} \subseteq [m]$ denotes the set of honest machines; and $\mathsf{Crupt} := [m] \backslash \mathsf{Honest}$ denotes the set of corrupt machines all of which are controlled $\mathcal{S}$:
  1. First, $\mathcal{Z}$ chooses and provides inputs $x_1, x_2, \ldots, x_m$ for each of the m machines.
  2. Every honest machine $i \in \mathsf{Honest}$ sends the input x_i received from $\mathcal{Z}$ to $\mathcal{F}^f$, and $\mathcal{F}^f$ records $\widetilde{x}_i := x_i$;
  3. For a corrupt machine $j \in \mathsf{Crupt}$, $\mathcal{F}^f$ may receive an input x'_j from j, and if so, it records $\widetilde{x}_j := x'_j$. Note that corrupt machines may use arbitrary inputs, and not necessarily the ones provided by $\mathcal{Z}$.
  4. As soon as all m machines have provided input, $\mathcal{F}^f$ computes the outputs $(y_1, y_2, \ldots, y_m) := f(\widetilde{x}_1, \widetilde{x}_2, \ldots, \widetilde{x}_m)$ and gives $\{y_i\}_{i \in \mathsf{Crupt}}$ to $\mathcal{S}$.
  5. Upon receiving `deliver` from the ideal-world adversary denoted $\mathcal{S}$, if any corrupt machine j has not yet provided input, set $\widetilde{x}_j := \perp$ and compute $(y_1, y_2, \ldots, y_m) := f(\widetilde{x}_1, \widetilde{x}_2, \ldots, \widetilde{x}_m)$. Now for $i \in [m]$, give y_i to machine i.
  6. Upon receiving an output from $\mathcal{F}^f$, an honest machine forwards the output to $\mathcal{Z}$.

In this paper, we define a notion of compositional security for multi-party computation analogous to the guarantees of Universal Composability [30]; moreover, our definition captures the requirement of *guaranteed output*, i.e., a small corrupted coalition should not be able to stop honest machines from producing output. The formal definition is provided below – note that the requirement of guaranteed output is captured since we require that the ideal-world adversary $\mathcal{S}$ send `deliver` to the ideal functionality:

▶ **Definition 21** (SMPC for MPC). *We say that an MPC protocol Π securely realizes some ideal functionality $\mathcal{F}^f$ against a t-bounded adversary where $t < m$ iff for any non-uniform polynomial-time adversary $\mathcal{A}$ that statically corrupts at most t out of m machines, there is a non-uniform polynomial-time ideal-world adversary $\mathcal{S}$ which is required to send* `deliver` *to $\mathcal{F}^f$, such that for any non-uniform polynomial-time environment $\mathcal{Z}$,*

$$\left\{ \mathsf{View}^{\mathcal{Z}}(\mathbf{Real}^{\mathcal{A},\mathcal{Z},\Pi}(1^\kappa, 1^m, 1^s)) \right\}_\kappa \approx \left\{ \mathsf{View}^{\mathcal{Z}}(\mathbf{Ideal}^{\mathcal{S},\mathcal{Z},\mathcal{F}^f}(1^\kappa, 1^m, 1^s)) \right\}_\kappa$$

where $\mathsf{View}^{\mathcal{Z}}(\mathbf{Expt})$ denotes the view of $\mathcal{Z}$ in the experiment $\mathbf{Expt}$ and $\approx$ denotes computational indistinguishability of two probability ensembles.

Resilience assumption

Henceforth we will assume that the adversary controls no more than $\frac{1}{3} - \eta$ fraction of the machines for an arbitrarily small constant η. Such a corruption threshold is (almost) the best one can hope for in a pairwise point-to-point network, since it takes at least 2/3 honest to realize broadcast [48, 82] (note that multi-party computation with guaranteed output implies broadcast).

5.3 Building Block: Constant-Round, Weakly Space-Efficient SMPC

We will leverage a universally composable SMPC protocol as a building block, and we would like this protocol to be not only constant round, but also somewhat efficient in space. Specifically, if the function $f(x_1, \ldots, x_{m'})$ evaluated requires $S \geq |x_1| + |x_2| + \ldots |x_{m'}|$ space to compute insecurely on a RAM, then for m' machines to securely compute the function f would require each machine to expend about $O(S)$ space, and moreover, allowing a $\mathsf{poly}(\kappa)$ blowup due to the use of cryptography. While this seems somewhat conserving in terms of space from the perspective of each individual machine, from the perspective of all m' machines, we are expending m' times more total space than the original RAM – for this reason, we call this notion weakly space-efficient. More formally, we require the following efficiency guarantees:

1. *Constant round*: the protocol must complete in $O(1)$ rounds;
2. *Weak space efficiency*: suppose that the function $f(x_1, x_2, \ldots, x_{m'})$ can be computed insecurely on a Random Access Machine (RAM) with S space where S must account for the space needed to write down all m' inputs and outputs, we would then like a protocol running on m' machines that securely realize $\mathcal{F}^f$, requiring only $\mathsf{poly}(\kappa) \cdot O(S)$ space on each machine.
3. *Communication efficiency*: total communication must be asymptotically not more than the total space of all machines;

Note that we cannot directly use this SMPC protocol to compile an MPC protocol to a secure counterpart if we want to preserve the efficiency of the original (insecure) MPC, since weak space efficiency still blows up each machine's space complexity to at least the size of the whole input. Looking ahead, we will run this SMPC building block within a randomly elected poly-logarithmic-sized committee to emulate a MPC machine with small space.

On the other hand, since a machine cannot receive messages of length greater than its space complexity, a constant-round, weakly space efficient SMPC protocol has communication complexity at most $\mathsf{poly}(\kappa) \cdot O(m' \cdot S)$, independent of the time complexity of the functionality f. Constant-round protocols based on garbled circuits or garbled RAM do not achieve desired efficiency. Thus, we consider FHE-based SMPC protocols, and furthermore, we would require the FHE schemes to satisfy a strong notion of *compactness* [55, 58] to avoid dependency on the circuit depth complexity of the functionality (which is the case if standard leveled FHE schemes are used). FHE schemes with this compactness property can be achieved by assuming circular security for standard FHE constructions [55,58] or using indistinguishability obfuscation [31, 53].

There is a long line of work on constant-round FHE-based SMPC that construct SMPC protocols based on threshold FHE (TFHE) or multi-key FHE (MKFHE) [14, 28, 72, 75, 91]. For our purpose, relying on these protocols would requires to assume above-mentioned compactness for TFHE or MKFHE (and trusted setup for some of them). Ideally, we would like to construct a protocol based on any compact (plain) FHE.

To achieve this goal, we follow the approach of constructing SMPC based on threshold FHE (e.g., [8]) and obtain the threshold FHE in use by applying the universal thresholdizer of Boneh et al. [23] to any compact FHE. We show that this yields a constant-round weakly space efficient SMPC protocol, but it requires a trusted setup. We further remove the setup by invoking another constant-round FHE-based SMPC protocol of Badrinarayanan et al. [14] to instantiate the setup. The protocol of [14] does not require a trusted setup. Furthermore, since the setup of threshold FHE has complexity independent of the complexity of the functionality, we do not need to assume compactness for the underlying (multi-key) FHE in the protocol of [14] to achieve weak space efficiency.

Later in our construction, we actually require that each machine receives different output in this SMPC protocol. Namely, the functionality to be securely computed is $f = (f_1, \ldots, f_{m'})$ where each machine M_i receives output $f_i(x_1, \ldots, x_{m'})$. In this case, we consider the space complexity as the maximal space complexity of $f_1, \ldots, f_{m'}$. Formally, we obtain the following theorem, which will serve as the building block of our SMPC for MPC construction. We prove the theorem in Appendix A.

▶ **Theorem 22** (Constant-round, weakly space-efficient, and communication efficient SMPC)**.** *Assume that the LWE assumption holds, the existence of enhanced trapdoor permutations, and the existence of FHE with an appropriate notion of compactness defined in Appendix A.1. Then, for any polynomial-time computable functions $f = (f_1, \ldots, f_{m'})$, there is a constant-round, weakly space-efficient, and communication efficient protocol that securely realizes $\mathcal{F}^f$ on m' machines against a t-bounded adversary as long as $m' \geq 3t + 1$.*

Proof. Deferred to Appendix A. ◀

5.4 Intuition

Given an original insecure MPC protocol that computes some function f over m machines' respective inputs, we would like to compile it to an SMPC protocol that securely realizes the functionality $\mathcal{F}^f$. We would like the compilation to be efficiency-preserving, that is, if the original MPC protocol completes in R rounds consuming s space per machine, then the compiled SMPC protocol completes in $O(R)$ rounds, and consumes $O(s) \cdot \mathsf{poly} \log \lambda \cdot \mathsf{poly}(\kappa)$ space per machine. Specifically, κ and λ denote a computational and a statistical security parameter respectively: the $\mathsf{poly}(\kappa)$ blowup is due to the use of cryptography and the $\mathsf{poly} \log \lambda$ blowup stems from random committee election.

Inspired by Boyle et al. [24, 27], our idea is to randomly elect a small, polylogarithmically sized committee to securely emulate each machine of the original MPC protocol. We use $m' = \mathsf{poly} \log \lambda$ to denote the size of each committee to distinguish from the total number of machines m. Suppose that $(1/3 - \eta)m$ machines are corrupt where η is an arbitrarily small constant, then by Chernoff bound, within each committee, only $t' \leq (1/3 - \eta/2)m'$ are corrupt except with negligible (in both λ and κ) probability.

Without loss of generality, henceforth we may assume that the original MPC protocol is communication-oblivious. If not, we can always take the compiler of Section 4.6 and compile the protocol to a communication-oblivious counterpart incurring only constant round and space blowup.

The state of each machine $i \in [m]$ in the original MPC protocol will now be secret shared among the i-th committee using a t'-out-of-m' robust secret sharing scheme. This means that at the beginning of the protocol, after each machine $i \in [m]$ receives an input it will invoke a protocol to secret share its input among the i-th committee. To accomplish this, machine i and the i-th committee will jointly perform a constant-round, weakly space-efficient,

and communication efficient SMPC (see Section 5.3) that realizes a robust secret-sharing functionality. Within each round, each machine $i \in [m]$ in the original MPC protocol must perform some local computation; in the compiled secure protocol, this computation will now be jointly performed by the i-th committee using a constant-round, weakly space-efficient, and communication efficient SMPC protocol (see Section 5.3). At end of this SMPC protocol, every committee member obtains a robust secret-share of machine i's new state in the original MPC. Finally, for a machine $i \in [m]$ to send a message to a machine $j \in [m]$ in the original MPC protocol, this communication will now also be implemented by an instance of a constant-round, weakly space-efficient SMPC protocol among the i-th and the j-th committees. At the end of the protocol, each member of the j-th committee should receive a robust secret share of the message.

5.5 Assumptions and Notations

5.5.1 Assumptions on the Original MPC

Without loss of generality, we can make the following assumptions on the original MPC to be compiled:

WLOG_1: We assume that the original MPC to be compiled has a deterministic communication pattern; and moreover, in every round every machine sends at most s words (where sending the same word to two machines is counted twice). Not only so, we may assume that in every round, every machine can compute on the fly and write down 1) an ordered list of at most s machines it wants to send words to and 2) an ordered list of at most s machines it is expecting to receive data from; and moreover this can be accomplished in $O(s)$ space.

If this is not the case, we can always apply the oblivious-compiler of Section 4.6 to make it so while incurring only constant blowup in round complexity and space.

WLOG_2: We may in fact assume that in the original MPC, at the end of the computation step in every round, every machine writes down at a designated location in memory (e.g., address 0) a list of at most s words to be sent. Recall that by WLOG_1, the destinations of these outgoing words are deterministic and a-priori known.

WLOG_3: We assume that in each round, after receiving messages from the network, a machine appends the received messages to its local memory in an arbitrary order. This is without loss of generality since during the computation step, the machine can always sort the received messages locally based on any order that is desired.

5.5.2 Notations

We will use the following notations:

- Let $\mathsf{mem}' \leftarrow M_r^i(\mathsf{mem})$ be the description of the RAM corresponding to machine i's computation in the r-th round in the original MPC; it takes in machine i's current memory mem and outputs a new memory state mem'.
- Let $m' = \mathsf{poly}\log\lambda$ denote each small committee's size, let η be an arbitrarily small constant, and let $t' = (1/3 - \eta/2)m'$ such that each committee has at most t' corrupt nodes except with negligible probability (see also the proof of Theorem 23).
- Let $(\mathsf{Share}, \mathsf{Recons})$ denote a t'-out-of-m' robust secret sharing scheme (see Appendix D). If the Share algorithm is provided with a string consisting of multiple words, we always assume that Share will perform secret sharing *word by word*; and similarly Recons will be performed word by word too.

- Later in our protocol, a small committee will be elected to emulate each machine in the original MPC protocol. Henceforth the committee that is emulating machine i in the original MPC is called the i-th committee.

5.5.3 Computing Relevant Committee Information on the Fly

To enable the pseudo-random committee election, a common reference string crs will be distributed to all honest machines at protocol start, and thus crs is common knowledge. In our protocol, committee election relies only on the crs. Specifically, the i-th committee is decided by $\mathsf{PRF}_{\mathsf{crs}}(i)$ where PRF is a pseudorandom function. At this point, it might seem safe to assume that the members of all committees are common knowledge. There is a slight subtlety here in that a machine in fact cannot store the members of all committees since this would consume too much space. Fortunately, by consuming $\mathsf{poly}\log\lambda \cdot O(s)$ additional space, a machine i can always compute on the fly and temporarily store members of committees relevant to itself in some round, including

1. which committees it is serving on, and all members of every committee it is serving on – we will show that every machine serves on at most $\Theta(m') = \mathsf{poly}\log\lambda$ committees except with negligible probability (see proof of Theorem 23);
2. all members of every committee it wants to communicate with in the present round: there are at most $2s$ such committees due to the s-sender-constraint and the s-receiver-constraint – note that to write this information down we rely on the fact that the original MPC to be compiled has a deterministic and fixed communication pattern;
3. all members of the committee emulating machine i itself.

Because of the above observations, later in Section 5.6, for simplicity it is unambiguous to parametrize our ideal functionalities that serve 1 or 2 committees with the relevant committee's indices – if a machine needs interact with some ideal functionality, it can be computed on-the-fly exactly which other machines will also be involved. Except with negligible probability, the additional per-machine space needed to compute on-the-fly and store the committee information relevant to the present round is bounded by $\mathsf{poly}\log\lambda \cdot O(s)$.

5.6 Intermediate Building Blocks

We will adopt the constant-round, weakly space efficient, and communication efficient SMPC protocol of Section 5.3 among one to two small committee(s) to securely realize a few useful ideal functionalities which we can adopt as intermediate building blocks:

- $\mathcal{F}^{\mathsf{share}[i]}$ is the ideal functionality that allows machine i to secret share its state among the i-th committee; $\mathcal{F}^{\mathsf{share}[i]}$ involves $m'+1$ participants among whom at most $t'+1$ can be corrupt;
- $\mathcal{F}^{\mathsf{comp}^r[i]}$ is the ideal functionality that allows the i-th committee to jointly emulate the computation of machine i in the r-th round in the original MPC; $\mathcal{F}^{\mathsf{comp}^r[i]}$ involves m' participants among whom at most t' corruptions; and
- $\mathcal{F}^{\mathsf{send}[i,i']}$ is the ideal functionality that emulates machine i sending a message to machine i' in the original MPC; $\mathcal{F}^{\mathsf{send}[i,i']}$ involves $2m'$ participants (i.e., the sending and receiving committees), among whom at most $2t'$ can be corrupt.

More formally, to define each of $\mathcal{F}^{\mathsf{share}[i]}$, $\mathcal{F}^{\mathsf{comp}^r[i]}$, and $\mathcal{F}^{\mathsf{send}[i,i']}$, we only need to specify what the functions $\mathsf{share}[i]$, $\mathsf{comp}^r[i]$, and $\mathsf{send}[i,i']$ compute respectively and among which players.

1. $\mathsf{share}[i]$: a function that anticipates inputs from machine i as well as members of the i-th committee. The function ignores everyone's input and looks at only machine i's input henceforth denoted x, and computes $(\overline{x}_1, \ldots, \overline{x}_{m'}) \leftarrow \mathsf{Share}(x)$. It outputs $\perp$ to machine i, and $\overline{x}_j$ to the j-th member of the i-th committee for $j \in [m']$.
2. $\mathsf{comp}^r[i]$: the function $\mathsf{comp}^r[i]$ anticipates inputs from members of the i-th committee denoted $\overline{\mathsf{mem}}_1, \overline{\mathsf{mem}}_2, \ldots, \overline{\mathsf{mem}}_{m'}$. It internally evaluates $\mathsf{mem}' \leftarrow M_r^i(\mathsf{Recons}(\{\overline{x}_j\}_{j\in[m']}))$, and $(\overline{\mathsf{mem}}'_1, \ldots, \overline{\mathsf{mem}}'_{m'}) \leftarrow \mathsf{Share}(\mathsf{mem}')$. Now, the j-th member of the i-th committee is supposed to get $\overline{\mathsf{mem}}'_j$ for $j \in [m']$.
3. $\mathsf{send}[i, i']$: this function anticipates inputs from the i-th committee and the i'-th committee. It ignores the inputs from the i'-th committee, and looks at only inputs from the i-th committee henceforth denoted $\overline{x}_1, \ldots, \overline{x}_{m'}$. It internally evaluates $x \leftarrow \mathsf{Recons}(\overline{x}_1, \ldots, \overline{x}_{m'})$, and $\overline{y}_1, \ldots, \overline{y}_{m'} \leftarrow \mathsf{Share}(x)$. Now, $\overline{y}_j$ is meant as the output for the j-th member of the i'-th committee for $j \in [m']$.

5.7 Compilation to a Hybrid Protocol

Since a machine may participate in multiple committees, henceforth when the machine we are concerned with is clear from the context, we often use the notation $\overline{\mathsf{mem}}_j$ a machine's robust secret share pertaining to the j-th committee (assuming that i indeed participates in the j-th committee). Given an original communication-oblivious MPC protocol, our SMPC compiler works as follows where $\mathsf{PRF} : \{0,1\}^\kappa \times [m] \to [m]^{m'}$ denotes a pseudo-random function:

- *Initialize.* At protocol start, a random common reference string denoted $\mathsf{crs} \xleftarrow{\$} \{0,1\}^\kappa$ is chosen which will be used for committee election. Every machine i now receives an input x_i from the environment $\mathcal{Z}$, and each machine is also informed of crs.
- *Committee election.* Now, the pseudo-random string $\mathsf{PRF}_{\mathsf{crs}}(j)$ can be used to determine the m' members of the j-th committee for $j \in [m]$. Note that instead of storing all members of every committee, relying on the observations in Section 5.5.3, machines will instead compute all the relevant committee information on the fly, in all of the subsequent steps of the protocol; and this will not consume too much space. For simplicity, in our description below, we will not explicitly describe how a machine computes the relevant committee information needed in every round.
- *Secret-share input.* Each machine $i \in [m]$ sends its input x_i to $\mathcal{F}^{\mathsf{share}[i]}$. For every committee i serves on, machine i sends $\perp$ to $\mathcal{F}^{\mathsf{share}[i]}$. As a result every member of the i-th committee obtains a robust secret share of x_i from $\mathcal{F}^{\mathsf{share}[i]}$. When a machine i participating in the j-th committee receives a robust secret share $\overline{v}$ from $\mathcal{F}^{\mathsf{share}[j]}$, it sets $\overline{\mathsf{mem}}_j := \overline{v}$.
- *Emulate protocol.* For every round $r \in [R]$ where R denotes the worst-case round complexity of the original MPC protocol, every machine $i \in [m]$ does the following:
  - *Emulate computation:* For each committee $j \in [m]$ the machine i serves on, machine i sends $\overline{\mathsf{mem}}_j$ to $\mathcal{F}^{\mathsf{comp}^r[i]}$. It will obtain from $\mathcal{F}^{\mathsf{comp}^r[i]}$ an updated share of the new memory denoted $\overline{\mathsf{mem}}'$. Machine i overwrites its $\overline{\mathsf{mem}}_j$ variable with $\overline{\mathsf{mem}}'$.
  - *Emulate sending:* For each committee $j \in [m]$ the machine i serves on, do the following: recall that by WLOG_2, shares of the words to be sent are written at a designated location in $\overline{\mathsf{mem}}_j$. By WLOG_1, the total number of words committee j wants to send in this round $s' \le s$ is deterministic and a-priori known; and moreover the s' destination machines can be computed on the fly and written down in $O(s)$ space. Henceforth let $d_1, d_2, \ldots, d_{s'}$ be the $s' \le s$ destination machines' identifiers, and let $\overline{y}_1, \ldots, \overline{y}_{s'}$ denote the shares of the words to send to them respectively. For each d_k where $k \in [s']$, send $\overline{y}_k$ to $\mathcal{F}^{\mathsf{send}[j,d_k]}$.

- *Emulate receiving:* For every committee j' that machine i serves on, if in the original MPC machine $j \in [m]$ is supposed to send message to machine j' (all such j's can be computed and written down on the fly due to WLOG_1), then for every such j:
  1. send $\perp$ to $\mathcal{F}^{\mathsf{send}[j,j']}$ indicating participation as a receiver;
  2. the machine i will receive a secret share $\overline{y}$ from $\mathcal{F}^{\mathsf{send}[j,j']}$, now append $\overline{y}$ to its $\overline{\mathsf{mem}}_j$.
- *Output.* After emulating all rounds of the original MPC in the above manner, machine i does the following: for every committee j it serves on, send $\overline{\mathsf{mem}}_j$ to machine j. When each machine i receives shares from at least $2m'/3$ members of the i-th committee, call Recons with the $2m'/3$ shares received and output the corresponding output.

▶ **Theorem 23.** *Suppose that the PRF scheme employed is secure*[8]*, and that the MPC protocol* Π *to be compiled obliviously realizes the ideal functionality* f *by Definition 6. Then, if we apply the above compiler to* Π *to obtain a hybrid-world protocol* Π_{hyb}*,* Π_{hyb} *must securely realizes* $\mathcal{F}^f$ *by Definition 21, as long as the adversary controls no more than* $1/3 - \eta$ *fraction of the machines (for an arbitrarily small constant* η*).*

Proof. Let $\mathcal{A}$ be a PPT adversary and $\mathsf{Crupt} \subset [m]$ of size $|\mathsf{Crupt}| \leq (1/3 - \eta) \cdot m$ be the set of machines corrupted by $\mathcal{A}$. We first show that except with negligible probability, (i) all the committees elected by $\mathsf{PRF}_{\mathsf{crs}}$ have at most $(1/3 - \eta/2) \cdot m'$ corrupted machines, and (ii) each machine participates in at most $2m'$ committees.[9] Suppose the committees were elected using truly uniform randomness, then the expected corrupted machines in a committee is $m' \cdot |\mathsf{Crupt}|/m \leq (1/3 - \eta) \cdot m'$ and the expected number of committees a machine participate is m'. By a Chernoff Bound and union bound, both (i) and (ii) hold except with probability $e^{-\Omega(m')} = \mathsf{negl}(\lambda)$. Since both properties can be checked efficiently, by the security of PRF, the probability that the committees elected by $\mathsf{PRF}_{\mathsf{crs}}$ violates violate (i) or (ii) is at most $\mathsf{negl}(\kappa) + \mathsf{negl}(\lambda)$. Hence, in the rest of the proof, we assume both (i) and (ii) hold. We proceed to define a simulator $\mathcal{S}$:

- *Initialize and Committee election.* $\mathcal{S}$ simply simulate a random crs. There is no communication in the committee election step.
- *Secret-share input.* In this step, $\mathcal{S}$ extracts the adversary $\mathcal{A}$'s input $\{x_i\}_{\mathsf{Crupt}}$ from $\mathcal{F}^{\mathsf{share}[i]}$ for $i \in \mathsf{Crupt}$. $\mathcal{S}$ sends $\{x_i\}_{\mathsf{Crupt}}$ to the ideal functionality. For the output of $\mathcal{F}^{\mathsf{share}[\cdot]}$ that $\mathcal{A}$ receives, since each committee has at most $(1/3 - \eta/2) \cdot m'$ corrupted machines, $\mathcal{S}$ can simulate the shares outputted by each $\mathcal{F}^{\mathsf{share}[\cdot]}$ by generating a fresh $\mathsf{Share}(0)$.
- *Emulate protocol.* Note both $\mathcal{F}^{\mathsf{comp}^r[\cdot]}$ and $\mathcal{F}^{\mathsf{send}[\cdot,\cdot]}$ also output shares, by the same reason, $\mathcal{S}$ can simulate the shares outputted by each $\mathcal{F}^{\mathsf{comp}^r[\cdot]}$ and $\mathcal{F}^{\mathsf{send}[\cdot,\cdot]}$ to $\mathcal{A}$ by fresh secret sharings $\mathsf{Share}(0)$.
- *Output.* $\mathcal{S}$ sends `deliver` to the ideal functionality and receives the output $\{y_i\}_{i \in \mathsf{Crupt}}$. For each $i \in \mathsf{Crupt}$, $\mathcal{S}$ generates $\mathsf{Share}(y_i)$ and sends the shares that machine i should receive from the honest machines in the i-th committee.

By perfect privacy of the RSS, it is clear that the shares outputted by $\mathcal{F}^{\mathsf{share}[\cdot]}$, $\mathcal{F}^{\mathsf{comp}^r[\cdot]}$ and $\mathcal{F}^{\mathsf{send}[\cdot,\cdot]}$ are simulated perfectly.

---

[8] In fact, this hybrid-world theorem secures against even computationally unbounded adversaries despite the use of the PRF, since the PRF is used to defeat only the polynomial checkable function whether some committee has 1/3 or more corruption.

[9] In fact, property (ii) is not needed for proving security but we will use it to analyze efficiency of the protocol later.

Now, note that our protocol Π_{hyb} emulate the underlying MPC protocol by committees, where all the input and messages are stored by shares of the robust secret sharing scheme (RSS) in each committee. Since all committee has at most $(1/3 - \eta) \cdot m' < m'/3$ corrupted machines, by robustness of RSS, the adversary cannot change the value stored in the RSS. Hence the underlying MPC protocol is emulated correctly, and at the end, each machine $i \in \mathsf{Crupt}$ receives shares of $\mathsf{Share}(y_i)$ from the i-th committee. Hence, $\mathcal{S}$ also simulates the shares in the output step perfectly.

Therefore, Π_{hyb} securely realizes $\mathcal{F}^f$, in fact, with statistical security in this hybrid model. ◀

5.8 Compilation to a Real-World Protocol

In Section 5.7, we compiled a communication-oblivious MPC protocol Π to a secure counterpart Π_{hyb} assuming the existence of ideal functionalities $\mathcal{F}^{\mathsf{share}[i]}$, $\mathcal{F}^{\mathsf{comp}^r[i]}$, and $\mathcal{F}^{\mathsf{send}[i,i']}$. Eventually we would like to replace these ideal functionalities with real-world building blocks. This is easy:

- Let $\Pi^{\mathsf{share}[i]}$, $\Pi^{\mathsf{comp}^r[i]}$, and $\Pi^{\mathsf{send}[i,i']}$ be SMPC protocols that securely realize (by Definition 21) $\mathcal{F}^{\mathsf{share}[i]}$, $\mathcal{F}^{\mathsf{comp}^r[i]}$, and $\mathcal{F}^{\mathsf{send}[i,i']}$ respectively. Note that $\Pi^{\mathsf{share}[i]}$ is a protocol among machine i and the i-th committee, $\Pi^{\mathsf{comp}^r[i]}$ is a protocol among the i-th committee, and $\Pi^{\mathsf{send}[i,i']}$ is a protocol among the i-th committee and the i'-th committee.
- In the compiled hybrid-world protocol in Section 5.7, whenever a machine invokes some ideal functionality $\mathcal{F}^{\mathsf{share}[i]}$, $\mathcal{F}^{\mathsf{comp}^r[i]}$, or $\mathcal{F}^{\mathsf{send}[i,i']}$ with the input x, it now invokes the corresponding protocol, $\Pi^{\mathsf{share}[i]}$, $\Pi^{\mathsf{comp}^r[i]}$, or $\Pi^{\mathsf{send}[i,i']}$ respectively, also with the input x.
- In the compiled hybrid-world protocol in Section 5.7, whenever a machine is to receive output from the ideal functionality $\mathcal{F}^{\mathsf{share}[i]}$, $\mathcal{F}^{\mathsf{comp}^r[i]}$, or $\mathcal{F}^{\mathsf{send}[i,i']}$, it now instead receives the output from $\Pi^{\mathsf{share}[i]}$, $\Pi^{\mathsf{comp}^r[i]}$, or $\Pi^{\mathsf{send}[i,i']}$ respectively.

Note that due to the observations made in Section 5.5.3, at the beginning of every round, every machine can compute on the fly and temporarily store members of all committees relevant to itself in this round, including committees it serves on and committees it will interact with – and this will only incur $O(s) \cdot \mathsf{poly}\log\lambda$ additional space. Therefore, for every protocol $\Pi^{\mathsf{share}[i]}$, $\Pi^{\mathsf{comp}^r[i]}$, or $\Pi^{\mathsf{send}[i,i']}$ invoked, the machine already knows exactly who are the other machines involved in the protocol. Not only so, in fact, at the beginning of every round, a machine has written down a receiving schedule for this round, again consuming $O(s) \cdot \mathsf{poly}\log\lambda$ additional space. As mentioned, if a machine receives any message from unanticipated senders or excessive messages from anticipated senders, these messages get discarded immediately and will not be stored or processed.

Efficiency of the compiled protocol

Let Π_{real} denote the compiled real-world protocol by applying the compiler described above to an original MPC communication-oblivious protocol Π. Since all of $\Pi^{\mathsf{share}[i]}$, $\Pi^{\mathsf{comp}^r[i]}$, and $\Pi^{\mathsf{send}[i,i']}$ complete in constant number of rounds, clearly, the round complexity of Π_{real} is only a constant factor more than the original Π.

We now analyze the per-machine space complexity. We will use the fact that $\Pi^{\mathsf{share}[i]}$, $\Pi^{\mathsf{comp}^r[i]}$, and $\Pi^{\mathsf{send}[i,i']}$ are weakly space efficient. We first focus on the space expended by each machine *for every committee it serves on*. Recall that each robust secret share of a machine's state in the original MPC is only $O(s)$ in size. The dominant part is the space consumed during $\Pi^{\mathsf{share}[i]}$, $\Pi^{\mathsf{comp}^r[i]}$, and $\Pi^{\mathsf{send}[i,i']}$ protocols. Note that all of the

functions evaluated by $\Pi^{\mathsf{share}[i]}$, $\Pi^{\mathsf{comp}^r[i]}$, and $\Pi^{\mathsf{send}[i,i']}$ protocols have at most $O(s)$ RAM-space complexity. Now, observe that a RAM consuming space s can be converted to a layered circuit of width s. Recall also that at most $2m'$ machines participate in each of $\Pi^{\mathsf{share}[i]}$, $\Pi^{\mathsf{comp}^r[i]}$, and $\Pi^{\mathsf{send}[i,i']}$. Thus by weak space efficiency, the space consumed by each $\Pi^{\mathsf{share}[i]}$, $\Pi^{\mathsf{comp}^r[i]}$, or $\Pi^{\mathsf{send}[i,i']}$ instance is at most $O(m' \cdot s) \cdot \mathsf{poly}(\kappa)$. So far we have focused on the space per machine per committee it serves on. The total space consumption of any single machine is at most $2m' \cdot O(m' \cdot s) \cdot \mathsf{poly}(\kappa)$, where $2m'$ is an upper bound on the number of committees a machine can participate in by the proof of Theorem 23 (except for a negligible probability). Since $m' = \mathsf{poly} \log \lambda$, the total space per machine is upper bounded by $\mathsf{poly} \log \lambda \cdot \mathsf{poly}(\kappa) \cdot O(s)$ (for some other suitable polynomial poly).

Finally, the total communication is asymptotically no more than the total space by the communication efficiency requirement.

▶ **Corollary 24.** *Suppose that the PRF employed is secure, and that the* $\Pi^{\mathsf{share}[i]}$, $\Pi^{\mathsf{comp}^r[i]}$, *and* $\Pi^{\mathsf{send}[i,i']}$ *protocols employed securely realize* $\mathcal{F}^{\mathsf{share}[i]}$, $\mathcal{F}^{\mathsf{comp}^r[i]}$, *and* $\mathcal{F}^{\mathsf{send}[i,i']}$ *protocols respectively by Definition 21 as long as at least* $2/3$ *fraction of the machines participating in each protocol instance are honest; and moreover suppose that they are constant-round, weakly space efficiency, and communication efficient as defined in Section 5.3. Suppose that the MPC protocol* Π *obliviously realizes the ideal functionality* f *by Definition 6. Then, if we apply the above compiler to* Π *to obtain a real-world protocol* Π_{real}, Π_{real} *must securely realizes* $\mathcal{F}^f$ *by Definition 21. Moreover,* Π_{real}*'s round complexity is asymptotically the same as* Π*; its per-machine space consumption is at most* $O(s) \cdot \mathsf{poly}(\kappa)$ *where* κ *denotes the security parameter and* s *is the per-machine space of the original* Π*, and its total communication is upper bounded* $O(m \cdot s) \cdot \mathsf{poly}(\kappa)$.

Proof. As shown in the proof of Theorem 23, indeed, in each committee at least $2/3$ fraction of the machines must be honest. Now, security follows from a standard compositional argument: for any real-world adversary $\mathcal{A}$ attacking Π_{real}, we can construct a hybrid-world adversary $\mathcal{A}'$ that basically calls the simulators of all instances of $\Pi^{\mathsf{share}[i]}$, $\Pi^{\mathsf{comp}^r[i]}$, and $\Pi^{\mathsf{send}[i,i']}$, and no polynomial-time environment $\mathcal{Z}$ should be able distinguish whether it is in the real world interacting with $\mathcal{A}$ or the hybrid world interacting with $\mathcal{A}'$. Now, by Theorem 23, for this hybrid-world adversary $\mathcal{A}'$ corresponding to the real-world adversary $\mathcal{A}$, we can construct an ideal-world adversary $\mathcal{S}$, such that no polynomial-time environment $\mathcal{Z}$ can distinguish whether it is in the hybrid world or the ideal world. Thus we can conclude that no polynomial-time environment $\mathcal{Z}$ can distinguish whether it is in the real world interacting with $\mathcal{A}$ or the ideal world interacting with $\mathcal{S}$.

Finally, the efficiency statements follow from the analysis in the paragraph before the corollary. ◀

Main theorem statement for SMPC-for-MPC

Recall that the main theorem statement for our "MPC to SMPC-for-MPC" compiler is Theorem 2. We restate it below for the reader's convenience and complete our presentation with a proof.

▶ **Theorem 25** (Secure computation for MPC: Restatement of Theorem 2)**.** *Assume the existence of a common random string, the Learning With Errors (LWE) assumption, enhanced trapdoor permutations, as well as the existence of an FHE scheme with a suitable notion of compactness (see Appendix A.1 for a formal definition of compactness). Suppose that* $s = N^{\varepsilon}$ *and that* m *is upper bounded by a fixed polynomial in* N*. Let* κ *denote a security parameter,*

and assume that $s \geq \kappa$. Given any MPC algorithm Π that completes in R rounds where each of the m machines has s local space, there is an MPC algorithm $\widetilde{\Pi}$ that securely realizes the same function computed by Π in the presence of an adversary that statically corrupts at most $\frac{1}{3} - \eta$ fraction of the machines for an arbitrarily small constant η. Moreover, $\widetilde{\Pi}$ completes in $O(R)$ rounds, consumes at most $O(s) \cdot \mathsf{poly}(\kappa)$ space per-machine, and incurs $O(m \cdot s) \cdot \mathsf{poly}(\kappa)$ total communication per round.

Proof. We may first apply the communication-oblivious compiler corresponding to Theorem 1 to compile Π to a communication-oblivious counterpart Π', we then apply the compiler corresponding to Corollary 24 on Π'. The theorem then follows in a straightforward fashion due to Theorem 1 and Corollary 24. ◀

References

1 Arash Afshar, Zhangxiang Hu, Payman Mohassel, and Mike Rosulek. How to Efficiently Evaluate RAM Programs with Malicious Security. In *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*, pages 702–729, 2015.

2 Kook Jin Ahn and Sudipto Guha. Access to data and number of iterations: Dual primal algorithms for maximum matching under resource constraints. *ACM Transactions on Parallel Computing (TOPC)*, 4(4):17, 2018.

3 Prabhanjan Ananth, Yu-Chi Chen, Kai-Min Chung, Huijia Lin, and Wei-Kai Lin. Delegating RAM Computations with Adaptive Soundness and Privacy. In *Proceedings, Part II, of the 14th International Conference on Theory of Cryptography - Volume 9986*, 2016.

4 Alexandr Andoni, Aleksandar Nikolov, Krzysztof Onak, and Grigory Yaroslavtsev. Parallel algorithms for geometric graph problems. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 574–583, 2014. doi:10.1145/2591796.2591805.

5 Alexandr Andoni, Zhao Song, Clifford Stein, Zhengyu Wang, and Peilin Zhong. Parallel Graph Connectivity in Log Diameter Rounds. In *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 674–685, 2018.

6 Alexandr Andoni, Clifford Stein, and Peilin Zhong. Log Diameter Rounds Algorithms for 2-Vertex and 2-Edge Connectivity. *arXiv preprint*, 2019. arXiv:1905.00850.

7 Gilad Asharov, T-H. Hubert Chan, Kartik Nayak, Rafael Pass, Ling Ren, and Elaine Shi. Bucket Oblivious Sort: A Simple Oblivious Sort. In *SOSA*, 2019.

8 Gilad Asharov, Abhishek Jain, Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan, and Daniel Wichs. Multiparty Computation with Low Communication, Computation and Interaction via Threshold FHE. In *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings*, pages 483–501, 2012.

9 Gilad Asharov and Yehuda Lindell. A Full Proof of the BGW Protocol for Perfectly Secure Multiparty Computation. *J. Cryptol.*, 30(1):58–151, January 2017.

10 Sepehr Assadi. Simple Round Compression for Parallel Vertex Cover. *CoRR*, abs/1709.04599, 2017.

11 Sepehr Assadi, MohammadHossein Bateni, Aaron Bernstein, Vahab Mirrokni, and Cliff Stein. Coresets meet EDCS: algorithms for matching and vertex cover on massive graphs. *arXiv preprint*, 2017. arXiv:1711.03076.

12 Sepehr Assadi and Sanjeev Khanna. Randomized Composable Coresets for Matching and Vertex Cover. In *Proceedings of the 29th ACM Symposium on Parallelism in Algorithms and Architectures*, pages 3–12. ACM, 2017.

13 Sepehr Assadi, Xiaorui Sun, and Omri Weinstein. Massively Parallel Algorithms for Finding Well-Connected Components in Sparse Graphs. *CoRR*, abs/1805.02974, 2018. arXiv:1805.02974.

14 Saikrishna Badrinarayanan, Aayush Jain, Nathan Manohar, and Amit Sahai. Threshold Multi-Key FHE and Applications to Round-Optimal MPC. Cryptology ePrint Archive, Report 2018/580, 2018.

15 Bahman Bahmani, Ravi Kumar, and Sergei Vassilvitskii. Densest subgraph in streaming and mapreduce. *Proceedings of the VLDB Endowment*, 5(5):454–465, 2012.

16 Bahman Bahmani, Benjamin Moseley, Andrea Vattani, Ravi Kumar, and Sergei Vassilvitskii. Scalable k-means++. *Proceedings of the VLDB Endowment*, 5(7):622–633, 2012.

17 MohammadHossein Bateni, Aditya Bhaskara, Silvio Lattanzi, and Vahab Mirrokni. Distributed balanced clustering via mapping coresets. In *Advances in Neural Information Processing Systems*, pages 2591–2599, 2014.

18 D. Beaver, S. Micali, and P. Rogaway. The Round Complexity of Secure Protocols. In *Proceedings of the Twenty-second Annual ACM Symposium on Theory of Computing*, STOC '90, pages 503–513, New York, NY, USA, 1990. ACM. doi:10.1145/100216.100287.

19 Soheil Behnezhad, Mahsa Derakhshan, MohammadTaghi Hajiaghayi, and Richard M. Karp. Massively Parallel Symmetry Breaking on Sparse Graphs: MIS and Maximal Matching. *CoRR*, abs/1807.06701, 2018.

20 Soheil Behnezhad, MohammadTaghi Hajiaghayi, and David G Harris. Exponentially Faster Massively Parallel Maximal Matching. *arXiv preprint*, 2019. arXiv:1901.03744.

21 Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Foundations of garbled circuits. In *ACM Conference on Computer and Communications Security (CCS)*, 2012.

22 Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness Theorems for Non-cryptographic Fault-tolerant Distributed Computation. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, STOC '88, pages 1–10, 1988.

23 Dan Boneh, Rosario Gennaro, Steven Goldfeder, Aayush Jain, Sam Kim, Peter M. R. Rasmussen, and Amit Sahai. Threshold Cryptosystems from Threshold Fully Homomorphic Encryption. In *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part I*, pages 565–596, 2018.

24 Elette Boyle, Kai-Min Chung, and Rafael Pass. Large-Scale Secure Computation: Multi-party Computation for (Parallel) RAM Programs. In *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II*, pages 742–762, 2015.

25 Elette Boyle, Kai-Min Chung, and Rafael Pass. Oblivious Parallel RAM and Applications. In *Theory of Cryptography - 13th International Conference, TCC 2016-A, Tel Aviv, Israel, January 10-13, 2016, Proceedings, Part II*, pages 175–204, 2016.

26 Elette Boyle, Niv Gilboa, and Yuval Ishai. Breaking the Circuit Size Barrier for Secure Computation Under DDH. In *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part I*, pages 509–539, 2016.

27 Elette Boyle, Shafi Goldwasser, and Stefano Tessaro. Communication Locality in Secure Multi-party Computation: How to Run Sublinear Algorithms in a Distributed Setting. In *Proceedings of the 10th Theory of Cryptography Conference on Theory of Cryptography*, TCC'13, pages 356–376, Berlin, Heidelberg, 2013. Springer-Verlag. doi:10.1007/978-3-642-36594-2_21.

28 Zvika Brakerski, Shai Halevi, and Antigoni Polychroniadou. Four Round Secure Computation without Setup. In *TCC*, 2017.

29 Sebastian Brandt, Manuela Fischer, and Jara Uitto. Matching and MIS for Uniformly Sparse Graphs in the Low-Memory MPC Model. *CoRR*, abs/1807.05374, 2018.

30 R. Canetti. Universally Composable Security: A New Paradigm for Cryptographic Protocols. In *FOCS*, 2001.

31 Ran Canetti, Huijia Lin, Stefano Tessaro, and Vinod Vaikuntanathan. Obfuscation of Probabilistic Circuits and Applications. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *Theory of Cryptography*, pages 468–497, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

32 Hubert Chan, Kai-Min Chung, and Elaine Shi. On the Depth of Oblivious Parallel ORAM. In *Asiacrypt*, 2017.

33 T.-H. Hubert Chan, Yue Guo, Wei-Kai Lin, and Elaine Shi. Cache-Oblivious and Data-Oblivious Sorting and Applications. In *SODA*, pages 2201–2220. SIAM, 2018.

34 T.-H. Hubert Chan, Kartik Nayak, and Elaine Shi. Perfectly Secure Oblivious Parallel RAM. In *Theory of Cryptography - 16th International Conference, TCC 2018*, pages 636–668, 2018.

35 T.-H. Hubert Chan and Elaine Shi. Circuit OPRAM: Unifying Statistically and Computationally Secure ORAMs and OPRAMs. In *Theory of Cryptography - 15th International Conference, TCC*, pages 72–107, 2017.

36 Yi-Jun Chang, Manuela Fischer, Mohsen Ghaffari, Jara Uitto, and Yufan Zheng. The Complexity of (Delta+1) Coloring inCongested Clique, Massively Parallel Computation, and Centralized Local Computation. *arXiv preprint*, 2018. arXiv:1808.08419.

37 Yu-Chi Chen, Sherman S. M. Chow, Kai-Min Chung, Russell W. F. Lai, Wei-Kai Lin, and Hong-Sheng Zhou. Cryptography for Parallel RAM from Indistinguishability Obfuscation. In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science, Cambridge, MA, USA, January 14-16, 2016*, pages 179–190, 2016.

38 Kai-Min Chung and Luowen Qian. Adaptively Secure Garbling Schemes for Parallel Computations. In *TCC*, 2019.

39 Geoffroy Couteau. A Note on the Communication Complexity of Multiparty Computation in the Correlated Randomness Model. In *Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part II*, pages 473–503, 2019.

40 Artur Czumaj, Jakub Łącki, Aleksander Mądry, Slobodan Mitrović, Krzysztof Onak, and Piotr Sankowski. Round compression for parallel matching algorithms. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 471–484, 2018. doi:10.1145/3188745.3188764.

41 Rafael da Ponte Barbosa, Alina Ene, Huy L Nguyen, and Justin Ward. A New Framework for Distributed Submodular Maximization. In *FOCS*, pages 645–654, 2016.

42 Ivan Damgård and Yuval Ishai. Constant-round Multiparty Computation Using a Black-box Pseudorandom Generator. In *Proceedings of the 25th Annual International Conference on Advances in Cryptology*, CRYPTO'05, pages 378–394, Berlin, Heidelberg, 2005. Springer-Verlag.

43 Ivan Damgård and Yuval Ishai. Scalable Secure Multiparty Computation. In *Proceedings of the 26th Annual International Conference on Advances in Cryptology*, CRYPTO'06, pages 501–520, Berlin, Heidelberg, 2006. Springer-Verlag. doi:10.1007/11818175_30.

44 Ivan Damgård, Kasper Green Larsen, and Jesper Buus Nielsen. Communication Lower Bounds for Statistically Secure MPC, With or Without Preprocessing. In *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part II*, pages 61–84, 2019.

45 Ivan Damgård, Jesper Buus Nielsen, Antigoni Polychroniadou, and Michael A. Raskin. On the Communication Required for Unconditionally Secure Multiplication. In *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part II*, pages 459–488, 2016.

46 Alina Ene, Sungjin Im, and Benjamin Moseley. Fast clustering using MapReduce. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 681–689. ACM, 2011.

47 Alina Ene and Huy Nguyen. Random coordinate descent methods for minimizing decomposable submodular functions. In *International Conference on Machine Learning*, pages 787–795, 2015.

48 Michael J. Fischer, Nancy A. Lynch, and Michael Merritt. Easy Impossibility Proofs for Distributed Consensus Problems. In *Proceedings of the Fourth Annual ACM Symposium on Principles of Distributed Computing*, PODC '85, pages 59–70, New York, NY, USA, 1985. ACM. doi:10.1145/323596.323602.

49 Christopher Fletcher, Muhammad Naveed, Ling Ren, Elaine Shi, and Emil Stefanov. Bucket ORAM: Single Online Roundtrip, Constant Bandwidth Oblivious RAM. Cryptology ePrint Archive, Report 2015/1065, 2015. URL: `https://eprint.iacr.org/2015/1065`.

50 Christopher W. Fletcher, Ling Ren, Albert Kwon, Marten van Dijk, Emil Stefanov, and Srinivas Devadas. RAW Path ORAM: A low-latency, low-area hardware ORAM controller with integrity verification. *IACR Cryptology ePrint Archive*, 2014:431, 2014.

51 Christopher W. Fletcher, Ling Ren, Xiangyao Yu, Marten van Dijk, Omer Khan, and Srinivas Devadas. Suppressing the Oblivious RAM timing channel while making information leakage and program efficiency trade-offs. In *HPCA*, pages 213–224, 2014.

52 Buddhima Gamlath, Sagar Kale, Slobodan Mitrović, and Ola Svensson. Weighted Matchings via Unweighted Augmentations. *arXiv preprint*, 2018. `arXiv:1811.02760`.

53 Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate Indistinguishability Obfuscation and Functional Encryption for all circuits. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, 2013.

54 Sanjam Garg, Rafail Ostrovsky, and Akshayaram Srinivasan. Adaptive Garbled RAM from Laconic Oblivious Transfer. In *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part III*, pages 515–544, 2018.

55 Craig Gentry. Fully homomorphic encryption using ideal lattices. In *ACM symposium on Theory of computing (STOC)*, 2009.

56 Craig Gentry, Shai Halevi, Steve Lu, Rafail Ostrovsky, Mariana Raykova, and Daniel Wichs. Garbled RAM Revisited. In *EUROCRYPT*, pages 405–422, 2014.

57 Craig Gentry, Shai Halevi, Mariana Raykova, and Daniel Wichs. Outsourcing Private RAM Computation. In *FOCS*, 2014.

58 Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic Encryption from Learning with Errors: Conceptually-Simpler, Asymptotically-Faster, Attribute-Based. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013*, pages 75–92, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

59 Mohsen Ghaffari. Massively Parallel Algorithms. `http://people.csail.mit.edu/ghaffari/MPA19/Notes/MPA.pdf`.

60 Mohsen Ghaffari, Themis Gouleakis, Slobodan Mitrovic, and Ronitt Rubinfeld. Improved Massively Parallel Computation Algorithms for MIS, Matching, and Vertex Cover. *CoRR*, abs/1802.08237, 2018.

61 Mohsen Ghaffari, Silvio Lattanzi, and Slobodan Mitrović. Improved Parallel Algorithms for Density-Based Network Clustering. In *International Conference on Machine Learning*, pages 2201–2210, 2019.

62 Mohsen Ghaffari, Krzysztof Nowicki, and Mikkel Thorup. Faster Algorithms for Edge Connectivity via Random 2-Out Contractions, 2019. `arXiv:1909.00844`.

63 Mohsen Ghaffari and Jara Uitto. Sparsifying Distributed Algorithms with Ramifications in Massively Parallel Computation and Centralized Local Computation. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 1636–1653, 2019.

64 O. Goldreich. Towards a theory of software protection and simulation by oblivious RAMs. In *ACM Symposium on Theory of Computing (STOC)*, 1987.

65 O. Goldreich, S. Micali, and A. Wigderson. How to play ANY mental game. In *ACM symposium on Theory of computing (STOC)*, 1987.

66 Oded Goldreich and Rafail Ostrovsky. Software protection and simulation on oblivious RAMs. *J. ACM*, 1996.

67 M. Goodrich. Communication-Efficient Parallel Sorting. *SIAM Journal on Computing*, 29(2):416–432, 1999.

68 Michael T. Goodrich. Data-oblivious External-memory Algorithms for the Compaction, Selection, and Sorting of Outsourced Data. In *Proceedings of the Twenty-third Annual ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '11, pages 379–388, New York, NY, USA, 2011. ACM. `doi:10.1145/1989493.1989555`.

69 Michael T. Goodrich and Michael Mitzenmacher. Privacy-Preserving Access of Outsourced Data via Oblivious RAM Simulation. In *International Colloquium on Automata, Languages and Programming (ICALP)*, pages 576–587, 2011.

70 Michael T. Goodrich, Nodari Sitchinava, and Qin Zhang. Sorting, Searching, and Simulation in the MapReduce Framework. In *Algorithms and Computation*, pages 374–383, 2011.

71 S. Dov Gordon, Jonathan Katz, Vladimir Kolesnikov, Fernando Krell, Tal Malkin, Mariana Raykova, and Yevgeniy Vahlis. Secure two-party computation in sublinear (amortized) time. In *ACM Conference on Computer and Communications Security (CCS)*, 2012.

72 S. Dov Gordon, Feng-Hao Liu, and Elaine Shi. Constant-Round MPC with Fairness and Guarantee of Output Delivery. In *CRYPTO*, pages 63–82, 2015.

73 Vipul Goyal, Yanyi Liu, and Yifan Song. Communication-Efficient Unconditional MPC with Guaranteed Output Delivery. In *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part II*, pages 85–114, 2019.

74 Jens Groth and Rafail Ostrovsky. Cryptography in the Multi-string Model. In *CRYPTO*, 2007.

75 Yue Guo, Rafael Pass, and Elaine Shi. Synchronous, with a Chance of Partition Tolerance. In *CRYPTO*, 2019.

76 Sungjin Im, Benjamin Moseley, and Xiaorui Sun. Efficient massively parallel methods for dynamic programming. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 798–811, 2017. doi:10.1145/3055399.3055460.

77 Sungjin Im, Benjamin Moseley, and Xiaorui Sun. Efficient Massively Parallel Methods for Dynamic Programming. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2017, pages 798–811, New York, NY, USA, 2017. ACM.

78 Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Founding Cryptography on Oblivious Transfer — Efficiently. In *CRYPTO*, 2008.

79 Howard J. Karloff, Siddharth Suri, and Sergei Vassilvitskii. A Model of Computation for MapReduce. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, pages 938–948, 2010. doi:10.1137/1.9781611973075.76.

80 Ravi Kumar, Benjamin Moseley, Sergei Vassilvitskii, and Andrea Vattani. Fast Greedy Algorithms in MapReduce and Streaming. *TOPC*, 2(3):14:1–14:22, 2015. doi:10.1145/2809814.

81 Jakub Łącki, Vahab S. Mirrokni, and Michal Wlodarczyk. Connected Components at Scale via Local Contractions. *CoRR*, abs/1807.10727, 2018.

82 Leslie Lamport, Robert Shostak, and Marshall Pease. The Byzantine Generals Problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, July 1982.

83 Silvio Lattanzi, Benjamin Moseley, Siddharth Suri, and Sergei Vassilvitskii. Filtering: a method for solving graph problems in MapReduce. In *SPAA 2011: Proceedings of the 23rd Annual ACM Symposium on Parallelism in Algorithms and Architectures, San Jose, CA, USA, June 4-6, 2011 (Co-located with FCRC 2011)*, pages 85–94, 2011. doi:10.1145/1989493.1989505.

84 Yehuda Lindell, Benny Pinkas, Nigel P. Smart, and Avishay Yanai. Efficient Constant-Round Multi-party Computation Combining BMR and SPDZ. *J. Cryptol.*, 32(3):1026–1069, July 2019.

85 Chang Liu, Xiao Shaun Wang, Kartik Nayak, Yan Huang, and Elaine Shi. ObliVM: A programming framework for secure computation. In *IEEE Symposium on Security and Privacy*, 2015.

86 Steve Lu and Rafail Ostrovsky. How to Garble RAM Programs. In *EUROCRYPT*, 2013.

87 Steve Lu and Rafail Ostrovsky. Black-Box Parallel Garbled RAM. In *Advances in Cryptology – CRYPTO 2017*, pages 66–92, 2017.

88 Alfonso Cevallos Manzano. Reducing the Share Size in Robust Secret Sharing. Master's thesis, `http://algant.eu/documents/theses/cevallos.pdf`, 2011.

89 Vahab S. Mirrokni and Morteza Zadimoghaddam. Randomized Composable Core-sets for Distributed Submodular Maximization. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 153–162, 2015. `doi:10.1145/2746539.2746624`.

90 Baharan Mirzasoleiman, Amin Karbasi, Rik Sarkar, and Andreas Krause. Distributed submodular maximization: Identifying representative elements in massive data. In *Advances in Neural Information Processing Systems*, pages 2049–2057, 2013.

91 Pratyay Mukherjee and Daniel Wichs. Two Round Multiparty Computation via Multi-key FHE. In *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II*, pages 735–763, 2016.

92 Kartik Nayak, Xiao Shaun Wang, Stratis Ioannidis, Udi Weinsberg, Nina Taft, and Elaine Shi. GraphSC: Parallel Secure Computation Made Easy. In *IEEE S & P*, 2015.

93 Krzysztof Onak. Round Compression for Parallel Graph Algorithms in Strongly Sublinear Space. *CoRR*, abs/1807.08745, 2018.

94 Merav Parter and Eylon Yogev. Distributed Algorithms Made Secure: A Graph Theoretic Approach. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 1693–1710, 2019.

95 Merav Parter and Eylon Yogev. Secure Distributed Computing Made (Nearly) Optimal. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, PODC '19, pages 107–116, New York, NY, USA, 2019. ACM. `doi:10.1145/3293611.3331620`.

96 Sarvar Patel, Giuseppe Persiano, and Kevin Yeo. CacheShuffle: A Family of Oblivious Shuffles. In *45th International Colloquium on Automata, Languages, and Programming (ICALP 2018)*, 2018.

97 Vibhor Rastogi, Ashwin Machanavajjhala, Laukik Chitnis, and Anish Das Sarma. Finding connected components in map-reduce in logarithmic rounds. In *29th IEEE International Conference on Data Engineering, ICDE 2013, Brisbane, Australia, April 8-12, 2013*, pages 50–61, 2013.

98 Ling Ren, Xiangyao Yu, Christopher W. Fletcher, Marten van Dijk, and Srinivas Devadas. Design space exploration and optimization of path oblivious RAM in secure processors. In *ISCA*, pages 571–582, 2013.

99 Tim Roughgarden, Sergei Vassilvitskii, and Joshua R. Wang. Shuffles and Circuits: (On Lower Bounds for Modern Parallel Computation). In *Proceedings of the 28th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA 2016, Asilomar State Beach/Pacific Grove, CA, USA, July 11-13, 2016*, pages 1–12, 2016. `doi:10.1145/2935764.2935799`.

100 Elaine Shi, T.-H. Hubert Chan, Emil Stefanov, and Mingfei Li. Oblivious RAM with $O((\log N)^3)$ Worst-Case Cost. In *ASIACRYPT*, pages 197–214, 2011.

101 Emil Stefanov, Elaine Shi, and Dawn Song. Towards Practical Oblivious RAM. In *Network and Distributed System Security Symposium (NDSS)*, 2012.

102 Emil Stefanov, Marten van Dijk, Elaine Shi, Christopher Fletcher, Ling Ren, Xiangyao Yu, and Srinivas Devadas. Path ORAM – an Extremely Simple Oblivious RAM Protocol. In *ACM Conference on Computer and Communications Security (CCS)*, 2013.

103 Leslie G. Valiant. A Bridging Model for Parallel Computation. *Commun. ACM*, 33(8):103–111, August 1990.

104 Xiao Shaun Wang, T-H. Hubert Chan, and Elaine Shi. Circuit ORAM: On Tightness of the Goldreich-Ostrovsky Lower Bound. In *CCS*, 2015.

105 Andrew Chi-Chih Yao. Protocols for Secure Computations (Extended Abstract). In *IEEE symposium on Foundations of Computer Science (FOCS)*, 1982.

106 Andrew Chi-Chih Yao. How to generate and exchange secrets. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, 1986.
107 Grigory Yaroslavtsev and Adithya Vadapalli. Massively Parallel Algorithms and Hardness for Single-Linkage Clustering under ℓ_p-Distances. In *Proceedings of the 35th International Conference on Machine Learning*, 2018.

A Proof of Theorem 22: the CommitteeMPC Protocol

In this section, we construct a constant-round, weakly space efficient, and communication efficient SMPC protocol as stated in Theorem 22. Starting from a compact FHE, we first apply the universal thresholdizer of Boneh et al. [23] to obtain a compact threshold FHE. We show that the resulting threshold FHE has desired security by Definition 21. Thus we can use it to construct a semi-malicious secure constant-round, weakly space efficient SMPC in a trusted setup model. The protocol can then be converted to a maliciously secure one by a generic transformation [14, 75], and the setup can be removed by invoking the SMPC protocol of Badrinarayanan et al. [14]. We start with the definitions.

Notation

We will use the variable m to denote the number of machines, although keep in mind that when the SMPC protocol in this section is employed in our "MPC to SMPC-for-MPC" compiler, this SMPC building block is in fact applied to at most $2m' = \mathsf{poly}\log\lambda$ number of machines.

A.1 Preliminaries

Fully Homomorphic Encryption

We first define fully homomorphic encryption schemes (FHE) with a strong compactness property. A FHE scheme is a tuple of PPT algorithms $\Pi_{\mathsf{FHE}} = (\mathsf{FHE.KeyGen}, \mathsf{FHE.Enc}, \mathsf{FHE.Eval}, \mathsf{FHE.Dec})$ defined as follows:

- $\mathsf{FHE.KeyGen}(1^\kappa) \to (\mathsf{pk}, \mathsf{sk})$: On input the security parameter κ, the key generation algorithm outputs a public key pk and a secret key sk.
- $\mathsf{FHE.Enc}(\mathsf{pk}, x) \to \mathsf{ct}$: On input a public key pk and a message $x \in \{0,1\}$, the encryption algorithm outputs a ciphertext ct. For convenience, for a message $x \in \{0,1\}^\ell$, we use $\mathsf{FHE.Enc}(\mathsf{pk}, x) = \mathsf{FHE.Enc}(\mathsf{pk}, x_1), \ldots \mathsf{FHE.Enc}(\mathsf{pk}, x_\ell)$ to denote the bit by bit encryptions of x.
- $\mathsf{FHE.Eval}(\mathsf{pk}, C, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell) \to \hat{\mathsf{ct}}$: On input a public key pk, a circuit $C : \{0,1\}^\ell \to \{0,1\}$ and ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_\ell$, the homomorphic evaluation algorithm outputs another ciphertext $\hat{\mathsf{ct}}$.
- $\mathsf{FHE.Dec}(\mathsf{sk}, \hat{\mathsf{ct}}) \to \hat{\mu}$: On input a secret key sk and a ciphertext $\hat{\mathsf{ct}}$, the decryption algorithm outputs a bit $\hat{\mu}$.

Correctness

We require that for all $\kappa \in \mathbb{N}$, $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{FHE.KeyGen}(1^\kappa)$, circuit $C : \{0,1\}^\ell \to \{0,1\}$ and corresponding inputs $x_1, \ldots, x_\ell \in \{0,1\}$, it holds that

$$\Pr[\mathsf{FHE.Dec}(\mathsf{sk}, \mathsf{FHE.Eval}(\mathsf{pk}, C, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell)) \neq C(x_1, \ldots, x_\ell)] \leq \mathsf{negl}(\kappa)$$

where $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{FHE.KeyGen}(1^k)$ and $\mathsf{ct}_i \leftarrow \mathsf{FHE.Enc}(\mathsf{pk}, x_i)$.

Security

We require the usual semantic security. Namely, we require that for all $\kappa \in \mathbb{N}$, $(\mathsf{pk}, \mathsf{Enc}(\mathsf{pk}, 0)) \approx_c (\mathsf{pk}, \mathsf{Enc}(\mathsf{pk}, 1))$, where $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{FHE.KeyGen}(1^k)$.

Compactness

We require the following strong compactness property. There exists a polynomial poly such that the following holds. $|\mathsf{pk}|, |\mathsf{sk}|, |\mathsf{ct}| \leq \mathsf{poly}(\kappa)$ for the public and secret key, and any ciphertext ct generated from the algorithms of FHE. Furthermore, for a layered circuit[10] C with width w, homomorphic evaluation of C can be done in space $\mathsf{poly}(\kappa) \times w$, independent of the size or depth of the circuit.[11]

FHE schemes with this compactness property can be achieved by assuming circular security for standard FHE constructions [55, 58] or using indistinguishability obfuscation [31, 53].

Universal Thresholdizer

The following definition of universal thresholdizer is taken from Boneh et al. [23], who constructed universal thresholdizer based on the learning with error assumption. For our purpose, we do not require the verification algorithm, so we omit it from the definition for simplicity.

▶ **Definition 26.** *Fix a security parameter κ and a data space $\mathcal{X}$. A universal thresholdizer scheme is a tuple of algorithm $\Pi_{\mathsf{UT}} = (\mathsf{UT.Setup}, \mathsf{UT.Eval}, \mathsf{UT.Combine})$ defined as follows:*

- $\mathsf{UT.Setup}(1^\kappa, 1^m, 1^t, 1^d, x) \to (\mathsf{pp}, \{\mathsf{sk}_i\}_{i\in[m]})$*: On input the security parameter κ, a number of users in the system m, a threshold $t \in [m]$, a bound on the depth d, and a secret $x \in \mathcal{X}$, the setup algorithm generates the public parameters pp and a set of secret keys $\mathsf{sk}_1, \ldots, \mathsf{sk}_m$ for each user in the system.*
- $\mathsf{UT.Eval}(\mathsf{pp}, \mathsf{sk}_i, C) \to p_i$*: On input the public parameters pp, a secret key sk_i, and a circuit C, the evaluation algorithm outputs a partial evaluation p_i.*
- $\mathsf{UT.Combine}(\mathsf{pp}, \{p_i\}_{i\in S}) \to \hat{\mu}$*: On input the public parameter pp, and a set of partial evaluations $\{p_i\}_{i\in S}$, the combining algorithm outputs the final evaluation μ.*

Evaluation Correctness

We say that a universal thresholdizer scheme $\Pi_{\mathsf{UT}} = (\mathsf{UT.Setup}, \mathsf{UT.Eval}, \mathsf{UT.Combine})$ satisfies evaluation correctness if the following conditions are true. For all $\kappa, m, t, d \in \mathbb{N}, x \in \mathcal{X}$, let $(\mathsf{pp}, \{sk_i\}_{i\in[m]}) \leftarrow \mathsf{UT.Setup}(1^\kappa, 1^m, 1^t, 1^d, x)$, $S \subset [m]$ of size $|S| = t$, and circuit $C : \mathcal{X} \to \{0, 1\}$ of depth at most d, we have that

$$\Pr[\mathsf{UT.Combine}(\mathsf{pp}, \{\mathsf{UT.Eval}(\mathsf{pp}, \mathsf{sk}_i, C)\}_{i\in S}) = C(x)] = 1 - \mathsf{negl}(\kappa),$$

where the probability is over the randomness of $\mathsf{UT.Setup}, \mathsf{UT.Eval}$, and $\mathsf{UT.Combine}$.

[10] A circuit is layered if the circuit can be represented as a layered graph with no wires crossing the layers.
[11] Here the space complexity measures the working space of $\mathsf{FHE.Eval}$ but not the description size of C. Looking ahead, we will consider uniform circuits whose description can be generated in small space.

Privacy

We say that a universal thresholdizer scheme $\Pi_{\mathsf{UT}} = (\mathsf{UT.Setup}, \mathsf{UT.Eval}, \mathsf{UT.Combine})$ satisfies privacy if there exists a PPT simulator Sim such that for all $\kappa \in \mathbb{N}$, polynomial m, t, d, PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$, there exists a negligible function $\mathsf{negl}(\kappa)$ such that

$$\left|\Pr[\mathsf{Expt}^{\mathsf{Real}}_{\Pi_{\mathsf{UT}},\mathcal{A}}(\kappa, m, t, d) = 1] - \Pr[\mathsf{Expt}^{\mathsf{Rand}}_{\Pi_{\mathsf{UT}},\mathcal{A}}(\kappa, m, t, d) = 1]\right| \leq \mathsf{negl}(\kappa)$$

where the experiments $\mathsf{Expt}^{\mathsf{Real}}_{\Pi_{\mathsf{UT}},\mathcal{A}}$ and $\mathsf{Expt}^{\mathsf{Rand}}_{\Pi_{\mathsf{UT}},\mathcal{A}}$ are defined as follows:

- $\mathsf{Expt}^{\mathsf{Real}}_{\Pi_{\mathsf{UT}},\mathcal{A}}(\kappa, m, t, d)$:
  1. $(x^*, \mathsf{st}_1) \leftarrow \mathcal{A}_1(1^\kappa)$.
  2. $(\mathsf{pp}, \{\mathsf{sk}_i\}_{i\in[m]}) \leftarrow \mathsf{UT.Setup}(1^\kappa, 1^m, 1^t, 1^d, \underline{x^*})$.
  3. $(S^*, \mathsf{st}_2) \leftarrow \mathcal{A}_2(\mathsf{pp}, \mathsf{st}_1)$ where $|S^*| = t - 1$.
  4. $b \leftarrow \mathcal{A}_3^{\mathcal{O}_{\mathsf{Eval}}(\underline{\{\mathsf{sk}_i\}_{i\in[m]}}, \cdot, \cdot)}(\{\mathsf{sk}_i\}_{i\in S^*}, \mathsf{st}_2)$.
  5. Output b.
- $\mathsf{Expt}^{\mathsf{Rand}}_{\Pi_{\mathsf{UT}},\mathcal{A}}(\kappa, m, t, d)$:
  1. $(x^*, \mathsf{st}_1) \leftarrow \mathcal{A}_1(1^\kappa)$.
  2. $(\mathsf{pp}, \{\mathsf{sk}_i\}_{i\in[m]}) \leftarrow \mathsf{UT.Setup}(1^\kappa, 1^m, 1^t, 1^d, \underline{0^{|x^*|}})$.
  3. $(S^*, \mathsf{st}_2) \leftarrow \mathcal{A}_2(\mathsf{pp}, \mathsf{st}_1)$ where $|S^*| = t - 1$.
  4. $b \leftarrow \mathcal{A}_3^{\mathsf{Sim}^{\mathcal{O}_{\mathsf{Sim}}(\cdot)}(\underline{\{\mathsf{sk}_i\}_{i\in S^*}}, \cdot, \cdot)}(\{\mathsf{sk}_i\}_{i\in S^*}, \mathsf{st}_2)$.
  5. Output b.

where the oracles $\mathcal{O}_{\mathsf{Eval}}(\{\mathsf{sk}_i\}_{i\in[m]}, \cdot, \cdot)$ and $\mathcal{O}_{\mathsf{Sim}}(\cdot)$ are defined as follows

- $\mathcal{O}_{\mathsf{Eval}}(\{\mathsf{sk}_i\}_{i\in[m]}, C, j)$: On input the set of key $\{\mathsf{sk}_i\}_{i\in[m]}$, a circuit C, and an index $j \in [m]\backslash S^*$, outputs $\mathsf{UT.Eval}(\mathsf{pp}, \mathsf{sk}_j, C)$.
- $\mathcal{O}_{\mathsf{Sim}}(C)$: On input a circuit C, if there exists a query (C, j) for some $j \in [m]\backslash S^*$ previously made by $\mathcal{A}_3$, the algorithm outputs $C(x^*)$. Otherwise, it outputs $\perp$.

A.2 Threshold FHE

We now define a notion of threshold FHE schemes with a simulation security that is sufficient to directly construct a semi-malicious SMPC protocol in a trusted setup model. We then show that applying the above universal thresholdizer to a FHE scheme yields such a threshold FHE scheme.

Syntax

A threshold FHE scheme is a tuple of PPT algorithms $\Pi_{\mathsf{TFHE}} = (\mathsf{TFHE.Setup}, \mathsf{TFHE.SimSetup}, \mathsf{TFHE.Enc}, \mathsf{TFHE.Eval}, \mathsf{TFHE.PartDec}, \mathsf{TFHE.FinDec})$ defined as follows:

- $\mathsf{TFHE.Setup}(1^\kappa, 1^m, 1^t) \to (\mathsf{tpk}, \{\mathsf{tsk}_i\}_{i\in[m]})$: On input the security parameter κ, a number of users in the system m, and a threshold $t \in [m]$, the setup algorithm generates the public key tpk and a set of secret keys $\mathsf{tsk}_1, \ldots, \mathsf{tsk}_m$ for each user in the system.
- $\mathsf{TFHE.SimSetup}(1^\kappa, 1^m, 1^t) \to (\mathsf{tpk}, \{\mathsf{tsk}_i\}_{i\in[m]})$: On input the security parameter κ, a number of users in the system m, and a threshold $t \in [m]$, the simulation setup algorithm generates the simulated public key tpk and a set of simulated secret keys $\mathsf{tsk}_1, \ldots, \mathsf{tsk}_m$ for each user in the system.
- $\mathsf{TFHE.Enc}(\mathsf{tpk}, x) \to \mathsf{ct}$: On input a public key tpk and a message $x \in \{0, 1\}$, the encryption algorithm outputs a ciphertext ct.

- $\mathsf{TFHE.Eval}(\mathsf{tpk}, C, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell) \to \hat{\mathsf{ct}}$: On input a public key tpk, a circuit $C : \{0,1\}^\ell \to \{0,1\}$ and ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_\ell$, the homomorphic evaluation algorithm outputs another ciphertext $\hat{\mathsf{ct}}$.
- $\mathsf{TFHE.PartDec}(i, \mathsf{tpk}, \mathsf{tsk}_i, \hat{\mathsf{ct}}) \to p_i$: On input an index $i \in [m]$, a secret key tsk_i, and a ciphertext $\hat{\mathsf{ct}}$, the partial decryption algorithm outputs a partial decryption p_i.
- $\mathsf{TFHE.FinDec}(\mathsf{tpk}, \{p_i\}_{i\in S}) \to \hat{\mu}$: On input the public key tpk, and a set of partial decryptions $\{p_i\}_{i\in S}$, the final decryption algorithm outputs the final decryption value $\hat{\mu}$.

Correctness

We require that for all $\kappa, m, t \in \mathbb{N}$, $S \subset [m]$, circuit $C : \{0,1\}^\ell \to \{0,1\}$ and corresponding input $x_1, \ldots, x_\ell \in \{0,1\}$, the following holds except with negligible probability in κ: Let $(\mathsf{tpk}, \{\mathsf{tsk}_i\}_{i\in[m]}) \leftarrow \mathsf{TFHE.Setup}(1^\kappa, 1^m, 1^t)$, $\mathsf{ct}_i \leftarrow \mathsf{TFHE.Enc}(\mathsf{tpk}, x_i)$ for $i \in [\ell]$, let $\hat{\mathsf{ct}} = \mathsf{TFHE.Eval}(\mathsf{tpk}, C, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell)$, let $p_i \leftarrow \mathsf{TFHE.PartDec}(i, \mathsf{tpk}, \mathsf{tsk}_i, \hat{\mathsf{ct}})$, and $\hat{\mu} \leftarrow \mathsf{TFHE.FinDec}(\mathsf{tpk}, \{p_i\}_{i\in S})$. Then $\hat{\mu} = C(x_1, \ldots, x_\ell)$ if $|S| \geq t$, and $\hat{\mu} = \perp$ otherwise.

Compactness

We require the same compactness property as in FHE. There exists a polynomial poly such that the following holds. $|\mathsf{pk}|, |\mathsf{sk}|, |\mathsf{ct}| \leq \mathsf{poly}(\kappa)$ for the public and secret key, and any ciphertext ct generated from the algorithms of TFHE. Furthermore, for a layered circuit C with width w, homomorphic evaluation of C can be done in space $\mathsf{poly}(\kappa) \times w$, independent of the size or depth of the circuit.

Simulation Security

We require the following simulation-based security for the purpose of constructing SMPC protocols. There exists PPT algorithms $\mathsf{Sim}_1, \mathsf{Sim}_2$ such that for all κ, polynomial $m, t, s \in \mathbb{N}$ with $m \geq 3t + 1$, $S \subset [m]$ of size $|S| \leq t$, polynomial size circuits $C_j : \{0,1\}^{m\cdot s} \to \{0,1\}$ for $j \in S$, PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$, there exists a negligible function $\mathsf{negl}(\kappa)$ such that

$$\left|\Pr[\mathsf{Expt}^{\mathsf{Real}}_{\Pi_{\mathsf{TFHE}},\mathcal{A}}(\kappa, m, t, S, \{C_j\}_{j\in S}) = 1] - \Pr[\mathsf{Expt}^{\mathsf{Ideal}}_{\Pi_{\mathsf{TFHE}},\mathcal{A}}(\kappa, m, t, S, \{C_j\}_{j\in S}) = 1]\right| \leq \mathsf{negl}(\kappa)$$

where the experiments $\mathsf{Expt}^{\mathsf{Real}}_{\Pi_{\mathsf{TFHE}},\mathcal{A}}$ and $\mathsf{Expt}^{\mathsf{Ideal}}_{\Pi_{\mathsf{TFHE}},\mathcal{A}}$ are defined as follows:

- $\mathsf{Expt}^{\mathsf{Real}}_{\Pi_{\mathsf{TFHE}},\mathcal{A}}(\kappa, m, t, S, \{C_j\}_{j\in S})$:
  1. $(\mathsf{tpk}, \{\mathsf{tsk}_i\}_{i\in[m]}) \leftarrow \mathsf{TFHE.Setup}(1^\kappa, 1^m, 1^t)$.
  2. $(\{x_i\}_{i\in[m]\setminus S}, \mathsf{st}_1) \leftarrow \mathcal{A}_1(\mathsf{tpk}, \{\mathsf{tsk}_i\}_{i\in S})$ where $x_i \in \{0,1\}^{\beta s}$.
  3. $\mathsf{ct}_i \leftarrow \mathsf{TFHE.Enc}(\mathsf{tpk}, x_i)$ for $i \in [m]\setminus S$.
  4. $(\{(x_i, r_i^{\mathsf{Enc}})\}_{i\in S}, \mathsf{st}_2) \leftarrow \mathcal{A}_2(\mathsf{st}_1, \{\mathsf{ct}_i\}_{i\in[m]\setminus S})$.
  5. $\mathsf{ct}_i \leftarrow \mathsf{TFHE.Enc}(\mathsf{tpk}, x_i; r_i^{\mathsf{Enc}})$ for $i \in S$; $\hat{\mathsf{ct}}_j \leftarrow \mathsf{TFHE.Eval}(\mathsf{tpk}, C_j, \{\mathsf{ct}_i\}_{i\in[m]})$ for $j \in S$.
  6. $p_{j,i} \leftarrow \mathsf{TFHE.PartDec}(i, \mathsf{tpk}, \mathsf{tsk}_i, \hat{\mathsf{ct}}_j)$ for $i \in [m]\setminus S$ and $j \in S$.
  7. $b \leftarrow \mathcal{A}_3(\mathsf{st}_2, \{p_{j,i}\}_{i\in[m]\setminus S, j\in S})$.
  8. Output b.
- $\mathsf{Expt}^{\mathsf{Ideal}}_{\Pi_{\mathsf{TFHE}},\mathcal{A}}(\kappa, m, t, S, \{C_j\}_{j\in S})$:
  1. $(\mathsf{tpk}, \{\mathsf{tsk}_i\}_{i\in[m]}) \leftarrow \mathsf{TFHE.SimSetup}(1^\kappa, 1^m, 1^t)$.
  2. $(\{x_i\}_{i\in[m]\setminus S}, \mathsf{st}_1) \leftarrow \mathcal{A}_1(\mathsf{tpk}, \{\mathsf{tsk}_i\}_{i\in S})$ where $x_i \in \{0,1\}^{\beta s}$.
  3. $(\{\mathsf{ct}_i\}_{i\in[m]\setminus S}, \mathsf{st}_1) \leftarrow \mathsf{Sim}_1(\mathsf{tpk}, \{\mathsf{tsk}_i\}_{i\in S})$
  4. $(\{(x_i, r_i^{\mathsf{Enc}})\}_{i\in S}, \mathsf{st}_2) \leftarrow \mathcal{A}_2(\mathsf{st}_1, \{\mathsf{ct}_i\}_{i\in[m]\setminus S})$.
  5. $\mathsf{ct}_i \leftarrow \mathsf{TFHE.Enc}(\mathsf{tpk}, x_i; r_i^{\mathsf{Enc}})$ for $i \in S$; $\hat{\mathsf{ct}}_j \leftarrow \mathsf{TFHE.Eval}(\mathsf{tpk}, C_j, \{\mathsf{ct}_i\}_{i\in[m]})$ for $j \in S$.

6. $\{p_{j,i}\}_{i\in[m]\setminus S, j\in S} \leftarrow \mathsf{Sim}_2(\mathsf{st}_1, \{(x_j, r_j^{\mathsf{Enc}})\}_{j\in S}, \{\hat{\mu}_j\}_{j\in S})$, where $\hat{\mu}_j = C_j(x_1, \ldots, x_m)$ for $j \in S$.
7. $b \leftarrow \mathcal{A}_3(\mathsf{st}_2, \{p_{j,i}\}_{i\in[m]\setminus S, j\in S})$.
8. Output b.

Construction

We show that applying the universal thresholdizer to a FHE scheme yields a threshold FHE scheme with above security. Let $\Pi_{\mathsf{FHE}} = (\mathsf{FHE.KeyGen}, \mathsf{FHE.Enc}, \mathsf{FHE.Eval}, \mathsf{FHE.Dec})$ be a FHE scheme with circular security. Let $\Pi_{\mathsf{UT}} = (\mathsf{UT.Setup}, \mathsf{UT.Eval}, \mathsf{UT.Combine})$ be a universal thresholdizer. Formally, we construct a threshold FHE scheme as follows.

- $\mathsf{TFHE.Setup}(1^\kappa, 1^m, 1^t)$: Run $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{FHE.KeyGen}(1^\kappa)$. Let d be the circuit depth of FHE decryption algorithm $\mathsf{FHE.Dec}$. Run $(\mathsf{pp}, \{\mathsf{sk}_i\}_{i\in[m]}) \leftarrow \mathsf{UT.Setup}(1^\kappa, 1^m, 1^{t+1}, 1^d, \mathsf{sk})$. Let $\mathsf{tpk} = (\mathsf{pp}, \mathsf{pk})$ and $\mathsf{tsk}_i = \mathsf{sk}_i$ for $i \in [m]$. Output $(\mathsf{tpk}, \{\mathsf{tsk}_i\}_{i\in[m]})$.
- $\mathsf{TFHE.SimSetup}(1^\kappa, 1^m, 1^t)$: Run $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{FHE.KeyGen}(1^\kappa)$. Let d be the circuit depth of FHE decryption algorithm $\mathsf{FHE.Dec}$. Run $(\mathsf{pp}, \{\mathsf{sk}_i\}_{i\in[m]}) \leftarrow \mathsf{UT.Setup}(1^\kappa, 1^m, 1^{t+1}, 1^d, 0^{|\mathsf{sk}|})$. Let $\mathsf{tpk} = (\mathsf{pp}, \mathsf{pk})$ and $\mathsf{tsk}_i = \mathsf{sk}_i$ for $i \in [m]$. Output $(\mathsf{tpk}, \{\mathsf{tsk}_i\}_{i\in[m]})$.
- $\mathsf{TFHE.Enc}(\mathsf{tpk}, x)$: Output $\mathsf{ct} \leftarrow \mathsf{FHE.Enc}(\mathsf{pk}, x)$.
- $\mathsf{TFHE.Eval}(\mathsf{tpk}, C, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell)$: Output $\hat{\mathsf{ct}} \leftarrow \mathsf{FHE.Eval}(\mathsf{pk}, C, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell)$.
- $\mathsf{TFHE.PartDec}(i, \mathsf{tpk}, \mathsf{tsk}_i, \hat{\mathsf{ct}})$: Output $p_i \leftarrow \mathsf{UT.Eval}(\mathsf{pp}, \mathsf{tsk}_i, \mathsf{FHE.Dec}(\cdot, \hat{\mathsf{ct}}))$.
- $\mathsf{TFHE.FinDec}(\mathsf{tpk}, \{p_i\}_{i\in S})$: Output $\hat{\mu} \leftarrow \mathsf{UT.Combine}(\mathsf{pp}, \{p_i\}_{i\in S})$.

It is clear by inspection that correctness follows by that of the underlying FHE scheme and universal thresholdizer. For compactness, note that universal thresholdizer is applied to evaluate the FHE decryption circuit $\mathsf{FHE.Dec}$, which has a fixed polynomial complexity in κ. Hence, the complexity of universal thresholdizer is upper bounded by a fixed $\mathsf{poly}(\kappa)$ and compactness follows by compactness of the underlying FHE scheme.

Security

We now show that the above construction satisfies simulation security defined above. We define simulators Sim_1, Sim_2 using the simulator of the universal thresholdizer (denoted by $\mathsf{UT.Sim}$) as follows.

- $\mathsf{Sim}_1(\mathsf{tpk}, \{\mathsf{tsk}_i\}_{i\in S})$: Simply run $\mathsf{ct}_i \leftarrow \mathsf{FHE.Enc}(\mathsf{pk}, 0^{\beta s})$ for $i \in [m]\setminus S$, store $\mathsf{tpk}, \{\mathsf{tsk}_i\}_{i\in S}$ in st_1, and output $(\{\mathsf{ct}_i\}_{i\in[m]\setminus S}, \mathsf{st}_1)$.
- $\mathsf{Sim}_2(\mathsf{st}_1, \{(x_i, r_i^{\mathsf{Enc}})\}_{i\in S}, \{\hat{\mu}_j\}_{j\in S})$: Run $p_{j,i} \leftarrow \mathsf{UT.Sim}^{\mathcal{O}_{\mathsf{Sim}}}(\{\mathsf{sk}_i\}_{i\in S}, C_j, i)$ for $i \in [m]\setminus S$ and $j \in S$, where $\mathcal{O}_{\mathsf{Sim}}$ on input query C' returns $\hat{\mu}_j$ if $C' = C_j$ for $j \in S$, and $\perp$ otherwise. Output $\{p_{j,i}\}_{i\in[m]\setminus S, j\in S}$.

Indistinguishability of $\mathsf{Expt}^{\mathsf{Real}}_{\Pi_{\mathsf{TFHE}},\mathcal{A}}$ and $\mathsf{Expt}^{\mathsf{Ideal}}_{\Pi_{\mathsf{TFHE}},\mathcal{A}}$ follows by considering a hybrid experiment that runs $\mathsf{TFHE.SimSetup}$ and Sim_2 in Step 1 and 6 respectively as the ideal experiment, but still encrypts x_i in Step 3. Formally, we define

- $\mathsf{Expt}^{\mathsf{Hyb}}_{\Pi_{\mathsf{TFHE}},\mathcal{A}}(\kappa, m, t, C, S)$:
  1. $(\mathsf{tpk}, \{\mathsf{tsk}_i\}_{i\in[m]}) \leftarrow \mathsf{TFHE.SimSetup}(1^\kappa, 1^m, 1^t)$.
  2. $(\{x_i\}_{i\in[m]\setminus S}, \mathsf{st}_1) \leftarrow \mathcal{A}_1(\mathsf{tpk}, \{\mathsf{tsk}_i\}_{i\in S})$ where $x_i \in \{0,1\}^{\beta s}$.
  3. $\mathsf{ct}_i \leftarrow \mathsf{TFHE.Enc}(\mathsf{tpk}, x_i)$ for $i \in [m]\setminus S$.
  4. $(\{(x_i, r_i^{\mathsf{Enc}})\}_{i\in S}, \mathsf{st}_2) \leftarrow \mathcal{A}_2(\mathsf{st}_1, \{\mathsf{ct}_i\}_{i\in[m]\setminus S})$.
  5. $\mathsf{ct}_i \leftarrow \mathsf{TFHE.Enc}(\mathsf{tpk}, x_i; r_i^{\mathsf{Enc}})$ for $i \in S$; $\hat{\mathsf{ct}}_j \leftarrow \mathsf{TFHE.Eval}(\mathsf{tpk}, C_j, \{\mathsf{ct}_i\}_{i\in[m]})$ for $j \in S$.
  6. $\{p_{j,i}\}_{i\in[m]\setminus S, j\in S} \leftarrow \mathsf{Sim}_2(\mathsf{st}_1, \{(x_j, r_j^{\mathsf{Enc}})\}_{j\in S}, \{\hat{\mu}_j\}_{j\in S})$, where $\hat{\mu}_j = C_j(x_1, \ldots, x_m)$ for $j \in S$.
  7. $b \leftarrow \mathcal{A}_3(\mathsf{st}_2, \{p_{j,i}\}_{i\in[m]\setminus S, j\in S})$.
  8. Output b.

We claim that indistinguishability of $\mathsf{Expt}^{\mathsf{Real}}_{\Pi_{\mathsf{TFHE}},\mathcal{A}}$ and $\mathsf{Expt}^{\mathsf{Hyb}}_{\Pi_{\mathsf{TFHE}},\mathcal{A}}$ follows directly by privacy of universal thresholdizer. Indeed, observe that the difference between $\mathsf{TFHE.Setup}$ and $\mathsf{TFHE.SimSetup}$ is in the call of $\mathsf{UT.Setup}$, where$\mathsf{TFHE.Setup}$ uses actual FHE secret key sk and $\mathsf{TFHE.SimSetup}$ uses $0^{|\mathsf{sk}|}$. Also in Step 6, partial decryptions $\mathsf{TFHE.PartDec}$ of $\hat{\mathsf{ct}}_j$ for $j \in S$, which in turn are the partial evaluations $\mathsf{UT.Eval}$ on the FHE decryption circuit $\mathsf{FHE.Dec}(\cdot, \mathsf{sk})$, is replaced by the simulator of the universal thresholdizer $\mathsf{UT.Sim}$, where $\mathcal{O}_{\mathsf{Sim}}$ is emulated correctly by returning $\hat{\mu}_j = C_j(x_1, \ldots, x_m)$ when queried by C_j for $j \in S$. Hence, $\mathsf{Expt}^{\mathsf{Real}}_{\Pi_{\mathsf{TFHE}},\mathcal{A}}$ and $\mathsf{Expt}^{\mathsf{Hyb}}_{\Pi_{\mathsf{TFHE}},\mathcal{A}}$ correspond to $\mathsf{Expt}^{\mathsf{Real}}_{\Pi_{\mathsf{UT}},\mathcal{A}}$ and $\mathsf{Expt}^{\mathsf{Rand}}_{\Pi_{\mathsf{UT}},\mathcal{A}}$ for universal thresholdizer, and the indistinguishability follows by privacy of universal thresholdizer.

Now, observe that the difference between $\mathsf{Expt}^{\mathsf{Hyb}}_{\Pi_{\mathsf{TFHE}},\mathcal{A}}$ and $\mathsf{Expt}^{\mathsf{Ideal}}_{\Pi_{\mathsf{TFHE}},\mathcal{A}}$ is only the messages encrypted in Step 3 and that the FHE secret key is not used in the experiments. Hence, indistinguishability of $\mathsf{Expt}^{\mathsf{Hyb}}_{\Pi_{\mathsf{TFHE}},\mathcal{A}}$ and $\mathsf{Expt}^{\mathsf{Ideal}}_{\Pi_{\mathsf{TFHE}},\mathcal{A}}$ follows directly by semantic security of FHE. This completes the proof of security for the constructed TFHE scheme.

A.3 Semi-Malicious Secure SMPC in a Trusted Setup Model

We proceed to construct a semi-malicious secure constant-round, weakly space efficient SMPC in a trusted setup model using threshold FHE. We consider SMPC protocols over m machines. Henceforth let β denote the bit-width of each word. Each machine holds input $x_i \in \{0,1\}^{\beta s}$ and wishes to learn $f_i(x_1, \ldots, x_m)$ for functions $f_i : \{0,1\}^{m \cdot \beta \cdot s} \to \{0,1\}$.[12] The construction is rather straightforward: We use the setup to run the threshold FHE setup algorithm $\mathsf{TFHE.Setup}$ and distribute the keys. Upon receiving the keys, each machine encrypts its input and output the ciphertext. Then they locally evaluate the output ciphertexts homomorphically, partially decrypt them, and send the partial decryptions to the corresponding machines, who can then learn their own outputs.

Formally, let $\Pi_{\mathsf{TFHE}} = (\mathsf{TFHE.Setup}, \mathsf{TFHE.SimSetup}, \mathsf{TFHE.Enc}, \mathsf{TFHE.Eval}, \mathsf{TFHE.PartDec}, \mathsf{TFHE.FinDec})$ be a threshold FHE scheme. We construct the following SMPC protocol Π_{SMPC} over m machines. Let t be an upper bound on the number of corrupted machines, where $m \geq 3t + 1$.

- **Input and functionality**:
  - Each machine M_i has input $x_i \in \{0,1\}^{\beta s}$ and wishes to learn $f_i(x_1, \ldots, x_m)$ for functions $f_i : \{0,1\}^{m \cdot \beta \cdot s} \to \{0,1\}$.
- **Setup Stage**:
  - Run $(\mathsf{tpk}, \{\mathsf{tsk}_i\}_{i \in [m]}) \leftarrow \mathsf{TFHE.Setup}(1^\kappa, 1^m, 1^t)$.
  - Send $(\mathsf{tpk}, \mathsf{tsk}_i)$ to each machine M_i for $i \in [m]$.
- **Round** 1: Each machine M_i does the following:
  - Run $\mathsf{ct}_i \leftarrow \mathsf{TFHE.Enc}(\mathsf{tpk}, x_i)$.
  - Broadcast ct_i.
- **Round** 2: Each machine M_i does the following:
  - Record each ct_j received from machine M_j. If M_j aborts, then use $\mathsf{ct}_j \leftarrow \mathsf{TFHE.Enc}(\mathsf{tpk}, 0^{\beta s}; 0^*)$ as a default ciphertext.
  - Compute $\hat{\mathsf{ct}}_j \leftarrow \mathsf{TFHE.Eval}(\mathsf{tpk}, C_j, \{\mathsf{ct}_i\}_{i \in [m]})$ for $j \in [m]$.
  - Compute $p_{j,i} \leftarrow \mathsf{TFHE.PartDec}(i, \mathsf{tpk}, \mathsf{tsk}_i, \hat{\mathsf{ct}}_j)$ for $j \in [m]$.
  - Send $p_{j,i}$ to machine M_j for each $j \in [m]$.

[12] For notational simplicity, we consider functions with one-bit output. It is straightforward to extend the protocol to handle long outputs.

- **Output Computation**: Each machine M_i does the following:
  - Record each $p_{i,j}$ received from machine M_j. Let S_i be the set of partial decryptions received by M_i.
  - Compute $\hat{\mu}_i \leftarrow \mathsf{TFHE.FinDec}(\mathsf{tpk}, \{p_{i,j}\}_{j \in S_i})$.
  - Output $\hat{\mu}_i$.

Correctness

Correctness follows directly from the correctness of TFHE. Furthermore, since the number of honest machines is greater than $t+1$, the honest machines always learn the output.

Weak Space Efficiency

We first note that by compactness, except for the homomorphic evaluation TFHE.Eval, the remaining step takes space at most $\mathsf{poly}(\kappa) \cdot O(ms)$. Let S_{RAM} denote the space complexity for computing $f_1, \ldots, f_m$ using a RAM machine. It is not hard to see that we can emulate the RAM computation by a uniform layered circuit of width $O(S_{\mathsf{RAM}})$: for example, a naïve way is to emulate CPU and memory by circuits and for each RAM computation step, emulate CPU accessing memory by constructing a linear-sized circuit gadget that goes over every memory cell to selects the position requested. Thus, it follows by compactness that TFHE.Eval can be done in space $\mathsf{poly}(\kappa) \cdot O(S_{\mathsf{RAM}})$.

Semi-malicious Security

The security follows directly from the simulation security of TFHE. For completeness, we formally define the simulator $\mathcal{S}$ for Π_{SMPC} as follows. Let Honest and Crupt denote the set of honest and corrupted machines, respectively.

- **Setup Stage**: $\mathcal{S}$ simulates the setup by running TFHE.SimSetup instead of TFHE.Setup. Namely,
  - Run $(\mathsf{tpk}, \{\mathsf{tsk}_i\}_{i \in [m]}) \leftarrow \mathsf{TFHE.SimSetup}(1^\kappa, 1^m, 1^t)$.
  - Send $\{(\mathsf{tpk}, \mathsf{tsk}_i)\}_{i \in \mathsf{Crupt}}$ to the adversary $\mathcal{A}$.
- **Round 1**:
  - $\mathcal{S}$ simulates the honest machines' messages by running Sim_1 of TFHE. Namely, $\mathcal{S}$ runs $(\{\mathsf{ct}_i\}_{i \in \mathsf{Honest}}, \mathsf{st}_1) \leftarrow \mathsf{Sim}_1(\mathsf{tpk}, \{\mathsf{tsk}_i\}_{i \in \mathsf{Crupt}})$. $\mathcal{S}$ sends $\{\mathsf{ct}_i\}_{i \in \mathsf{Honest}}$ to $\mathcal{A}$.
  - Upon receiving $(\{(x_i, r_i^{\mathsf{Enc}})\}_{i \in \mathsf{Crupt}}$ from $\mathcal{A}$, $\mathcal{S}$ sends it to the ideal functionality. If any corrupted machine i aborts, then $\mathcal{S}$ sends $x_i = 0^{\beta s}$ to the ideal functionality.
  - $\mathcal{S}$ receives the outputs $\{\hat{\mu}_i\}_{i \in \mathsf{Crupt}}$ from the ideal functionality.
- **Round 2**:
  - $\mathcal{S}$ simulates the honest machines' messages by running Sim_2 of TFHE. Namely, $\mathcal{S}$ runs $\{p_{j,i}\}_{i \in \mathsf{Honest}, j \in \mathsf{Crupt}} \leftarrow \mathsf{Sim}_2(\mathsf{st}_1, \{(x_j, r_j^{\mathsf{Enc}})\}_{j \in \mathsf{Crupt}}, \{\hat{\mu}_j\}_{j \in \mathsf{Crupt}})$. $\mathcal{S}$ sends $\{p_{j,i}\}_{i \in \mathsf{Honest}, j \in \mathsf{Crupt}}$ to $\mathcal{A}$.
  - $\mathcal{S}$ sends `deliver` to the ideal functionality.

It is not hard to see that the real world and ideal world execution of Π_{SMPC} directly corresponds to the experiments $\mathsf{Expt}^{\mathsf{Real}}_{\Pi_{\mathsf{TFHE}}, \mathcal{A}}$ and $\mathsf{Expt}^{\mathsf{Ideal}}_{\Pi_{\mathsf{TFHE}}, \mathcal{A}}$ in the simulation security of TFHE with corresponding adversary. Hence, indistinguishability of the simulation for Π_{SMPC} follows by the simulation security of TFHE.

A.4 Achieving Malicious Security and Removing the Trusted Setup

Finally, we briefly discuss how to upgrade to malicious security and remove the trusted setup in Π_{SMPC}. To upgrade to malicious security, we can apply the standard generic transformation using a simulation-extractable multi-string NIZK which can be constructed from enhanced trapdoor permutations without extra setup [14, 74, 75]. Note that NIZK is used to prove that $\mathsf{TFHE.Enc}$ and $\mathsf{TFHE.PartDec}$ are done correctly, both statements have a fixed $\mathsf{poly}(\kappa)$ complexity. The NIZK proofs can be generated in a fixed $\mathsf{poly}(\kappa)$ space as well. Thus, the transformation preserves weak space efficiency.

To remove the setup, we rely on an SMPC protocol by Badrinarayanan et al. [14]. The protocol of [14] (Theorem 10) is constant-round, achieves guaranteed output delivery, and does not require any setup. We remove the trusted setup by invoking the protocol of Badrinarayanan et al. [14] to securely realize the setup stage in Π_{SMPC}. Note that $\mathsf{TFHE.Setup}$ has a fixed $\mathsf{poly}(\kappa)$ complexity (independent of the functionalities $f_1, \ldots, f_m$). As a result, we obtain a malicious security, constant-round, weakly space efficient and communication efficient SMPC protocol, as required in Theorem 22.

B Potential Barriers Towards Achieving Statistical Security

We have shown how to compile an MPC protocol to a secure counterpart that defends against slightly less than 1/3 corruption while preserving its efficiency; but our compiler relied on a few computational assumptions such as enhanced trapdoor permutations, LWE, and an appropriate notion of compact FHE. One intriguing question is whether we can accomplish the same, but *unconditionally*, i.e., without making any cryptographic hardness assumptions. Such protocols are also said to be *statistically* secure. We now show that if one could achieve the same result unconditionally, it will imply solutions to long-standing open questions in cryptography. Specifically, we prove the following theorem:

▶ **Theorem 27.** *Let κ denote the security parameter. Suppose that there exists an compiler that compiles any MPC protocol Π computing the function f among m machines into an SMPC-for-MPC protocol Π' among m machines that securely realizes $\mathcal{F}^f$ unconditionally, as long as m is polynomially bounded by s and $s \geq \kappa$. Furthermore, suppose that the compiler incurs only $O(1)$ blowup in round complexity and $\mathsf{poly}(\kappa)$ blowup in terms of per-machine space complexity.*

Then, for $m \geq \kappa$, for any m-input, m-output uniform layered circuit C with width m, as long as C's size is a sufficiently large polynomial in m (related to the parameter α later), then there exists a constant-round protocol that allows m parties to securely realize $\mathcal{F}^C$ unconditionally, incurring total communication that is $|C|^\alpha$ for an arbitrarily small constant $\alpha \in (0, 1)$.

Proof. If such a compiler existed, we can use it to compile the following insecure 1-round MPC protocol among m machines each of which has $s = O(m)$ space. Every one now sends their input to the first machine, the first machine computes the circuit C locally, and sends to each machine their respective output. Note that since the circuit is uniform and layered with maximum width m, the first machine can evaluate it in total space $O(m)$.

Now consider the compiled protocol: it will complete in $O(1)$ rounds, and moreover the total communication must be upper bounded by the number of rounds multiplied by $m \cdot O(s) \cdot \mathsf{poly}(\kappa) = O(m^2)$, which can be made $|C|^\alpha$ as long as the circuit size is a sufficiently large polynomial in m where $|C|$ denotes the size of C. ◀

As noted in numerous works in the cryptography literature [26, 39, 44, 45], the existence of such constant-round, sublinear-communication multi-party computation (for circuits) with *statistical* security has been a long-standing open problem, even for the special class of circuits that we consider. To the best of our knowledge, the best known n-party, statistically secure computation protocol achieves the following[13]

- $O(n|C|)$ total communication and d number of rounds without preprocessing [43, 73] where d denotes the circuit depth, and
- $O(n|C|/\log\log|C|)$ total communication and $d/\log\log|C|$ number of rounds with (polynomially-bounded) preprocessing (for layered circuits) [39].

Interestingly, we note that barring strong assumptions such as Indistinguishable Obfuscation [53], the only known approach to construct constant-round, sublinear-communication *multi*-party computation for circuits of unbounded polynomial size is also through compact FHE [55, 58]. In our earlier sections, we essentially showed that making a similar assumption, combined with other standard cryptographic assumptions, we can construct an efficiency-preserving "MPC to SMPC-for-MPC" compiler. From a technical perspective, the main new challenge we encountered is the fact that the machines are now also *space-constrained* (which was not a concern in the standard multi-party computation for circuit literature), and thus we could not just apply existing techniques to the entire set of machines.

Besides the feasibility of achieving statistical security, another interesting direction is to weaken the cryptographic assumption necessary in achieving such a compiler. Similarly, new results in this vein would imply new breakthroughs for constant-round, sublinear-communication *multi*-party computation for circuits too – and even partial results for special classes of circuits (like the family we considered in Theorem 27) would be interesting.

C Removing the Sender Constraint

As mentioned earlier, some works in the MPC literature do not seem to respect the s-sender-constraint, and only respect the s-receiver-constraint. In this section, we generalize our results even to such MPC algorithms.

To achieve this, it suffices to show that given an MPC protocol denoted Π that respects only the s-receiver-constraint, we can compile it to a counterpart denoted Π' that satisfies not just the $O(s)$-receiver-constraint, but also the $O(s)$-sender-constraint. Furthermore, the compilation should preserve both round- and space-complexity; and moreover, the compiled protocol Π' should run in a *fixed* number of rounds (since the oblivious Routing primitive applied to emulate the communication of Π' will not hide total round complexity).

Intuition

The idea is the following: in the first phase (called the replication phase), if a sender wants to send in total μ words in some communication round (where sending the same word to two machines is counted twice), it will *replicate* all of its local memory to $\lceil\mu/s\rceil$ helper-machines where each helper is assigned a unique index from 1 to $\lceil\mu/s\rceil$. This must be accomplished using an s-sender-bounded communication pattern. In the second phase (called the distribution phase), each helper distributes s words on behalf of the sender it represents. Note that in the same round, many machines may be trying to send data simultaneously.

[13] For this reason, in fact even if we relaxed the round complexity blowup in Theorem 27 to poly-logarithmically many rounds or any number that does not depend on circuit depth, having such an MPC-to-SMPC compiler would still imply improving the state-of-the-art for statistically secure MPC.

Thus the above procedure is performed in parallel among all senders. It must be guaranteed that *every machine serves as a helper at most for one sender*. In this way, the distribution phase will satisfy the s-sender-constraint.

C.1 Replication Protocol

A replication protocol allows senders to replicate their local memory to an appropriate number of helpers.

Definition

Formally, replication, henceforth denoted Replicate, is the following problem.

- **Input**. Suppose that among the m machines, some machines are senders and others are non-senders. Each machine i obtains an input pair (β_i, c_i) where $\beta_i \in \{0, 1\}$ is a bit indicating whether machine i is a sender; and if $\beta_i = 1$, $c_i \geq 1$ denotes the total number of machines to replicate machine i's state – henceforth we refer to c_i as sender i's *multiplicity*.

  It is guaranteed that $\sum_{i=1}^{m} \beta_i c_i \leq m$, i.e., in total there are enough machines around to act as receivers.
- **Output**. At the end of the replication protocol, the following output configuration is produced:
  - each sender i has its entire machine state (i.e., a total of s words) replicated on exactly c_i receivers;
  - each machine acts as a receiver for at most 1 sender;
  - suppose sender i has c_i receivers, each of these receivers output i and also a unique index j from the range $[c_i]$, i.e., each of these receiver knows that it acts as the j-th receiver for its sender.

We will next construct a Replicate protocol. Note that the protocol need not be communication-oblivious. The idea is that all machines will first perform a prefix sum computation which allows each sender to discover a range of indices which are meant to become its helpers; moreover, all senders' helpers, identified by the range, are disjoint. It is well-known that prefix sum can be accomplished on MPC in $O(1)$ rounds with an s-sender-bounded communication style. Now, we employ a RangeCast protocol for the sender to replicate its state to the range of machines discovered above. Below we first explain how to construct the RangeCast building block and then describe our Replicate protocol.

Building block: RangeCast

As mentioned, RangeCast allows a sender to replicate its state to a set of machines defined by a range $[a, b] \subseteq [m]$. To realize such a RangeCast primitive, we first realize a weaker form denoted WeakRangeCast which only works if the range's size is at most s. Our RangeCast is similar to the "broadcast" algorithm in the "bulk-synchronous parallel (BSP)" model [103], but we describe it again for completeness.

WeakRangeCast

Input: let a, b be any two machines such that $1 \leq a \leq b \leq m$, let I be an array of s words. The machine a receives the input (I, a, b) where *the range* $[a, b]$ *is small enough such that* $b - a + 1 \leq s$.

Output: every machine $k \in [a, b]$ outputs I.

Protocol:

1. Let $c := b - a + 1$ and $t := \lceil s/c \rceil$. For each $j \in [c]$, let $I_j := I[(j-1)t + 1 : jt]$ be the substring of I where I_j consists of at most t words (if jt or $(j-1)t+1$ is less than s, I_j is by definition a shorter or empty string). In this round, the machine a sends to the machine $a + j - 1$ the message tuple (a, b, I_j) for each $j \in [c]$, while all other machines send nothing.
2. In the next round, machine $a + j - 1$ receives (a, b, I_j) for each $j \in [c]$, and it sends the same message (j, I_j) to every machine $k \in [a, b]$.
3. Every machine $k \in [a, b]$ receives a copy of (j, I_j) for all $j \in [c]$, and it recovers I by concatenating $(I_j)_{j \in [c]}$.

▶ **Lemma 28.** *For any* $1 \leq a \leq b \leq m$ *such that* $b - a + 1 \leq s$, *WeakRangeCast correctly implements range-cast in 2 rounds and satisfies both* $O(s)$*-sender- and* $O(s)$*-receiver-constraints, where each machine takes* $O(s)$ *space and time locally.*

Proof. The correctness, rounds, and local-machine complexity follows directly. The $O(s)$-sender-constraint holds because in Step 1, machine a sends to each machine $O(s/c+2)$ words, and thus the total number of words sent is $O(s + c) = O(s)$. The $O(s)$-receiver-constraint holds because in Step 3, each machine receives c messages each consists of $O(s/c + 1)$ words. ◀

Given WeakRangeCast as a building block, we can construct RangeCast where the range $[a, b]$ may be arbitrary in size. The protocol basically builds a distribution tree of fanout s such that the sender first distributes to s machines, then the s machines distribute to upto s^2 machines, and so on. The protocol can be described formally below.

RangeCast

Input and output: Same as WeakRangeCast but without any constraint on the range $[a, b]$.

Protocol:

1. (Base case.) Let $c := b - a + 1$. If $c \leq 1$, then there is only one machine a and it outputs I directly. Otherwise $c > 1$, continue with the following steps.
2. Let $r := \min(c, s)$. Run WeakRangeCast on the first r machines (i.e., in the range $[a, a + r - 1]$) to copy I from machine a to machines $[a, a + r - 1]$.
3. The machine a computes a partition $[a_1, b_1], [a_2, b_2], \ldots, [a_r, b_r]$ of the range $[a, b]$ such that the ranges $[a_i, b_i]$ are as even as possible (i.e., for any $i_1, i_2 \in [r]$, it holds that $|(b_{i_1} - a_{i_1}) - (b_{i_2} - a_{i_2})| \leq 1$). The pair (a_i, b_i) is sent from machine a to both machines $a + i - 1$ and a_i; Afterwards, machine $a + i - 1$ sends the received I to machine a_i for each $i \in [r]$.
4. For each $i \in [r]$, the machine a_i performs recursively RangeCast on the received I and the range $[a_i, b_i]$.

▶ **Lemma 29.** *For any $1 \leq a \leq b \leq m$, RangeCast correctly implements range-cast in $O(\frac{\log m}{\log s})$ rounds and satisfies both $O(s)$-sender- and $O(s)$-receiver-constraints, where each machine takes $O(s)$ space and time locally.*

Proof. In Step 3, if $c < s$, then $r = c$ and all c machines receive I in $O(1)$ rounds. Else, the first s machines receive I and then forward I to groups of size $\lceil c/s \rceil$, and it takes at most $O(\log_s m) = O(\frac{\log m}{\log s})$ iterations to divide any problem of size $c \leq m$ to a constant size. The correctness follows directly. The local-machine complexity, $O(s)$-sender and $O(s)$-receiver constraints follow by WeakRangeCast. ◀

Protocol Replicate

We are now ready to describe the Replicate protocol. We will use the following terminology: we use the term *ball* to refer to a machine's entire state consisting of upto s words. Each ball is always tagged the ball's identifier denoted $\mathsf{id} \in [m]$, i.e., which machine's state it represents.

The building block PrefixSum is the following primitive: every machine starts with a number, and machine i would like to learn the sum of machine 1 to machine i's numbers. As described by Goodrich et al. [70], this can be accomplished in $O(1)$ rounds with an s-sender-bounded MPC protocol, and consuming $O(m)$ total communication.

1. Each ball is additionally tagged with its outputting range computed as follows.
   a. Given as input (β_i, c_i), each machine $i \in [m]$ sets $c_i = 0$ iff $\beta_i = 0$. All m machines jointly run the PrefixSum protocol on $(c_i)_{i \in [m]}$. As the result, each machine i gets the prefix sum $p_i = \sum_{j \in [i]} c_j$.
   b. Let $p_0 = 0$. Every machine i calculates $p_{i-1} = p_i - c_i$ locally. For each machine i such that $c_i \geq 1$ (i.e., the range $[p_{i-1} + 1, p_i]$ is non-empty), machine i tags the range $[p_{i-1} + 1, p_i]$ to its ball and then sends the ball to machine $p_{i-1} + 1$.
2. Now each ball i is received by the first machine in its outputting range $[p_{i-1} + 1, p_i]$. To replicate the ball i to all machines in the range, for each ball i and the tagged range $[p_{i-1} + 1, p_i]$, the machine $p_{i-1} + 1$ performs RangeCast on the ball i and the range of machines $[p_{i-1} + 1, p_i]$. This RangeCast is performed simultaneously for all balls and hence all machines. To ensure all machines finish at the same round, every instance of RangeCast is programmed to finish at the $O(\frac{\log m}{\log s})$-th round specified in Lemma 29.
3. For each machine i such that has a ball tagged with a range $[a, b]$, let $j := i - a + 1$. Output the ball and j.

▶ **Lemma 30.** *The MPC protocol Replicate is a correct replicate protocol such that takes $O(1)$ rounds and $O(m \cdot s)$ communication, satisfies $O(s)$-sender- and $O(s)$-receiver-constraints, and each machine locally takes $O(s)$ time and $O(s)$ space.*

Proof. The correctness holds as each ball i is replicated exactly c_i copies and the ranges $[p_{i-1} + 1, p_i]$ are disjoint. The complexities follow directly by PrefixSum [70] and RangeCast (Lemma 29). ◀

C.2 Sender-Bounded Compiler

We will now compile any MPC protocol that respects only the s-receiver-constraint to one that respects both the s-receiver- and s-sender-constraints.

Without loss of generality, we may assume that at the end of the local computation stage of each round, there is a *deterministic* polynomial time algorithm[14] that takes each machine's local state as input, and can write down sequentially in a stream a set of *send instructions* where each send instruction contains an outgoing word to be sent and the destination machine's identifier. Note that the sender may not have space to write down all these instructions since each word sent multiple times need to be duplicated multiple times, taking more than $O(s)$ space. However, if the sender replicates its state to enough helpers, every helper can locally repeat the same computation, and write down the range of at most $O(s)$ instructions it is responsible for implementing.

Sender-bounded compiler

Every communication round of the original MPC is replaced with the following protocol:

1. Invoke an instance of the non-oblivious Replicate algorithm: if machine i is trying to send in total μ_i words[15], it replicates its local state onto $\lceil \mu_i/s \rceil$ machines. At the end of this phase, every machine i' may receive the local state of at most one machine i and if so, it also learns that it will act as the j-th helper for machine i.
2. If a machine i' is the j-th helper for machine i, it uses machine i's state to compute the $((j-1)s+1)$-th send instruction through the $\min(j \cdot s, \mu_i)$-th send instruction.
3. Now, every machines executes all send instructions written down in the previous step.

Note that this sender-bounded compiler may not compile each round of the original MPC to a fixed number of rounds. To obtain a compiler that always emits a protocol with a fixed number of rounds, we can simply pad the resulting protocol to the worst-case number of rounds: if there is no more work to be done, just execute empty rounds that do nothing.

▶ **Theorem 31** (Sender-bounded compiler with a fixed number of rounds)**.** *Assume that $s = N^{\varepsilon}$ and m is upper bounded by a fixed polynomial in N. Given any m-machine MPC protocol Π that completes in R rounds in the worst case and consuming s per-machine space, there is an MPC protocol Π' that computes the same function as Π, consuming $O(R)$ rounds, $O(s)$ per-machine space, and $O(m \cdot s)$ total communication per round, and moreover Π' additionally satisfies the $O(s)$-sender-constraint, and executes for a fixed number of rounds. Note that for well-formedness, both Π and Π' must satisfy the s-receiver-constraint.*

Proof. By Lemma 30, the above compiler replaces each round of the original MPC with $O(1)$ rounds of communication, and moreover the resulting protocol satisfies $O(s)$-sender- and $O(s)$-receiver-constraints. The fixed total rounds is guaranteed due the padding mechanism mentioned above. ◀

D Additional Preliminary: Robust Secret Sharing

We recall the notion of robust secret sharing schemes [88]. Here, we only consider robust secret sharing schemes with threshold $t < m/3$.

[14] In the case that the algorithm for generating send instructions is randomized, we may assume that the randomness is pseudo-randomly generated with a small seed using a cryptographically secure pseudo-random generator. This way, the MPC's outputs are computationally indistinguishable no matter whether true randomness or pseudorandomness is used in determining the send instructions.

[15] This can be determined with polynomial-time computation based on its local state.

▶ **Definition 32** (Robust Secret Sharing). *A* t-out-of-m robust secret sharing scheme *over a message space* $\mathcal{M}$ *and share space* $\mathcal{S}$ *is a tuple* $(\mathsf{Share}, \mathsf{Recons})$ *of algorithms defined as follows:*

- $\mathsf{Share}(\mathsf{msg}) \to (s_1, \ldots, s_m)$*: This is a randomized algorithm that takes as input a message* $\mathsf{msg} \in \mathcal{M}$ *and output a sequence of shares* $s_1, \ldots, s_m \in \mathcal{S}$*.*
- $\mathsf{Recons}(s_1, \ldots, s_m) \to \mathsf{msg}'$*: This is a deterministic algorithm that takes as input* m *shares* $(s_1, \ldots, s_m)$ *with* $s_i \in \mathcal{S} \cup \{\perp\}$ *and outputs a message* $\mathsf{msg}' \in \mathcal{M}$*.*

We require the following properties.

- ***Perfect Privacy****: Any* t *out of* m *shares of a secret give no information on the secret itself. Namely, for any* $\mathsf{msg}, \mathsf{msg}' \in \mathcal{M}$ *and* $S \subset [m]$ *of size* $|S| = t$*, the distributions of* $\mathsf{Share}(\mathsf{msg})_S$ *and* $\mathsf{Share}(\mathsf{msg}')_S$ *are identical. Here,* $\mathsf{Share}(\mathsf{msg})_S$ *denotes the set of shares* $\{s_i\}_{i \in S}$ *generated by* $\mathsf{Share}(\mathsf{msg})$*.*
- ***Robustness****: An adversary modifies up to* t *shares can cause the wrong secret to be recovered with probability at most* δ*. Specifically, for any* $\mathsf{msg} \in \mathcal{M}$*,* $S \subset [m]$ *of size* $|S| = t$ *and (unbounded) adversary* $\mathcal{A}$*,*

$$\Pr[\mathsf{Recons}(\mathsf{Share}(\mathsf{msg})_{[m]\setminus S}, \mathcal{A}(\mathsf{Share}(\mathsf{msg})_S)) \neq \mathsf{msg}] \leq \delta.$$

It is known that Shamir's secret sharing scheme is an efficient t-out-of-m robust secret sharing scheme for $t < m/3$.

Choice and Bias in Random Walks

Agelos Georgakopoulos
Mathematics Institute, University of Warwick, UK

John Haslegrave
Mathematics Institute, University of Warwick, UK
J.Haslegrave@warwick.ac.uk

Thomas Sauerwald
Department of Computer Science & Technology, University of Cambridge, UK
Thomas.Sauerwald@cl.cam.ac.uk

John Sylvester
Department of Computer Science & Technology, University of Cambridge, UK
John.Sylvester@cl.cam.ac.uk

Abstract

We analyse the following random walk process inspired by the power-of-two-choice paradigm: starting from a given vertex, at each step, unlike the simple random walk (SRW) that always moves to a randomly chosen neighbour, we have the choice between *two* uniformly and independently chosen neighbours. We call this process the *choice random walk* (CRW).

We first prove that for any graph, there is a strategy for the CRW that visits any given vertex in expected time $\mathcal{O}(|E|)$. Then we introduce a general tool that quantifies by how much the probability of a rare event in the simple random walk can be boosted under a suitable CRW strategy. We believe this result to be of independent interest, and apply it here to derive an almost optimal $\mathcal{O}(n \log\log n)$ bound for the cover time of bounded-degree expanders. This tool also applies to so-called *biased walks*, and allows us to make progress towards a conjecture of Azar et al. [STOC 1992]. Finally, we prove the following dichotomy: computing an optimal strategy to minimise the hitting time of a vertex takes polynomial time, whereas computing one to minimise the cover time is NP-hard.

2012 ACM Subject Classification Theory of computation → Random walks and Markov chains; Mathematics of computing → Stochastic processes

Keywords and phrases Power of Two Choices, Markov Chains, Random Walks, Cover Time, Markov Decision Processes

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.76

Related Version This is an extended abstract of [23, 26]. `https://arxiv.org/abs/1911.05170`.

Funding A. G. & J. H. were supported by ERC Starting Grant no. 639046 (RGGC). T. S. & J. S. were supported by ERC Starting Grant no. 679660 (DYNAMIC MARCH).

1 Introduction

Motivation and Related Work. The power of choice paradigm is the phenomenon that when a random process is offered a choice between two or more uniformly selected options, as opposed to just being supplied with a uniformly random one, then a series of choices can be made to improve overall performance [32]. The power of two choices was first considered for balanced allocation of balls to bins [7, 11, 31]. Here the surprising discovery was made that if each ball is offered two randomly selected bins and the bin containing fewer balls is chosen then the maximum load when assigning n balls to n bins decreases significantly from $\Theta\left(\frac{\log n}{\log\log n}\right)$ to $\Theta(\log\log n)$. The power of choice was later studied for random graphs under the broader class of rule-based random graph processes known as Achlioptas processes. In the standard random graph process, a graph on a fixed vertex set is built up by adding

random edges one by one. Achlioptas suggested that if instead an edge to add is chosen from two random options, this may be done in such a way as to shift the position of the critical window in which a giant component emerges. This is indeed the case and now much is known about the effect of various rules on the phase transition [12, 13, 2, 36, 35]. The effect of the power of choice on the degree distribution in the Preferential Attachment process has also been studied [30, 25].

In this paper we apply the power of two choices to a random walk on a graph with the hope of speeding up the cover and hitting times. One motivation behind this is to improve the efficiency of random walks used in algorithmic applications such as searching, routing, self-stabilization, and query processing in wireless networks, peer-to-peer networks and other distributed systems. One practical setting where routing using the power of choice walk may be advantageous is in relatively slowly evolving dynamic networks such as the internet. For example, say a packet has a target destination v and each node stores a pointer to a neighbour which it believes leads most directly to v. If this network is perturbed then the deterministic scheme may get stuck in "dead ends" whereas a random walk would avoid this fate. The choice random walk which prefers edges pointed to by a node may be the best of both worlds as it would also avoid traps but may see a speed up over the simple random walk when the original paths are still largely intact.

To the best of our knowledge, Avin and Krishnamachari [5] were the first to apply the principle of power of choices to random walks. However, their version only considers a simple choice rule where the vertex with fewer previous visits is always preferred, and ties are broken randomly.

Their results are mainly empirical and suggest a decrease in the variance of the cover time, and a significant improvement in visit load balancing. This is related to the greedy random walk of Orenshtein and Shinkar [34], which chooses uniformly from adjacent vertices that have not yet been visited (if possible). This model is well studied for expanders [10, 15].

Alon, Benjamini, Lubetzky and Sodin [4] studied the mixing rate and asymptotic number of visits made to vertices by the non-backtracking walk. These authors mention the power of two choices paradigm and ask if the number of visits to any vertex can be further reduced by choosing between two independent non-backtracking walks at each step. Fitzner and van der Hofstad [21] obtained more mixing time results and Bordenave, Lelarge and Massoulié have also studied this process in relation to community detection [14].

Perhaps closest to our work, Azar, Broder, Karlin, Linial and Phillips [6] introduced the ε-bias random walk where at each step with probability ε a controller can choose a neighbour of the current vertex, otherwise one is uniformly selected. They obtained bounds on the stationary probabilities and show that optimal strategies for max/minimising stationary probabilities or hitting times can be computed in polynomial time (cf. Section 4.3).

Other related strategies for speeding up the hitting and cover times include degree-biased random walk models [28, 1, 17] or performing multiple walks in parallel [3, 19, 20]. The Power of two choices concept has also been studied in the context of deterministic variants of random walks [16, 8].

Our Results and Techniques. Our first result is a general upper bound of $O(|E|) = O(n^2)$ on the maximum hitting time of a vertex (Theorem 3). This is tight and improves considerably over the well-known $O(n|E|)$ worst-case bound for the simple random walk. This is achieved by approximating the (not necessarily reversible) CRW by a suitable reversible walk.

In Section 4, we present bounds on hitting and cover times in terms of the spectral gap of the lazy random walk. Our general cover time result (Theorem 4) constitutes a significant improvement over the corresponding best possible bound for the simple random walk [33].

In particular, it implies an almost-optimal $O(n \log \log n)$ bound for any bounded-degree expander. The same result also holds for a natural variant of the biased random walk of [6]. Our hitting time result (Theorem 5) is quite different from Theorem 3 and shows that for a large class of graphs that the hitting times of the choice and biased random walk are sublinear. The main technical contribution to derive these bounds is Theorem 6, which shows that any rare event in the simple random walk case can be amplified substantially under a suitable choice (or time-biased) random walk strategy. If one thinks of a simple random walk as running a program with random bits as input then the "tree gadget" used to prove Theorem 6 is a novel way to quantify the effect of the non-determinism added by the power of two choices. We apply the results of this section to a conjecture of Azar et al. [6]; since our approach is orthogonal to theirs, we manage to confirm their conjecture for class of graphs different to those previously treated. As our amplification result applies to arbitrarily defined events and general stochastic processes, we believe this result may find further applications in other areas.

In Section 5, we investigate the complexity of computing optimal strategies for hitting times, cover times and maximising stationary probabilities. Our main insight is a surprising dichotomy, essentially saying that computing complete optimal strategies for hitting times is easy (i.e., polynomial-time), while computing a sequence of optimal choices for cover times, even in an on-line fashion, is NP-hard. To the best of our knowledge, this is the first negative result for processes involving random walks with choice.

2 Notation and Preliminaries

Throughout this paper all graphs will be finite and connected.

Choice Random Walk. The *Choice Random Walk* (CRW) is a discrete time stochastic process $(X_t)_{t \geqslant 0}$ on the vertices of a connected graph $G = (V, E)$, influenced by a controller. The starting state is a fixed vertex; at each time $t \in \mathbb{N}$ the controller is presented with two neighbours $\{c_1^t, c_2^t\}$ of the current state X_t chosen uniformly at random with replacement and must choose one of these neighbours as the next state X_{t+1}. We assume that at each time t the controller knows the graph G, its current position $X_t \in V$, and $\mathcal{H}_t = \left(X_i, \{c_1^i, c_2^i\}\right)_{i=0}^t$ the *history* of the process so far. The controller has access to arbitrary computational resources and an infinite string of random bits ω in order to choose X_{t+1} from $\{c_1^t, c_2^t\}$. A *strategy* for a given task on G is a function which given any t, $\mathcal{H}_t$ and ω outputs a single vertex from c^t where $c^t = \{c_1^t, c_2^t\} \subseteq \Gamma(X_t)$ (here, as is usual, we write $\Gamma(v) := \{w : vw \in E\}$ for the neighbourhood of v).

The aim of the CRW is to make a sequence of choices which most effectively complete a given objective. Examples of objectives may be as follows:

(1) to visit every vertex of the graph;

(2) to hit a given vertex or set of vertices;

(3) to maximise or minimise the stationary probability of a given vertex or set.

Efficacy in tasks (1) and (2) is determined by the expected number of steps taken. Note that an optimal solution to task (1) will necessarily make use of the history of the process, whereas task (3) only applies in the context of strategies which do not change over time. We say that a CRW strategy is *unchanging* if it is independent of both time and the history of the walk. As the walk has access to random bits the strategy may be randomised; we say a strategy is *deterministic* if random bits are not used to make a choice.

For a strategy α and for a vertex v and distinct neighbours i, j let $\alpha^j_{v,i}$ be the probability that when the walk is at v it chooses i when offered $\{i, j\}$ as choices, i.e.

$$\alpha^j_{v,i} := \mathbb{P}\left[\, X_{t+1} = i \mid X_t = v, c^t = \{i, j\} \,\right]$$

(this probability is also conditional on $\mathcal{H}_t$ but we suppress this for notational convenience). These are the only parameters we may vary, but we shall find it convenient to define $\alpha^i_{v,i} := 1/2$ for each i adjacent to v. Thus

$$\text{for each } v \in V: \ \alpha^j_{v,i} \in [0,1] \text{ and } \alpha^i_{v,j} = 1 - \alpha^j_{v,i} \text{ for all } i, j \in \Gamma(v). \tag{1}$$

The transition probabilities $q_{v,i}$ for the strategy α are then given by

$$q_{v,i} = \frac{2\sum_{j\in\Gamma(v)} \alpha^j_{v,i}}{d(v)^2}. \tag{2}$$

Note, any family of parameters $\alpha^j_{v,i}$ satisfying (1) gives a valid set of transition probabilities.

Let $C^{\mathsf{two}}_v(G)$ denote the minimum expected time (taken over all strategies) for the CRW to visit every vertex of G starting from v, and define the *cover time* $t^{\mathsf{two}}_{\mathsf{cov}}(G) := \max_{v\in V} C^{\mathsf{two}}_v(G)$. Analogously, let $H^{\mathsf{two}}_x(y)$ denote the minimum expected time for the CRW to reach y, which may be a single vertex or a set of vertices, starting from a vertex x and define the *hitting time* $t^{\mathsf{two}}_{\mathsf{hit}}(G) := \max_{x,y\in V} H^{\mathsf{two}}_x(y)$.

ε-Biased and ε-Time-Biased Random Walks. Azar et al. [6], building on earlier work [9], introduced the ε-biased random walk (ε-BRW) on a graph G. Each step of the ε-B walk is preceded by an $(\varepsilon, 1-\varepsilon)$-coin flip. With probability $1-\varepsilon$ a step of the simple random walk is performed, but with probability ε the controller gets to select which neighbour to move to. The selection can be probabilistic, but it is time independent. Thus if $\mathbf{P}$ is the transition matrix of a random walk, then the transition matrix $\mathbf{Q}^{\varepsilon\mathrm{B}}$ of the ε-biased random walk is given by

$$\mathbf{Q}^{\varepsilon\mathrm{B}} = (1-\varepsilon)\mathbf{P} + \varepsilon\mathbf{B}, \tag{3}$$

where $\mathbf{B}$ is an arbitrary stochastic matrix chosen by the controller, with support restricted to $E(G)$. In both the ε-Biased and Choice random walks the controller has full knowledge of G.

Azar et al. focused on bias strategies for maximising stationary probabilities and minimising or maximising hitting times of vertices or sets. For each of these tasks one may apply tools from Markov decision theory [18] to show there is a time-independent optimal strategy, so the definition above is sufficient for their purposes. For us a time-dependent version, where the bias matrix $\mathbf{B}_t$ may depend on the time t and the history of the process up to time t, will be useful; we refer to this as the ε-time-biased walk (ε-TBRW). We shall show that the ε-TBRW may be simulated, for suitable ε, by a CRW.

▶ **Proposition 1.** *For any graph G of maximum degree d_{max}, and for any $\varepsilon \leqslant 1/d_{\mathsf{max}}$, the CRW can simulate the ε-TBRW and ε-BRW on G.*

▶ Remark 2. The dependence of ε on d_{max} in Proposition 1 is tight. In the reverse direction, the ε-TB walk can only simulate the CRW if $\varepsilon > 1 - 1/d_{\mathsf{max}}$.

We write $t^{\varepsilon\mathsf{TB}}_{\mathsf{cov}}$ for the cover time of the ε-TBRW under an optimal strategy. There is always a time-independent optimal strategy for hitting a given vertex [6, Thm. 11], thus the maximum hitting times of the ε-TBRW and ε-BRW are the same; we use $t^{\varepsilon\mathsf{B}}_{\mathsf{hit}}$ to denote them. Any unchanging strategy on a finite connected graph results in an irreducible Markov chain and thus, when appropriate, we refer to its stationary distribution as π.

3 A Tight Upper Bound on the Hitting Time in General Graphs

Our first result is the following asymptotically tight bound on the maximal hitting time:

▶ **Theorem 3.** *For any graph G we have $t^{\mathsf{two}}_{\mathsf{hit}}(G) < 3e(G)$ and $t^{\mathsf{two}}_{\mathsf{hit}}(G) < n^2$.*

Both bounds are best possible up to the implied constants: for the path, $t^{\mathsf{two}}_{\mathsf{hit}}$ is about twice the number of edges, and for a clique with a pendant path, where the length of the path is growing much slower than the size of the clique, it is about $3n^2/8$.

We say that an unchanging strategy is *reversible* if it can be realised as a random walk on a weighted graph. The main idea used to prove Theorem 3 is that on any graph the CRW can implement a reversible strategy with a strong drift towards the target vertex. We can then employ tools from reversible Markov chains to bound the hitting time. See Appendix A.2 for a proof. While the reversible strategy constructed gives a bound on the optimal strategy, the latter need not be reversible; for an example, see Appendix A.2.

4 Hitting and Cover Times in Expanders

In this section we prove the following bounds on the cover and hitting times of the ε-TBRW and CRW on a graph G in terms of n, its extremal and average degrees $d_{\mathsf{max}}, d_{\mathsf{min}}$ and d_{avg}, and its relaxation time $t_{\mathsf{rel}} := \frac{1}{1-\lambda_2}$, where λ_2 is the second largest eigenvalue of the transition matrix of the lazy random walk (LRW) on G with loop probability $1/2$.

▶ **Theorem 4.** *For any graph G, and any $\varepsilon \in (0,1)$, we have*

$$t^{\mathsf{two}}_{\mathsf{cov}}(G) = \mathcal{O}\left(n \cdot d_{\mathsf{max}} \cdot \frac{d_{\mathsf{avg}}}{d_{\mathsf{min}}} \cdot \sqrt{t_{\mathsf{rel}}} \cdot \left(1 + \frac{\log t_{\mathsf{rel}}}{\log\log n}\right) \cdot \log\log n\right);$$

$$t^{\varepsilon\mathsf{TB}}_{\mathsf{cov}}(G) = \mathcal{O}\left(n \cdot \varepsilon^{-1} \cdot \frac{d_{\mathsf{avg}}}{d_{\mathsf{min}}} \cdot \sqrt{t_{\mathsf{rel}}} \cdot \left(1 + \frac{\log t_{\mathsf{rel}}}{\log\log n}\right) \cdot \log\log n\right).$$

In particular, the CRW cover time of a bounded degree (not necessarily regular) expander is $\mathcal{O}(n \log\log n)$, significantly less than that of the SRW, which is $\Theta(n \log n)$.

▶ **Theorem 5.** *For any graph G, and any $\varepsilon \in (0,1)$, we have*

$$t^{\mathsf{two}}_{\mathsf{hit}}(G) \leqslant 12\left(\frac{n \cdot d_{\mathsf{avg}}}{d_{\mathsf{min}}}\right)^{1-1/d_{\mathsf{max}}} \cdot t_{\mathsf{rel}} \cdot \ln n \quad \text{and} \quad t^{\varepsilon\mathsf{B}}_{\mathsf{hit}}(G) \leqslant 12\left(\frac{n \cdot d_{\mathsf{avg}}}{d_{\mathsf{min}}}\right)^{1-\varepsilon} \cdot t_{\mathsf{rel}} \cdot \ln n;$$

these bounds hold also for return times.

Theorems 4 and 5 will follow from Theorem 6. Let $G, t \geqslant 0$ and S be a set of trajectories of length t. In the following we use bold characters to denote trajectories in G and if $u \in V(G)$ then u will denote the length 0 trajectory from u. Let $p_{\mathbf{x},S}$ denote the probability that extending a trajectory $\mathbf{x}$ to length t according to the law of a SRW results in a member of S. Let $q_{\mathbf{x},S}(\varepsilon)$ and $\widetilde{q}_{\mathbf{x},S}$ denote the corresponding probabilities under the ε-TBRW or CRW laws respectively; the values of these probabilities will depend on the particular strategies used. These functions can encode probabilities of many events of interest such as "the graph is covered by time t", "the walk is in a set X at time t" or "the walk has hit a vertex x by time t" for example. However, let us emphasise that our result in fact applies to *any* possible event.

▶ **Theorem 6.** *Let G be a graph, $u \in V$, $t > 0$, $0 \leqslant \varepsilon \leqslant 1$ and S be a set of trajectories of length t from u. Then there exist strategies for the ε-TBRW and the CRW such that*

$$q_{u,S}(\varepsilon) \geqslant (p_{u,S})^{1-\varepsilon} \qquad \text{and} \qquad \widetilde{q}_{u,S} \geqslant (p_{u,S})^{1-1/d_{\mathsf{max}}}.$$

By Proposition 1, the results for the CRW in Theorems 4, 5 and 6 follow immediately from those for the ε-TBRW by taking $\varepsilon = 1/d_{\mathsf{max}}$. We shall therefore only consider the ε-TBRW. After stating a technical lemma in Section 4.1, we then explain an alternative way of considering the ε-TBRW in Section 4.2, which enables the proof of Theorem 6 to be completed. The proof of Theorems 4 and 5 via Theorem 6 is given in Appendix A.3.

4.1 The ε-Max/Average Operation

For $0 < \varepsilon < 1$ define the ε-max/average operator $\mathrm{MA}_\varepsilon : [0,\infty)^m \to [0,\infty)$ by

$$\mathrm{MA}_\varepsilon(x_1,\dots,x_m) = \varepsilon \cdot \max_{1\leqslant i\leqslant m} x_i + \frac{1-\varepsilon}{m}\cdot\sum_{i=1}^m x_i.$$

This can be seen as an average which is biased in favour of the largest element, indeed it is a convex combination between the largest element and the arithmetic mean.

For $p \in \mathbb{R}\setminus\{0\}$, the p-power mean M_p of non-negative reals $x_1,\dots,x_m$ is defined by

$$M_p(x_1,\dots,x_m) = \left(\frac{x_1^p+\cdots+x_m^p}{m}\right)^{1/p},$$

and

$$M_\infty(x_1,\dots,x_m) = \max\{x_1,\dots,x_m\} = \lim_{p\to\infty} M_p(x_1,\dots,x_m).$$

Thus we can express the ε-max/ave operator as $\mathrm{MA}_\varepsilon(\cdot) = (1-\varepsilon)M_1(\cdot) + \varepsilon\,\mathrm{M}_\infty(\cdot)$. We use a key lemma, Lemma 7, which could be be described as a multivariate anti-convexity inequality.

▶ **Lemma 7.** *Let $0 < \varepsilon < 1$, $m \geqslant 1$ and $\delta \leqslant \varepsilon/(1-\varepsilon)$. Then for any $x_1,\dots,x_m \in [0,\infty)$,*

$$M_{1+\delta}(x_1,\dots,x_m) \leqslant \mathrm{MA}_\varepsilon(x_1,\dots,x_m).$$

A proof of this Lemma may be found in [26].

▶ Remark 8. The dependence of δ on ε given in Lemma 7 is best possible. This can be seen by setting $x_1 = 0$ and $x_i = 1$ for $2 \leqslant i \leqslant m$, and letting m tend to ∞.

4.2 The Tree Gadget for Graphs

In this section we prove Theorem 6. To achieve this we introduce the *Tree Gadget* which encodes walks of length at most t from u in a rooted graph (G,u) by vertices of an arborescence $(\mathcal{T}_t,\mathbf{r})$, i.e. a tree with all edges oriented away from the root $\mathbf{r}$. Given (G,u) we represent each walk of length $i \leqslant t$ started from u in G as a node at distance i from the root $\mathbf{r}$ in the tree $\mathcal{T}_t$. The root $\mathbf{r}$ represents the walk of length 0 from u. There is an edge from $\mathbf{x}$ to $\mathbf{y}$ in $\mathcal{T}_t$ if $\mathbf{x}$ is obtained from $\mathbf{y}$ by deleting the final vertex.

Also for $\mathbf{x} \in V(\mathcal{T}_t)$ let $\Gamma^+(\mathbf{x}) = \{\mathbf{y} \in V(\mathcal{T}_t) : \mathbf{xy} \in E(\mathcal{T}_t)\}$ be the offspring of $\mathbf{x}$ in $\mathcal{T}$; as usual we write $d^+(\mathbf{x})$ for the number of offspring. Write $|\mathbf{x}|$ for the length of the walk $\mathbf{x}$. To prove Theorem 6 we shall need to discuss simple random walk paths; let $W_u(k) := \bigcup_{i=0}^k \{X_i\}$ be the trajectory of a simple random walk X_i on G up to time k, with $X_0 = u$.

Proof of Theorem 6. To each node $\mathbf{x}$ of the tree gadget $\mathcal{T}_t$ we assign the value $q_{\mathbf{x},S}$ under the the ε-TB strategy of biasing towards a neighbour in G which extends to a walk $\mathbf{y} \in \Gamma^+(\mathbf{x})$ maximising $q_{\mathbf{y},S}$. This is well defined because both the strategy and the values $q_{\mathbf{x},S}$ can be computed in a "bottom up" fashion starting at the leaves, where if $\mathbf{x} \in V(\mathcal{T}_t)$ is a leaf then $q_{\mathbf{x},S}$ is 1 if $\mathbf{x} \in S$ and 0 otherwise.

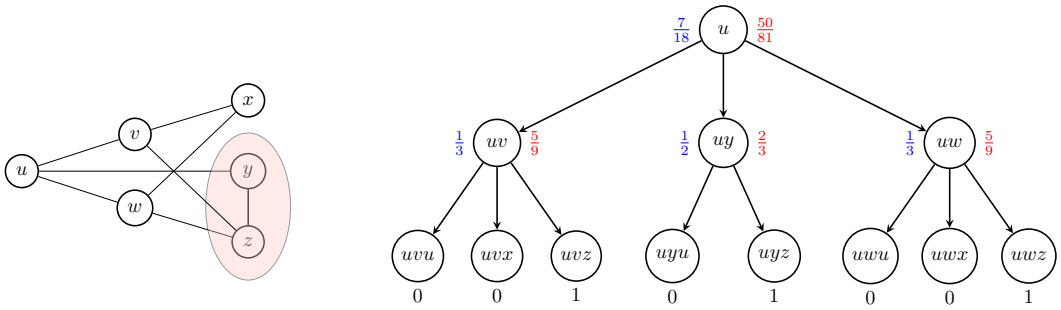


Figure 1 Illustration of a (non-lazy) walk on a non-regular graph starting from u with the objective of being at $\{y, z\}$ at step $t = 2$. The probabilities of achieving this are given in blue (left) for the SRW and in red (right) for the $\frac{1}{3}$-TBRW.

Suppose $\mathbf{x}$ is not a leaf. Then with probability $1 - \varepsilon$ we choose the next step of the walk uniformly at random in which case the probability of reaching S from $\mathbf{x}$ is just the average of $q_{\mathbf{y},S}$ over the offspring $\mathbf{y}$ of $\mathbf{x}$, otherwise we choose a maximal $q_{\mathbf{y},S}$. Thus the value of $\mathbf{x}$ is given by the ε-max/average of its offspring, that is

$$q_{\mathbf{x},S} = \mathrm{MA}_\varepsilon\left(\left(q_{\mathbf{y},S}\right)_{\mathbf{y}\in\Gamma^+(\mathbf{x})}\right). \tag{4}$$

We define the following potential function $\Phi^{(i)}$ on the i^{th} generation of the tree gadget $\mathcal{T}$:

$$\Phi^{(i)} = \sum_{|\mathbf{x}|=i} q_{\mathbf{x},S}^{1+\delta} \cdot \mathbb{P}\left[W_u(i) = \mathbf{x}\right]; \tag{5}$$

note that the sum ranges over all walks of length i. Notice that if $\mathbf{xy} \in E(\mathcal{T}_t)$ then $\mathbb{P}\left[W_u(|\mathbf{y}|) = \mathbf{y}\right] = \mathbb{P}\left[W_u(|\mathbf{x}|) = \mathbf{x}\right]/d^+(\mathbf{x})$. Also since each $\mathbf{y}$ with $|\mathbf{y}| = i$ has exactly one parent $\mathbf{x}$ with $|\mathbf{x}| = i - 1$ we can write

$$\Phi^{(i)} = \sum_{|\mathbf{x}|=i-1} \sum_{\mathbf{y}\in\Gamma^+(\mathbf{x})} q_{\mathbf{y},S}^{1+\delta} \cdot \frac{\mathbb{P}\left[W_u(i-1) = \mathbf{x}\right]}{d^+(\mathbf{x})}. \tag{6}$$

We now show that $\Phi^{(i)}$ is non-increasing in i. By combining (5) and (6) we can see that the difference $\Phi^{(i-1)} - \Phi^{(i)}$ is given by

$$\sum_{|\mathbf{x}|=i-1} \left(q_{\mathbf{x},S}^{1+\delta} - \frac{1}{d^+(\mathbf{x})} \sum_{\mathbf{y}\in\Gamma^+(\mathbf{x})} q_{\mathbf{y},S}^{1+\delta} \right) \mathbb{P}\left[W_u(i-1) = \mathbf{x}\right].$$

Recalling (4), to establish $\Phi^{(i-1)} - \Phi^{(i)} \geqslant 0$ it is sufficient to show the following inequality holds whenever $\mathbf{x}$ is not a leaf:

$$\mathrm{MA}_\varepsilon\left(\left(q_{\mathbf{y},S}\right)_{\mathbf{y}\in\Gamma^+(\mathbf{x})}\right)^{1+\delta} \geqslant \frac{1}{d^+(\mathbf{x})} \sum_{\mathbf{y}\in\Gamma^+(\mathbf{x})} q_{\mathbf{y},S}^{1+\delta}.$$

By taking $(1+\delta)^{\text{th}}$ roots this inequality holds for any $\delta \leqslant \varepsilon/(1-\varepsilon)$ by Lemma 7, and thus for δ in this range $\Phi^{(i)}$ is non-increasing in i.

Observe $\Phi^{(0)} = q_{u,S}^{1+\delta}$. Also if $|\mathbf{x}| = t$ then $q_{\mathbf{x},S} = 1$ if $\mathbf{x} \in S$ and 0 otherwise, it follows that

$$\Phi^{(t)} = \sum_{|\mathbf{x}|=t} q_{\mathbf{x},S}^{1+\delta} \cdot \mathbb{P}\left[W_u(t) = \mathbf{x}\right] = \sum_{|\mathbf{x}|=t} \mathbf{1}_{\mathbf{x}\in S} \cdot \mathbb{P}\left[W_u(t) = \mathbf{x}\right] = p_{u,S}.$$

Thus since $\Phi^{(t)}$ is non-decreasing $q_{u,S}^{1+\delta} = \Phi^{(t)} \geqslant \Phi^{(0)} = p_{u,S}$. The result for the ε-TBRW follows by taking $\delta = \varepsilon/(1-\varepsilon)$. If we let $\varepsilon = 1/d_{\mathsf{max}}$ we can apply the bound on $q_{u,S}$ for the ε-TBRW to the CRW by Proposition 1. ◀

4.3 A Conjecture of Azar et al. for the ε-Biased Walk

Azar, Broder, Karlin, Linial and Phillips [6] make the following conjecture for the ε-BRW.

▶ **Conjecture** ([6, Conjecture 1]). *In any graph, a controller can increase the stationary probability of any vertex from p to $p^{1-\varepsilon}$.*

They prove a weaker bound of $p^{1-O(\varepsilon)}$ for bounded-degree regular graphs. As a corollary of Theorem 5 we obtain a slightly weakened form of the conjecture for any graph where $d_{\mathsf{max}}/d_{\mathsf{min}}$ and t_{rel} are both subpolynomial in n. Our techniques are different and allow us to cover a larger class of graphs, including dense graphs as well as sparse ones, as well as getting closer to the conjectured bound.

▶ **Corollary 9.** *For any family of graphs such that $\log(t_{\mathsf{rel}} \cdot d_{\mathsf{max}}/d_{\mathsf{min}}) = o(\log n)$, a controller can increase the stationary probability of any vertex from p to $p^{1-\varepsilon+o_n(1)}$.*

The corollary follows from Theorem 5 and can be found in [26].

▶ Remark 10. As proven in Theorem 16, the optimal strategy is computable in polynomial time; thus a strategy achieving the above performance bound is also computable in polynomial time.

The original conjecture fails for the graph K_2, as no strategy for the ε-BRW can increase the stationary probability over that of a simple random walk. This motivates weakening the conjecture by replacing $p^{1-\varepsilon}$ by $p^{1-\varepsilon+o_n(1)}$, as in Corollary 9, however this fails for the star on n vertices, and non-bipartite counterexamples may be obtained by adding a small number of extra edges to the star. While these counterexamples have large degree discrepancy, their relaxation time is bounded. We believe the following should hold.

▶ **Conjecture 11.** *For any family of graphs such that $d_{\mathsf{max}}/d_{\mathsf{min}} = o(n)$, a controller can increase the stationary probability of any vertex from p to $p^{1-\varepsilon+o_n(1)}$.*

5 Computing Optimal Choice Strategies

In this section we focus on the following problem: given a graph G and an objective, how can we compute a series of choices for the walk which achieves the given objective in optimal expected time? In particular we consider the following computational problems related to our main objectives of max/minimising hitting times, cover times and stationary probabilities π_v.

`Stat` (G, w):	Find a CRW strategy min/maximising $\sum_{v \in V} w_v \pi_v$ for vertex weights $w_v \geqslant 0$.
`Hit` (G, v, S):	Find a CRW strategy minimising $H_v^{\mathsf{two}}(S)$ for a given $S \subseteq V(G)$ and $v \in V(G)$.
`Cov` (G, v):	Find a CRW strategy minimising $C_v^{\mathsf{two}}(G)$ for a given $v \in V(G)$.

We restrict the strategy in `Stat` to be unchanging so the stationary distribution is well defined. The analogous problems to `Stat` (G, w) and `Hit` (G, v, S) were studied in [6] for the biased random walk. While `Stat` is not one of our primary objectives, we include it here both as a natural problem to consider but also because of its relationship to `Hit` in the case where w is the indicator function of a set S; we shall abuse notation by writing `Stat`(G, S)

for this case. Clearly for `Stat` we must restrict ourselves to unchanging strategies for the stationary probabilities π_v to be well-defined; we shall show that `Hit` also has an unchanging optimal strategy.

For `Hit` and `Cov`, there are two possible interpretations of what it means to "find" a CRW strategy. Perhaps the most natural is to compute a sequence of optimal choices in an on-line fashion, that is at each time step to compute which of the two offered choices to accept. For any particular walk, with suitable memoisation, at most a polynomial number of such computations will be required for either problem: which choice to accept depends only on the current vertex, the two choices, and in the case of `Cov` the vacant set, which can change at most n times. We might alternatively want to compute a complete optimal strategy in advance; for `Hit` this requires only a polynomial number of single-choice computations, but for `Cov` the number of possible situations our strategy must cover will be exponential. However, we shall show that `Cov` is hard even for individual choices.

5.1 A Polynomial-Time Algorithm for `Stat` and `Hit`

First, we show how the (unknown) optimal values $H_x^{\mathsf{two}}(v)$ determine an optimal strategy for $\texttt{Hit}(G,\cdot,v)$. In the following two lemmas we will need to work with a multigraph F; in this context the choice offered at each stage is between two random edges from the current vertex.

▶ **Lemma 12.** *Let F be a multigraph and fix a vertex v. Let $v=v_0,v_1,\dots$ be an ordering of the vertices such that for all $i<j$ we have $H_{v_i}^{\mathsf{two}}(v)\leqslant H_{v_j}^{\mathsf{two}}(v)$. Let β be the deterministic unchanging strategy given by $\beta_{v_i,v_j}^{v_k}=1$ whenever $j<k$. Then β is optimal (among all strategies) for $\texttt{Hit}(F,x,v)$ for every $x\neq v$, and also for the problem of minimising $\mathbb{E}_v[\tau_v^+]$.*

▶ Remark 13. In particular, recalling that for an unchanging strategy $\pi_v=1/\mathbb{E}_v[\tau_v^+]$, it follows that β is an optimal strategy for $\texttt{Stat}(F,\{v\})$. However, this is true in a somewhat stronger sense, since optimality for `Stat` only requires minimising $\mathbb{E}_v[\tau_v^+]$ among unchanging strategies, whereas Lemma 12 shows that β minimises this quantity among all strategies; we shall need this extra strength.

Note that there may be other deterministic unchanging optimal strategies for $\texttt{Hit}(F,x,v)$. For example, if there are multiple vertices with the same optimal hitting time, we may choose between them arbitrarily, and in particular may have a cyclic order of preference which is not consistent with any single ordering. The following lemmas will enable us to show that a good enough approximation to an optimal strategy must itself be optimal.

▶ **Lemma 14.** *Let F be a multigraph with at most n vertices and at most $\binom{n}{2}$ edges, and fix a vertex v. Let α be any unchanging strategy for $\texttt{Stat}(F,\{v\})$. Suppose there exist vertices x,y,z with $y,z\in\Gamma^+(x)$, $H_y^{\mathsf{two}}(v)<H_z^{\mathsf{two}}(v)$ and $\alpha_{x,y}^z\leqslant 1/2$. Then π_v^α differs from the optimal value by at least $n^{-4(n+1)}(H_z^{\mathsf{two}}(v)-H_y^{\mathsf{two}}(v))$.*

▶ **Lemma 15.** *For any simple graph G of order n and every pair of vertices x,y with $H_x^{\mathsf{two}}(S)<H_y^{\mathsf{two}}(S)$ we have $H_y^{\mathsf{two}}(S)-H_x^{\mathsf{two}}(S)>n^{-2n^2}$.*

For any graph G, $v \in V$ and weighting $w : V \to [0, \infty)$ on the vertices of G we can phrase $\texttt{Stat}\,(G, w)$ as an optimisation problem as follows, where we shall encode our actions using the probabilities $\alpha^z_{x,y} = \mathbb{P}\,[\,X_{t+1} = y \mid X_t = x, c = \{y, z\}\,]$ from Section 2.

$$\begin{aligned}
\text{maximize:} \quad & \sum_{v \in V} w_v \pi(v) \\
\text{subject to:} \quad & \pi(x) = \sum_{y \in \Gamma(x)} \pi(y) \cdot \frac{2 \sum_{z \in \Gamma(y)} \alpha^z_{y,x}}{d(x)^2}, && \forall x \in V \\
& \sum_{x \in V} \pi(x) = 1, \\
& \alpha^y_{x,z} \in [0, 1], && \forall xz, xy \in E \\
& \alpha^y_{x,z} = 1 - \alpha^z_{x,y}, && \forall xz, xy \in E
\end{aligned} \tag{7}$$

For minimising the stationary probabilities we maximise -1 times the objective function.

▶ **Theorem 16.** *For any multigraph G and weight function $w : V \to [0, \infty)$ a policy solving the problem $\texttt{Stat}\,(G, w)$ to within an additive ε factor can be computed in time $\mathsf{poly}(|E|, \log(1/\varepsilon))$.*

To prove Theorem 16 the quadratic terms in (7) can be eliminated using the same substitution as [6], we can then solve (7) as a Linear Program.

▶ **Theorem 17.** *For any graph G and any $S \subset V$, a solution to $\texttt{Hit}\,(G, x, S)$ for every $x \in V \setminus S$ can be computed in time $\mathsf{poly}(n)$.*

Proof. Contract S to a single vertex v to obtain a multigraph F; where a vertex x has more than one edge to S in G, retain multiple edges between x and v in F. Note that F has at most n vertices and at most $\binom{n}{2}$ edges. Provided that the CRW on G has not yet reached S, there is a natural correspondence between strategies on G and F with the same transition probabilities, and it follows that $H^{\mathsf{two}}_x(S)$ for G and $H^{\mathsf{two}}_x(v)$ for F are equal for any $x \in V(G) \setminus S$. We compute an optimal strategy to $\texttt{Stat}(F, \{v\})$ to within an additive error of $\varepsilon := n^{-10n^2}$; note that $\log(1/\varepsilon) = o(n^3)$ and so this may be done in time $\mathsf{poly}(n)$ by Theorem 16. Applying Lemma 14 to F and Lemma 15 to G, using the equality of corresponding hitting times, implies that this strategy has $\alpha^z_{x,y} > 1/2$ whenever $H^{\mathsf{two}}_y(v) < H^{\mathsf{two}}_z(v)$, and so rounding each of the probabilities $\alpha^z_{x,y}$ to the nearest integer gives an optimal strategy (on F) for every x, which may easily be converted to an optimal strategy for G. ◀

5.2 A Hardness Result for $\texttt{Cov}\,(G, v)$

We show that in general even the on-line version of $\texttt{Cov}\,(G, v)$ is NP-hard. To that end we introduce the following problem, which represents a single decision in the on-line version. The input is a graph G, a current vertex u, two vertices v and w which are adjacent to u, and a visited set X, which must be connected and contain u.

$\texttt{NextStep}\,(G, u, v, w, X)$: Choose whether to move to v or w so as to minimise the expected time for the CRW to visit every vertex not in X, assuming an optimal strategy is followed thereafter.

Any such problem may arise during a random walk with choice on G starting from any vertex in X, no matter what strategy was followed up to that point, since with positive probability no real choice was offered in the previous walk.

▶ **Theorem 18.** $\texttt{NextStep}$ *is* NP*-hard, even if G is constrained to have maximum degree* 3.

Proof. We give a (Cook) reduction from the NP-hard problem of either finding a Hamilton path in a given graph H or determining that none exists. This is known to be NP-hard even if H is restricted to have maximum degree 3 [22].

We shall find it more convenient to work with the following problem, which takes as input a graph G, a current vertex u and a connected visited set X containing u.

`BestStep`(G, u, X): Choose a neighbour of u to move to so as to minimise the expected time for the CRW to visit every vertex not in X, assuming an optimal strategy is followed thereafter.

We may solve `BestStep`(G, u, X) by computing `NextStep`(G, u, v, w, X) for every pair v, w of neighbours of u; since all optimal neighbours must be preferred to all others, this will identify a set of one or more optimal choices for `BestStep`(G, u, X). Consequently, it is sufficient to reduce the Hamilton path search problem to `BestStep`.

Given an n-vertex graph H, construct the graph G as follows. First replace each edge of H by a path of length $2cn^2$ through new vertices. Next add a new pendant path of length n^3 starting at the midpoint of each path corresponding to an edge of H. Finally, add edges to form a cycle consisting of the end vertices of these pendant paths (in any order). Note that if H has maximum degree 3, so does G.

Fix a starting vertex u and a non-empty unvisited set $Y \subseteq V(H) \setminus \{u\}$, and set $X = V(G) \setminus Y$. (The purpose of the second and third stages of the construction is to make X connected without affecting the optimal strategy.) Suppose that H contains at least one path of length $|Y|$ starting at u which visits every vertex of Y; in particular if $Y = V(H) \setminus \{u\}$ this is a Hamilton path of H. We claim that any optimal next step is to move towards the next vertex on some such path. Assuming the truth of this claim, an algorithm to find a Hamilton path starting at x, if one exists, is to set $u = x$ and $Y = V(H) \setminus \{x\}$, then find the vertex y such that moving towards y is optimal, set $u = y$ and remove y from Y, then continue. If this fails to find a Hamilton path, repeat for other possible choices of x.

To prove the claim, first we argue by induction that there is a strategy to visit every vertex in $|Y|$ in expected time $(4cn^2 + O(n))|Y|$, where the implied constant does not depend on c. This is clearly true for $|Y| = 0$. Let y be the next vertex on a suitable path in H, and let z be the middle vertex of the path corresponding to the edge uy. Attempting to reach z by a straightforward strategy, the distance to z evolves as a random walk with probability $3/4$ of decreasing unless the current location is a branch vertex. We thus reach z in expected time $2cn^2$ plus an additional constant time for each visit to u, of which we expect $O(d(u)) = O(n)$, giving a total expected time of $2cn^2 + O(n)$ (if the walker is forced to a different branch vertex first, the expected time to return from this point is polynomial in n, but this event occurs with exponentially small probability). Similarly, the time taken to reach y from z is $2cn^2 + O(1)$. Once y is reached, there is (by choice of y) a path of length $|Y| - 1$ in H starting from y and visiting all of $Y \setminus \{y\}$. Thus, by induction, the required bound holds. Secondly, suppose that an optimal first step in a strategy from u moves towards a vertex y' of H which is not the first step in a suitable path. Since the expected remaining time decreases whenever an optimal step is taken, two successive optimal steps cannot be in opposite directions unless the walker visits a vertex of Y in between. Thus the optimal strategy is to continue in the direction of y' if possible, and such a strategy reaches y' before returning to u with at least constant probability p, and this takes at least $2cn^2$ steps. Note that the expected time taken to reach another vertex of H from a vertex in H, even if the walker is purely trying to minimise this quantity, is at least $4cn^2$, and from either u or y' at least $|Y|$ such transitions are necessary to cover Y. Thus such a strategy, conditioned on the first step being in the direction of y', has expected time at least $4cn^2 + 2pcn^2$, which, for suitable choice of c, proves the claim. ◀

5.3 Computing $\texttt{Cov}(G, v)$ via Markov Decision Processes

To compute a solution for $\texttt{Cov}(G, v)$ we can encode the cover time problem as a hitting time problem on a (significantly) larger graph. For a proof of the following lemma see [23].

▶ **Lemma 19.** *For any graph $G = (V, E)$ let the (directed) auxiliary graph $\widetilde{G} = (\widetilde{V}, \widetilde{E})$ be given by $\widetilde{V} = V \times \mathcal{P}(V)$ and $\widetilde{E} = \{((i, S), (j, S \cup j)) \mid ij \in E\}$. Then solutions to $\texttt{Cov}(G, v)$ correspond to solutions to $\texttt{Hit}\big(\widetilde{G}, \widetilde{v}, W\big)$ and vice versa, where $W = \{(u, V) \mid u \in V\}$.*

In light of Lemma 19 it may appear that we can solve $\texttt{Cov}(G, v)$ by converting it to an instance of $\texttt{Hit}\big(\widetilde{G}, \widetilde{v}, W\big)$ and appealing to Theorem 17. This is unfortunately not the case as $\widetilde{G}$ is a directed graph and Theorem 17 cannot handle directed graphs. Lemma 19 is still of use as we can phrase $\texttt{Hit}$ in terms of Markov Decision processes and then standard results tell us that an optimal strategy for the problem can be computed in finite time.

A Markov Decision Process (MDP) is a discrete time finite state stochastic process controlled by a sequence of decisions [18]. At each step a controller specifies a probability distribution over a set of actions which may be taken and this has a direct affect on the next step of the process. Costs are associated with each step/action and the aim of the controller is to minimise the total cost of performing a given task, for example hitting a given state. In our setting the actions are orderings of the vertices in each neighbourhood and the cost of each step/action is one unit of time. $\texttt{Hit}\big(G, u, v\big)$ is then an instance of the optimal first passage problem which can be solved as a linear program. In our setting actions are orderings of neighbourhoods and so the linear program has $\sum_{x \neq v} d(x)!$ many constraints [18, page. 58]. Since, by the construction in Lemma 19, the out degrees of vertices in the directed graph $\widetilde{G}$ are the same as those of the corresponding vertices of G we obtain.

▶ **Corollary 20.** *For any graph G and $v \in V$ an optimal policy for the problem $\texttt{Cov}(G, v)$ can be computed in exponential time.*

▶ Remark 21. Applying the LP from [18, page. 58] to graphs with degrees of order higher than $\mathsf{poly}(\log n)$ will not result in a polynomial time algorithm for $\texttt{Hit}$. This is why we took a different approach to find a polynomial time algorithm in Section 5.1.

6 Summary and Future Work

In this paper we proposed a new random walk process inspired by the power-of-two-choices paradigm. We derived several quantitative bounds on the hitting and cover times, and also presented a surprising dichotomy with regards to computing optimal strategies. Some tools we developed were also applicable to ε-biased walks and we made progress on a long standing conjecture [6].

While we were able to show that on expanders, the CRW significantly outperforms the SRW in terms of its cover time, it is natural to ask whether the cover time is $\Theta(n)$. In fact, it might even be possible for this bound to apply to *any* bounded-degree graph.

We have shown that $\texttt{Cov} \in \mathsf{EXP}$ and that the problem is NP-Hard, it would be interesting to find a complexity class for which the problem is complete.

Our focus here was on hitting and cover times, as well as maximising stationary probabilities, but another natural question is whether we can define a meaningful notion of mixing time and analyse the speed-up achieved by a CRW in comparison to a simple random walk.

References

1 Mohammed Amin Abdullah, Colin Cooper, and Moez Draief. Speeding up cover time of sparse graphs using local knowledge. In *Combinatorial algorithms*, volume 9538 of *Lecture Notes in Comput. Sci.*, pages 1–12. Springer, [Cham], 2016. doi:10.1007/978-3-319-29516-9_1.

2 Dimitris Achlioptas, Raissa M. D'Souza, and Joel Spencer. Explosive percolation in random networks. *Science*, 323(5920):1453–1455, 2009. doi:10.1126/science.1167782.

3 Noga Alon, Chen Avin, Michal Koucký, Gady Kozma, Zvi Lotker, and Mark R. Tuttle. Many random walks are faster than one. *Combin. Probab. Comput.*, 20(4):481–502, 2011. doi:10.1017/S0963548311000125.

4 Noga Alon, Itai Benjamini, Eyal Lubetzky, and Sasha Sodin. Non-backtracking random walks mix faster. *Commun. Contemp. Math.*, 9(4):585–603, 2007. doi:10.1142/S0219199707002551.

5 Chen Avin and Bhaskar Krishnamachari. The power of choice in random walks: An empirical study. *Computer Networks*, 52(1):44–60, 2008. (1) Performance of Wireless Networks (2) Synergy of Telecommunication and Broadcasting Networks. doi:10.1016/j.comnet.2007.09.012.

6 Yossi Azar, Andrei Z. Broder, Anna R. Karlin, Nathan Linial, and Steven Phillips. Biased random walks. *Combinatorica*, 16(1):1–18, 1996. doi:10.1007/BF01300124.

7 Yossi Azar, Andrei Z. Broder, Anna R. Karlin, and Eli Upfal. Balanced allocations. *SIAM J. Comput.*, 29(1):180–200, 1999. doi:10.1137/S0097539795288490.

8 Katy E. Beeler, Kenneth S. Berenhaut, Joshua N. Cooper, Meagan N. Hunter, and Peter S. Barr. Deterministic walks with choice. *Discrete Appl. Math.*, 162:100–107, 2014. doi:10.1016/j.dam.2013.08.031.

9 Michael Ben-Or and Nathan Linial. Collective coin flipping. In S. Micali, editor, *Randomness and Computation*, pages 91–115. Academic Press, New York, 1989.

10 Petra Berenbrink, Colin Cooper, and Tom Friedetzky. Random walks which prefer unvisited edges: exploring high girth even degree expanders in linear time. *Random Structures Algorithms*, 46(1):36–54, 2015. doi:10.1002/rsa.20504.

11 Petra Berenbrink, Artur Czumaj, Angelika Steger, and Berthold Vöcking. Balanced allocations: the heavily loaded case. In *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing*, pages 745–754. ACM, New York, 2000. doi:10.1145/335305.335411.

12 Tom Bohman and Alan Frieze. Addendum to "Avoiding a giant component" [Random Structures Algorithms **19** (2001), no. 1, 75–85; MR1848028]. *Random Structures Algorithms*, 20(1):126–130, 2002. doi:10.1002/rsa.10018.

13 Tom Bohman and David Kravitz. Creating a giant component. *Combin. Probab. Comput.*, 15(4):489–511, 2006. doi:10.1017/S0963548306007486.

14 Charles Bordenave, Marc Lelarge, and Laurent Massoulié. Nonbacktracking spectrum of random graphs: community detection and nonregular Ramanujan graphs. *Ann. Probab.*, 46(1):1–71, 2018. doi:10.1214/16-AOP1142.

15 Colin Cooper, Alan M. Frieze, and Tony Johansson. The Cover Time of a Biased Random Walk on a Random Cubic Graph. In James Allen Fill and Mark Daniel Ward, editors, *29th International Conference on Probabilistic, Combinatorial and Asymptotic Methods for the Analysis of Algorithms, AofA 2018, June 25-29, 2018, Uppsala, Sweden*, volume 110 of *LIPIcs*, pages 16:1–16:12. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018. doi:10.4230/LIPIcs.AofA.2018.16.

16 Colin Cooper, David Ilcinkas, Ralf Klasing, and Adrian Kosowski. Derandomizing random walks in undirected graphs using locally fair exploration strategies. *Distributed Computing*, 24(2):91–99, 2011. doi:10.1007/s00446-011-0138-4.

17 Roee David and Uriel Feige. Random walks with the minimum degree local rule have $O(n^2)$ cover time. *SIAM J. Comput.*, 47(3):755–768, 2018. doi:10.1137/17M1119901.

18 Cyrus Derman. *Finite state Markovian decision processes*. Mathematics in Science and Engineering, Vol. 67. Academic Press, New York-London, 1970.

19 Klim Efremenko and Omer Reingold. How well do random walks parallelize? In *Approximation, randomization, and combinatorial optimization*, volume 5687 of *Lecture Notes in Comput. Sci.*, pages 476–489. Springer, Berlin, 2009. doi:10.1007/978-3-642-03685-9_36.

20 Robert Elsässer and Thomas Sauerwald. Tight bounds for the cover time of multiple random walks. *Theoret. Comput. Sci.*, 412(24):2623–2641, 2011. doi:10.1016/j.tcs.2010.08.010.

21 Robert Fitzner and Remco van der Hofstad. Non-backtracking random walk. *J. Stat. Phys.*, 150(2):264–284, 2013. doi:10.1007/s10955-012-0684-6.

22 M. R. Garey, D. S. Johnson, and L. Stockmeyer. Some simplified NP-complete graph problems. *Theoret. Comput. Sci.*, 1(3):237–267, 1976. doi:10.1016/0304-3975(76)90059-1.

23 Agelos Georgakopoulos, John Haslegrave, Thomas Sauerwald, and John Sylvester. The Power of Two Choices for Random Walks. *Preprint*, 2019. arXiv:1911.05170.

24 Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric algorithms and combinatorial optimization*, volume 2 of *Algorithms and Combinatorics*. Springer-Verlag, Berlin, second edition, 1993. doi:10.1007/978-3-642-78240-4.

25 John Haslegrave and Jonathan Jordan. Preferential attachment with choice. *Random Structures Algorithms*, 48(4):751–766, 2016. doi:10.1002/rsa.20616.

26 John Haslegrave, Thomas Sauerwald, and John Sylvester. Time Dependent Biased Random Walks. *In preparation, to appear on ArXiv*, 2019.

27 Roger A. Horn and Charles R. Johnson. *Matrix Analysis, 2nd Ed.* Cambridge University Press, 2012. doi:10.1017/CBO9781139020411.

28 Satoshi Ikeda, Izumi Kubo, and Masafumi Yamashita. The hitting and cover times of random walks on finite graphs using local degree information. *Theoret. Comput. Sci.*, 410(1):94–100, 2009. doi:10.1016/j.tcs.2008.10.020.

29 N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4(4):373–395, 1984. doi:10.1007/BF02579150.

30 Yury Malyshkin and Elliot Paquette. The power of choice over preferential attachment. *ALEA Lat. Am. J. Probab. Math. Stat.*, 12(2):903–915, 2015.

31 Michael Mitzenmacher. The Power of Two Choices in Randomized Load Balancing. *IEEE Trans. Parallel Distrib. Syst.*, 12(10):1094–1104, 2001. doi:10.1109/71.963420.

32 Michael Mitzenmacher, Andréa W. Richa, and Ramesh Sitaraman. The power of two random choices: a survey of techniques and results. In *Handbook of randomized computing, Vol. I, II*, volume 9 of *Comb. Optim.*, pages 255–312. Kluwer Acad. Publ., Dordrecht, 2001. doi:10.1007/978-1-4615-0013-1_9.

33 Roberto I. Oliveira and Yuval Peres. Random walks on graphs: new bounds on hitting, meeting, coalescing and returning. In Marni Mishna and J. Ian Munro, editors, *Proceedings of the Sixteenth Workshop on Analytic Algorithmics and Combinatorics, ANALCO 2019, San Diego, CA, USA, January 6, 2019*, pages 119–126. SIAM, 2019. doi:10.1137/1.9781611975505.13.

34 Tal Orenshtein and Igor Shinkar. Greedy random walk. *Combin. Probab. Comput.*, 23(2):269–289, 2014. doi:10.1017/S0963548313000552.

35 Oliver Riordan and Lutz Warnke. Achlioptas process phase transitions are continuous. *Ann. Appl. Probab.*, 22(4):1450–1464, 2012. doi:10.1214/11-AAP798.

36 Joel Spencer and Nicholas Wormald. Birth control for giants. *Combinatorica*, 27(5):587–628, 2007. doi:10.1007/s00493-007-2163-2.

A Appendix

A.1 Proofs Omitted from Section 1

Proof of Proposition 1. It is sufficient to provide a strategy to simulate a given bias matrix, since we may then vary the strategy depending on t and $\mathcal{H}_t$ in order to simulate the ε-TBRW. Fix a bias matrix $\mathbf{B}$ with elements $b_{x,y}$ and let the weights $\alpha^z_{x,y}$ for the CRW be given by

$$\alpha^z_{x,y} = \frac{1}{2}\left[1 + \varepsilon d(x)\left(b_{x,y} - b_{x,z}\right)\right]$$

for each $x \in V(G)$ and $y, z \in \Gamma(x)$. Since $\varepsilon \leqslant 1/d_{\mathsf{max}} \leqslant 1/d(x)$, these weights satisfy (1), so this gives a valid CRW strategy.

For adjacent vertices x and y, let $q^{\mathsf{two}}_{x,y}$ and $q^{\varepsilon\mathrm{B}}_{x,y}$ denote the transition probabilities of the CRW and ε-biased walks respectively. By (3) we have

$$q^{\varepsilon\mathrm{B}}_{x,y} = \frac{1-\varepsilon}{d(x)} + \varepsilon b_{x,y},$$

and $\sum_{y\in\Gamma(x)} b_{x,y} = 1$ by definition of $\mathbf{B}$. Also, by (2) we have

$$\begin{aligned} q^{\mathsf{two}}_{x,y} &= \frac{2}{d(x)^2}\sum_{z\in\Gamma(x)} \alpha^z_{x,y} \\ &= \frac{1}{d(x)^2}\sum_{z\in\Gamma(x)} \left(1 + \varepsilon d(x)\left(b_{x,y} - b_{x,z}\right)\right) \\ &= \frac{1}{d(x)} + \varepsilon b_{x,y} - \frac{\varepsilon}{d(x)} \\ &= q^{\varepsilon\mathrm{B}}_{x,y}, \end{aligned}$$

as required. ◀

A.2 Proof of Theorem 3

We first need a lemma establishing that the CRW can simulate random walks on a suitable weighting of G.

▶ **Lemma 22.** *Fix a vertex v, and partition its neighbours into two sets, A and B. There is an unchanging strategy for the CRW such that whenever the walker is at v it moves to a random neighbour according to the probability distribution in which every vertex in B is twice as likely as every vertex in A.*

By considering the strategy at each vertex separately, we immediately obtain the following consequence.

▶ **Corollary 23.** *Let G be a weighted graph with weight function w, having the property that for any two incident edges xy, xz either $w(xy) = w(xz)$, $w(xy) = 2w(xz)$ or $2w(xy) = w(xz)$. Then there is an unchanging strategy for the CRW on G which simulates a random walk according to the weights w.*

For a weighted graph (G, w), write $w(G) = \sum_{e\in E(G)} w(e)$. We need an additional result on edge-crossing times of weighted graphs.

▶ **Lemma 24.** *Let (G, w) be a weighted graph, and let x be a vertex such that every edge incident with x has weight 1. Then for any vertex y adjacent to x, $H_y(x) \leqslant w(G) + w(G \setminus x)$.*

Proofs of the three Lemmas above can be found [23, Sec. 3]. We are now ready to prove the main result of this section.

Proof of Theorem 3. We have to show that the above bounds apply to $H_y^{\mathsf{two}}(x)$ for two arbitrary vertices x, y. Define a weight function $w : E(G) \to \mathbb{Q}^+$ by $w(uv) = 2^{-\min(d(u,x),d(v,x))}$. Note that w satisfies the requirements of Corollary 23, so we can bound $H_y^{\mathsf{two}}(x)$ by the corresponding hitting time of the random walk on (G, w). We will bound that hitting time now.

Write d for the maximum distance of a vertex from x, and V_k for the set of vertices at distance exactly k from x. Note that if $y \in V_{k+1}$ then

$$H_y(x) \leqslant H_y(V_k) + \max_{z \in V_k} H_z(x),$$

and consequently

$$\max_{y \in V(G)} H_y(x) \leqslant \sum_{k=0}^{d-1} \max_{z \in V_{k+1}} H_z(V_k).$$

For each $0 \leqslant k \leqslant d-1$ let G_k be the simple weighted graph obtained by deleting $\bigcup_{i<k} V_i$ and identifying vertices in V_k to give a vertex v_k; if a vertex in V_{k+1} has multiple edges to V_k, delete all but one of them to leave a simple graph. Since removing edges between V_{k+1} and V_k cannot reduce the hitting time of V_k, we have for any $z \in V_{k+1}$ that $H_z^G(V_k) \leqslant H_z^{G_k}(v_k)$. Note that the latter hitting time is unchanged by multiplying all weights by 2^k, and since every $z \in V_{k+1}$ is adjacent to v_k in G_k, by Lemma 24 we have $H_z^{G_k}(v_k) \leqslant 2^k(w(G_k) + w(G_k \setminus v_k))$. Thus

$$\max_{y \in V(G)} H_y(x) \leqslant \sum_{k=0}^{d-1} 2^k(w(G_k) + w(G_k \setminus v_k)).$$

If e is an edge between V_j and V_{j+1} then the contribution of e to the kth term of the above sum is 2^{k-j+1} if $k < j$, at most 1 if $k = j$ and 0 otherwise, so its total contribution is less than 3, and is less than 2 if e is one of the edges deleted to make G_j simple. If e is an edge within V_j then its contribution to the kth term is 2^{k-j+1} if $k < j$ and 0 otherwise, so its total contribution is less than 2. The first bound follows. Note that of the edges of the first type which are not deleted, there is exactly one from each vertex (other than x) to a vertex in a lower layer of G, and so these edges form a tree. Thus there are $n-1$ such edges, whose contribution is bounded by 3, and at most $\binom{n}{2} - (n-1)$ other edges, whose contribution is bounded by 2, giving a bound of $2\binom{n}{2} + n - 1 = n^2 - 1$. ◀

A.3 Deducing Theorems 4 & 5 from Theorem 6

To prove Theorems 4 & 5 from Theorem 6 we need some elementary lemmas concerning random walks. The Proofs of Lemmas 25 & 26 can be found in [23].

▶ **Lemma 25.** *Let $U(t)$ be the number of unvisited vertices at time t by a SRW on a graph and let $T_{n/2^x}$ be the number of SRW steps taken before $U \leqslant n/2^x$. Then*

$$\mathbb{E}\left[\, U(2x \cdot t_{\mathsf{hit}}) \,\right] \leqslant \frac{n}{2^x} \qquad \textit{and} \qquad \mathbb{E}\left[\, T_{n/2^x} \,\right] \leqslant 4(x+1)t_{\mathsf{hit}}.$$

Theorems 4 & 5 bound the hitting/cover times in terms of $t_{\mathsf{rel}} = 1/(1-\lambda_2)$, where λ_2 is the largest non-trivial eigenvalue of the transition matrix of the lazy random walk (LRW). The relaxation time of LRW is more commonly studied than that of the simple random walk

(SRW) since laziness ensures that the walk is an aperiodic Markov chain, and hence the relaxation time is well defined. This provides a further obstacle to overcome since Theorem 6 uses the SRW rather than the LRW; our next lemma resolves this issue.

Let $p^{(t)}_{x,\cdot}$ be the distribution of the simple random walk after t steps, and write $\pi(S)$ for the stationary probability of a set S.

▶ **Lemma 26.** *For any graph G, $S \subset V$ and $x \in V$ there exists $t \leqslant 4t_{\mathsf{rel}} \ln n$ such that $p^{(t)}_{x,S} \geqslant \pi(S)/3$.*

Proof of Theorem 4. We first let a simple random walk cover all but $\alpha = \lfloor n/\log^C n \rfloor$ vertices, for some C to be specified later. By Lemma 25 if we let a simple random walk run for $4t_{hit} \cdot C \log_2 \log n$ steps then the expected size of the unvisited set will be at most $n/\log^C n$ as required. For a simple random walk $t_{hit} = \mathcal{O}\left(\frac{d_{\mathsf{avg}}}{d_{\mathsf{min}}} n\sqrt{t_{\mathsf{rel}}}\right)$ by [33, Thm. 1]. Thus it follows that the expected time τ_1 to complete the first phase is $\mathcal{O}\left(C(\log\log n) \cdot \frac{d_{\mathsf{avg}}}{d_{\mathsf{min}}} n\sqrt{t_{\mathsf{rel}}}\right)$.

We then have α different phases, labelled $\alpha, \alpha-1, \ldots, 1$, where in each phase we reduce the number of uncovered vertices by one. Consider any phase i where a set of i vertices are still uncovered; call this set $S_i \subseteq V$. By Lemma 26 there is some $T \leqslant 4t_{\mathsf{rel}} \log n$ and $t \leqslant T$ such that $p^{(t)}_{x,S} \geqslant \pi(S)/3 \geqslant d_{\mathsf{min}} \cdot i/(3nd_{\mathsf{avg}})$ and thus $q^{(t)}_{u,S_i} \geqslant (d_{\mathsf{min}} \cdot i/(3nd_{\mathsf{avg}}))^{1-\varepsilon}$ by Theorem 6. By considering independent trials with walks of length T the expected time until at least one vertex in S_i is visited is at most

$$\mathcal{O}\left(\left(\frac{n \cdot d_{\mathsf{avg}}}{i \cdot d_{\mathsf{min}}}\right)^{1-\varepsilon} \cdot t_{\mathsf{rel}} \cdot \log n\right).$$

Hence the expected time τ_2 to complete all α phases satisfies

$$\begin{aligned}\tau_2 &= \sum_{i=1}^{n/\log^C n} \mathcal{O}\left(\left(\frac{nd_{\mathsf{avg}}}{id_{\mathsf{min}}}\right)^{1-\varepsilon} t_{\mathsf{rel}} \log n\right) \\ &= \mathcal{O}\left(\left(\frac{nd_{\mathsf{avg}}}{d_{\mathsf{min}}}\right)^{1-\varepsilon} t_{\mathsf{rel}} \log n\right) \sum_{i=1}^{n/\log^C n} i^{\varepsilon-1}.\end{aligned}$$

Then, since $\sum_{i=1}^{n/\log^C n} i^{\varepsilon-1} \leqslant \left(n/\log^C n\right)^{\varepsilon} \cdot \sum_{i=1}^{n/\log^C n} i^{-1} \leqslant \left(n/\log^C n\right)^{\varepsilon} \cdot \log n$,

$$\begin{aligned}\tau_2 &= \mathcal{O}\left(\left(\frac{nd_{\mathsf{avg}}}{d_{\mathsf{min}}}\right)^{1-\varepsilon} t_{\mathsf{rel}} \log n\right) \cdot \mathcal{O}\left(\left(\frac{n}{\log^C n}\right)^{\varepsilon} \cdot \log n\right) \\ &= \mathcal{O}\left(n \cdot \left(\frac{d_{\mathsf{avg}}}{d_{\mathsf{min}}}\right)^{1-\varepsilon} \cdot t_{\mathsf{rel}} \cdot \frac{\log^2 n}{\log^{C\varepsilon} n}\right).\end{aligned}$$

Now if we let $C = \left(2 + \frac{\log t_{\mathsf{rel}}}{\log\log n}\right)/\varepsilon$ then $\log^{C\varepsilon} n = t_{\mathsf{rel}} \cdot \log^2 n$ and thus $\tau_2 = o(\tau_1)$. It follows that the contribution from the first phase dominates the second. Thus the total time is $\mathcal{O}(\tau_1)$ and for C above this is given by $\tau_1 = \mathcal{O}\left(n \cdot \varepsilon^{-1} \cdot \frac{d_{\mathsf{avg}}}{d_{\mathsf{min}}} \cdot \sqrt{t_{\mathsf{rel}}} \cdot \left(2 + \frac{\log t_{\mathsf{rel}}}{\log\log n}\right) \cdot \log\log n\right)$. ◀

Proof of Theorem 5. Let $T = 4 \cdot t_{\mathsf{rel}} \cdot \ln n$ then for any $x, v \in V$ there exists some $t \leqslant T$ such that $p^{(t)}_{x,y} \geqslant \pi(y)/3$ by Lemma 26. By Theorem 6 for any $x, y \in V$ there exists an ε-TB strategy and some $t \leqslant T$ such that $q^{(t)}_{x,y} \geqslant (\pi(y)/3)^{1-\varepsilon} \geqslant (d_{\mathsf{min}}/nd_{\mathsf{avg}})^{1-\varepsilon}/3$. Thus for any target vertex y and start vertex x we need in expectation at most $3\,(nd_{\mathsf{avg}}/d_{\mathsf{min}})^{1-\varepsilon}$ attempts to hit y in at most $T = 4t_{\mathsf{rel}} \ln n$ steps, the result follows. ◀

A.4 Proofs from section 5.1

Proof of Theorem 16. We prove the simple graph case; this proof may be easily extended for multigraphs with suitably adapted notation. The optimisation problem (7) above can be rephrased as a Linear Program by making the substitution $r_{x,y,z} = \pi(x) \cdot \alpha^z_{x,y}$. Either the Ellipsoid method or Karmarkar's algorithm will approximate the solution to within an additive $\varepsilon > 0$ factor in time which is polynomial in the dimension of the problem and $\log(1/\varepsilon)$, see for example [29, 24]. ◀

Proof of Lemma 12. Fix an optimal strategy α for $\texttt{Hit}(F, x, v)$, and for each $y \in \Gamma(x)$ write q_y for the probability that the first step under this strategy is from x to y. Recall that $q_y = \sum_{z \in \Gamma x} \frac{2\alpha^z_{x,y}}{d(x)^2}$. Now given that the first step is at y, an optimal strategy for the remaining steps is precisely an optimal strategy for $\texttt{Hit}(F, y, v)$, and thus

$$H^{\mathsf{two}}_x(v) = \sum_{y \in \Gamma(x)} q_y H^{\mathsf{two}}_y(v).$$

Suppose there exist $y, z \in \Gamma(x)$ with $H^{\mathsf{two}}_y(v) < H^{\mathsf{two}}_z(v)$ but $\alpha^z_{x,y} < 1$ at the first step. By instead (at time 1 only) always choosing y in preference to z, the expected hitting time is decreased by $\frac{2}{d(x)^2}(1 - \alpha^z_{x,y})(H^{\mathsf{two}}_z(v) - H^{\mathsf{two}}_y(v))$, a contradiction. Thus we have $\alpha^z_{x,y} = 1$ if $H^{\mathsf{two}}_y(v) < H^{\mathsf{two}}_z(v)$ and $\alpha^z_{x,y} = 0$ if $H^{\mathsf{two}}_y(v) > H^{\mathsf{two}}_z(v)$. If $H^{\mathsf{two}}_y(v) = H^{\mathsf{two}}_z(v)$ then the expected hitting time does not depend on $\alpha^z_{x,y}$, and so any strategy satisfying these conditions at time 1, and thereafter following an optimal strategy, is itself optimal.

It follows by induction that following β for k turns and thereafter following α is optimal; since this gives arbitrarily good approximations of the expected hitting time under β, β is itself optimal for $\texttt{Hit}(F, x, v)$, and, since the definition of β does not depend on x, for $\texttt{Hit}(F, y, v)$ for any $y \neq v$.

Next we show that β is also an optimal strategy for minimising $\mathbb{E}_v[\,\tau^+_v\,]$. Suppose not, and let γ be an optimal strategy. Write q^γ_x for the probability of moving from v to x at time 1 under γ, and $H^\gamma_v(v^+)$ for $\mathbb{E}_v[\,\tau^+_v\,]$ under γ. Now

$$\begin{aligned} H^\gamma_v(v^+) &= \sum_{x \in \Gamma(v)} q^\gamma_x H^\gamma_x(v) \\ &\geqslant \sum_{x \in \Gamma(v)} q^\gamma_x H^\beta_x(v), \end{aligned}$$

by optimality of β for $\texttt{Hit}(F, x, v)$. Suppose $\gamma^y_{v,x} \neq \beta^y_{v,x}$ for some $x, y \in \Gamma(v)$. Replacing $\gamma^y_{v,x}$ and $\gamma^x_{v,y}$ by $\beta^y_{v,x}$ and $\beta^x_{v,y}$ respectively changes $\sum_{x \in \Gamma(v)} q^\gamma_x H^\beta_x(v)$ by $\frac{2}{d(v)^2}(\beta^y_{v,x} - \gamma^y_{v,x})(H^{\mathsf{two}}_x(v) - H^{\mathsf{two}}_y(v))$, which is non-positive by choice of β. Thus after a sequence of such changes we obtain

$$\begin{aligned} H^\gamma_v(v^+) &\geqslant \sum_{x \in \Gamma(v)} q^\gamma_x H^\beta_x(v) \\ &= H^\beta_v(v^+). \end{aligned}$$ ◀

Proof of Lemma 14. First we bound $H^\alpha_v(v^+) - H^\beta_v(v^+)$, where β is as described in Lemma 12. Consider the strategy of following α until the first time the walk either reaches v or is at x and offered a choice between y and z, and in the latter case following β until v is reached. The difference between this strategy and following α is $p(\alpha^z_{x,y} H^\alpha_y(v) + \alpha^y_{x,z} H^\alpha_z(v) - H^\beta_y(v))$, where p is the probability of the latter event occurring before the walk returns to v. Note

that

$$\begin{aligned}\alpha^z_{x,y}H^\alpha_y(v) + \alpha^y_{x,z}H^\alpha_z(v) - H^\beta_y(v) &\geqslant (\alpha^z_{x,y}-1)H^\beta_y(v) + \alpha^y_{x,z}H^\beta_z(v)\\ &= (1-\alpha^z_{x,y})(H^{\mathsf{two}}_z(v) - H^{\mathsf{two}}_y(v))\\ &\geqslant (H^{\mathsf{two}}_z(v) - H^{\mathsf{two}}_y(v))/2\end{aligned}$$

by Lemma 12 and the assumptions. Further,

$$p \geqslant 2\left(\frac{1}{\Delta(F)^2}\right)^{d(v,x)+1} \geqslant \binom{n}{2}^{-2n},$$

since with at least this probability the walk is forced along a specific shortest path to x, then offered a choice of y or z.

Thus the difference in $\mathbb{E}_v[\tau^+_v]$ between α and this hybrid strategy is at least

$$\zeta := \frac{1}{2}\binom{n}{2}^{-2n}(H^{\mathsf{two}}_z(v) - H^{\mathsf{two}}_y(v)),$$

and since β minimises this quantity among all strategies by Lemma 12, the same bound applies to the difference between α and β, giving

$$\pi_\alpha(v)^{-1} \geqslant \pi_\beta(v)^{-1} + \zeta,$$

and consequently

$$\pi_\alpha(v) \leqslant \pi_\beta(v) - \zeta\frac{\pi_\beta(v)^2}{1+\pi_\beta(v)\zeta}. \tag{8}$$

We have $1 \geqslant \pi_\beta(v) \geqslant \binom{n}{2}^{-1}$ by comparison with a simple random walk. Also we may crudely bound $t^{\mathsf{two}}_{\mathsf{hit}}F$ by noting that a SRW has probability at least $\binom{n}{2}^{1-n}$ of reaching any given vertex in at most $n-1$ steps, giving $\zeta < 1$. Combining these bounds with (8) gives the required result. ◀

Proof of Lemma 15. Note that the hitting times $(h_x)_{x\in V}$ for any given unchanging strategy are uniquely determined by the equations

$$h_x = \begin{cases} 1 + \sum_y \mathbf{P}_{xy}\cdot h_y & \text{if } x \notin S\\ 0 & \text{if } x \in S,\end{cases}$$

where $\mathbf{P}$ is the transition matrix for the strategy. This set of equations can be written as $\mathbf{Ah} = \mathbf{b}$, where $\mathbf{A} := (\mathbf{I} - \mathbf{Q})$, $\mathbf{Q}_{i,j} = \mathbf{P}_{x,y}$ if $i,j \notin S$ and 0 otherwise, and $\mathbf{b}$ is a 0-1 vector. Notice that since $S \neq \emptyset$ we have $\|\mathbf{Q}\| < 1$ and so $\mathbf{A}^{-1}$ exists [27, Cor. 5.6.16.] For any non-random strategy, and in particular for the optimal strategy described above, every transition probability from x is a multiple of $d(x)^{-2}$. Thus all the elements of $\mathbf{A}$ can be put over a common denominator D, where $D := \mathrm{LCM}(d(x)^2)_{x\in V} < (n!)^2 < n^{2n}/2$.

We have $\mathbf{h} = \mathbf{A}^{-1}\mathbf{b} = |\mathbf{A}|^{-1}\mathbf{C}^\mathsf{T}\mathbf{b}$, where $\mathbf{C}$ is the matrix of cofactors. Each entry in $\mathbf{C}$ can be put over a common denominator which is at most D^n, and so the same applies to each entry of $\mathbf{C}^\mathsf{T}\mathbf{b}$. Also, $|\mathbf{A}| < 2^n$ by Hadamard's inequality [27, Thm. 7.8.1]. It follows that if two hitting times differ, they differ by at least $(2D)^{-n}$. ◀

Graph Spanners in the Message-Passing Model

Manuel Fernández V
Computer Science Department, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA
manuelf@andrew.cmu.edu

David P. Woodruff
Computer Science Department, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA
dwoodruf@cs.cmu.edu

Taisuke Yasuda
Akuna Capital, Chicago, Illinois, USA
yasuda.taisuke1@gmail.com

Abstract

Graph spanners are sparse subgraphs which approximately preserve all pairwise shortest-path distances in an input graph. The notion of approximation can be additive, multiplicative, or both, and many variants of this problem have been extensively studied. We study the problem of computing a graph spanner when the edges of the input graph are distributed across two or more sites in an arbitrary, possibly worst-case partition, and the goal is for the sites to minimize the communication used to output a spanner. We assume the message-passing model of communication, for which there is a point-to-point link between all pairs of sites as well as a coordinator who is responsible for producing the output. We stress that the subset of edges that each site has is not related to the network topology, which is fixed to be point-to-point. While this model has been extensively studied for related problems such as graph connectivity, it has not been systematically studied for graph spanners. We present the first tradeoffs for total communication versus the quality of the spanners computed, for two or more sites, as well as for additive and multiplicative notions of distortion. We show separations in the communication complexity when edges are allowed to occur on multiple sites, versus when each edge occurs on at most one site. We obtain nearly tight bounds (up to polylog factors) for the communication of additive 2-spanners in both the with and without duplication models, multiplicative $(2k-1)$-spanners in the with duplication model, and multiplicative 3 and 5-spanners in the without duplication model. Our lower bound for multiplicative 3-spanners employs biregular bipartite graphs rather than the usual Erdős girth conjecture graphs and may be of wider interest.

2012 ACM Subject Classification Theory of computation → Communication complexity; Mathematics of computing → Graph algorithms

Keywords and phrases Graph spanners, Message-passing model, Communication complexity

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.77

Related Version A full version of the paper is available at `https://arxiv.org/abs/1911.05991`.

Funding *David P. Woodruff*: D. Woodruff would like to acknowledge support from the Office of Naval Research (ONR) grant N00014-18-1-2562.

Acknowledgements We would like to thank Gregory Kehne, Roie Levin, Chen Shao and Srikanta Tirthapura for helpful discussions, as well as the anonymous reviewers for their useful feedback.

1 Introduction

In modern computational settings, graphs are often stored in a distributed setting with edges living across multiple servers. This may happen when traditional, single-server methods for representing and processing massive graphs are no longer feasible and require parallel processing capability to complete. In other real world settings, different sites collect

information in different locations, naturally leading to a computational setting with an input graph distributed across servers. For example, the sites may correspond to sensor networks, different network servers, etc. Furthermore, the bottleneck in these settings is often in the *communication* between the servers, rather than the computation time within each of the servers. Computing synopses of distributed graphs in a communication-efficient manner has therefore become increasingly important.

We consider the problem of efficiently constructing a *graph spanner* in the message-passing model of communication. A graph spanner is a subgraph of the input graph, for which shortest path distances are approximately preserved in the subgraph. This property can immediately be used to approximately answer shortest path queries, diameter queries, connectivity queries, etc. Spanners have applications to internet routing [49, 22, 23, 47], using protocols in unsynchronized networks to simulate synchronized networks [46], distributed and parallel algorithms for shortest paths [19, 20, 27], and for constructing distance oracles [50, 12].

There are various notions of approximation provided by a spanner, such as *additive*, for which there is an integer $\beta \geq 1$ and one wants for all pairs u, v of vertices, that $d_H(u,v) \leq d_G(u,v) + \beta$, as well as *multiplicative*, in which case there is an integer $\alpha \geq 1$ and one wants for all pairs u, v of vertices, that $d_H(u,v) \leq \alpha \cdot d_G(u,v)$.

Message-Passing Model

In the message-passing model (see, e.g., [48, 55, 14, 56, 36]) there are s players, denoted $P^1, P^2, \ldots, P^s$, and each player holds part of the input. In our context, player P^i holds a subset E_i of a set of edges on a common vertex set V, and we define the graph G with vertex set V and edgeset $\bigcup_i E_i$. We focus on two input models, the *without duplication* edge model in which the E_i are pairwise disjoint, and the *with duplication* edge model in which the E_i are allowed to overlap.

In this model there is also a coordinator C who is required to compute a function defined on the union of the inputs of the players. The communication channels in this model are point-to-point. For example, if C is communicating with P^i, then the remaining $s-1$ players do not see the contents of the message between C and P^i. We also do not allow the players to talk directly with each other; rather, all communication happens between the coordinator and a given player at any given time[1]. The coordinator C is responsible for producing the output.

The main resource measure we study is the *communication complexity*, that is, the total number of bits required to be sent between the servers in order to output such a spanner with high probability. While graph spanners have been studied in the offline model, as well as in various distributed models such as the CONGEST and LOCAL models, e.g., [16, 28, 25, 34, 15, 45] as well as in the local computation algorithms model [44], they have not been systematically studied in the message-passing model. The few related results we are aware of in the message-passing model are given in [56], where (1) the problem of testing graph connectivity was studied, which can be viewed as a very special case of a spanner, and (2) a result on additive 2-spanners which we discuss more and improve upon below. There is also work in related models such as [41], but such models require that the edges be randomly distributed, which may not be a realistic assumption in certain applications, e.g., if data is collected at sensors with different input distributions.

[1] This model only mildly increases the communication cost over a complete point-to-point network in which each pair of players can communicate with each other. Indeed, if P^i wishes to speak to P^j, then P^i can forward a message through the coordinator who can send it to P^j. Thus the communication increases by a multiplicative factor of 2 and an additive $O(\log s)$ per message to specify where to forward the message. As these factors are small in our context, we will focus on the coordinator model.

Table 1 Our results.

	With duplication		Without duplication	
Spanner	LB	UB	LB	UB
$+2$	$\Omega(sn^{\frac{3}{2}})$	$\tilde{O}(sn^{\frac{3}{2}})$	$\Omega(\sqrt{s}n^{\frac{3}{2}} + sn)$	$\tilde{O}(\sqrt{s}n^{\frac{3}{2}} + sn)$
$+k$	$\Omega(sn^{\frac{4}{3}-o(1)})$	$\tilde{O}(sn^{\frac{3}{2}})$	$\Omega(n^{\frac{4}{3}-o(1)} + sn)$	$\tilde{O}(\sqrt{\frac{s}{k}}n^{\frac{3}{2}} + snk)$
$\times 3$	$\Omega(sn^{\frac{3}{2}})$	$\tilde{O}(sn^{\frac{3}{2}})$	$\Omega(s^{\frac{1}{2}}n^{\frac{3}{2}} + sn)$	$\tilde{O}(s^{1/2}n^{\frac{3}{2}} + sn)$
$\times 5$	$\Omega(sn^{\frac{4}{3}})$	$\tilde{O}(sn^{\frac{4}{3}})$	$\Omega(s^{\frac{1}{3}}n^{\frac{4}{3}} + sn)$	$\tilde{O}(s^{\frac{1}{3}}n^{\frac{4}{3}} + sn)$
$\times(2k-1)$, $k \geq 3$	$\Omega(sn^{1+\frac{1}{k}})$	$\tilde{O}(sn^{1+\frac{1}{k}})$	$\Omega(s^{\frac{1}{2}-\frac{1}{2k}}n^{1+\frac{1}{k}} + sn)$	$\tilde{O}(ks^{1-\frac{2}{k}}n^{1+\frac{1}{k}} + snk)$
$\times(2k-1)$, (sim.)	$\Omega(sn^{1+\frac{1}{k}})$	$\tilde{O}(sn^{1+\frac{1}{k}})$	$\Omega(sn^{1+\frac{1}{k}})$	$\tilde{O}(sn^{1+\frac{1}{k}})$

We also study a variant of the communication complexity in the message-passing model known as the *simultaneous communication complexity* for the multiplicative $(2k-1)$-spanner problem, in which each server is only allowed to send one round of communication to the coordinator [9, 52].

Turnstile Streaming Model

Finally, we present some simple results in the *turnstile streaming model*, in which the input graph is presented as a stream of insertion and deletion updates of edges. That is, we view our graph as an $\binom{n}{2}$-dimensional vector x starting with the zero vector, and we receive updates of the form $(e_i, \Delta_i) \in [\binom{n}{2}] \times \{\pm 1\}$ and increment the e_ith entry of x by Δ_i. Our input graph is then the graph that has the edge e iff $\sum_{i:e_i=e} \Delta_i > 0$. We assume that the input graph has no self-loops. In this model, we wish to design algorithms using low space and low number of passes through the stream. The study of graph problems in this model were pioneered by [2] and were subsequently studied by many other works, including [3, 4, 40, 38, 39].

1.1 Our Results

We summarize our results in Table 1. Note that the $\tilde{O}$ and $\tilde{\Omega}$ notation hides $\text{poly}(\log n)$ factors. Often our upper bounds are stated in terms of edges, but since each edge can be represented using $O(\log n)$ bits, we obtain the same upper bound in terms of bits up to an $O(\log n)$ factor. We study both the with duplication and without duplication edge models, and in all cases we consider a worst-case distribution of edges.

We give a number of communication versus approximation quality tradeoffs for additive spanners and multiplicative spanners. We describe each type of spanner we consider in the sections below, together with the results that we obtain. We obtain qualitatively different results depending on whether edges are allowed to be duplicated across the players, or if each edge is an input to exactly one player.

We point out some particular notable aspects of our results. First, we obtain nearly tight bounds (up to $\text{poly}(\log n)$ factors) for the communication of additive 2-spanners in both the with and without duplication models, multiplicative $(2k-1)$-spanners in the with duplication model, and multiplicative 3 and 5-spanners in the without duplication model. Second, in proving our tight lower bound for 3-spanners in the without duplication model (Theorem 16), we employ results from extremal graph theory on *biregular bipartite graphs*, which, to the best of our knowledge, is the first explicit use of such graphs in the context of lower bounds for spanners. All other lower bounds that we are aware of are obtained from

extremal graphs given by the Erdős girth conjecture (e.g., lower bounds in the streaming [10], local computation algorithm [44], and distributed [16, 28, 25, 34, 15, 45] models), and we believe that our use of biregular bipartite graphs may inspire tight lower bounds in other models in the future as well.

We note that our results slightly differ from traditional results on spanners, in that the sparsity of our spanner may be far from optimal. For instance, we show an algorithm for computing an additive 2-spanner in the without duplication model with near-optimal communication complexity of $\tilde{O}(\sqrt{s}n^{3/2})$ bits of communication, but the size of this spanner is $\tilde{O}(\sqrt{s}n^{3/2})$ edges, which may be much larger than the optimal $O(n^{3/2})$ edges when the number of servers s is very large. It is an interesting question to characterize the communication complexity of computing spanners of optimal size.

1.1.1 Additive Spanners

In the case of additive spanners, one is given an arbitrary graph G on a set V of n vertices and an integer parameter $\beta \geq 1$, and we want to output a subgraph H containing as few edges as possible so that $d_H(u, v) \leq d_G(u, v) + \beta$ for all pairs of vertices $u, v \in V$. The first such spanner was constructed by Aingworth et al. [5], which was slightly improved in [26, 29]. They showed, surprisingly, that for $\beta = 2$, it is always possible to achieve $|H| = O(n^{3/2})$. The next additive spanner was constructed in [11], where it was shown that for $\beta = 6$ one can achieve $O(n^{4/3})$ edges; see also [54] where the time complexity was optimized. Recently, it was shown in [18] how to achieve an additive spanner with $\tilde{O}(n^{7/5})$ edges for $\beta = 4$. In a breakthrough work [1], an $\Omega(n^{4/3-o(1)})$ lower bound was shown for any constant β.

The one previous result we are aware of for computing spanners in the message-passing model is for additive 2-spanners given in [56], for which an $\tilde{O}(sn^{3/2})$ upper bound was given which works with edge duplication. We first show that with edge duplication, the algorithm of [56] is *optimal*, by proving a matching $\Omega(sn^{3/2})$ lower bound. Our lower bound is a reduction from the s-player set disjointness problem [14]. We next consider the case when there is no edge duplication, and perhaps surprisingly, show that one can achieve an additive 2-spanner with $\tilde{O}(\sqrt{s}n^{3/2})$ communication, improving upon the $\tilde{O}(sn^{3/2})$ bound of [56], and given our lower bound in the case of edge duplication, providing a separation for additive spanners in the models with and without edge duplication. Our upper bound is based on observing that the dominant cost in implementing additive spanner algorithms in a distributed setting is that of performing a breadth-first search. We instead perform fewer breadth first searches to obtain a better overall communication cost than one would obtain by naïvely implementing an offline additive spanner algorithm, as is done in [56]. This algorithm is the starting point for our technically more involved upper bound, where we show that it is possible to obtain an additive k-spanner with $\tilde{O}(\sqrt{s/k}n^{3/2} + snk)$ total communication. We complement this result with a lower bound of $\Omega(n^{4/3-o(1)})$ for this problem.

We note that we are not able to obtain constant additive spanners with fewer than $n^{3/2}$ edges, as the dominant cost comes from having to do breadth first search trees, which is communication-intensive in the message-passing model. We conjecture that $\Theta(n^{3/2})$ may be the optimal communication bound for any additive spanner with constant distortion, unlike in the offline model where an $O(n^{4/3})$ edge bound is achievable.

1.1.2 Multiplicative Spanners

In the case of multiplicative spanners, we are given an arbitrary graph G on a set V of n vertices and an integer parameter $\alpha \geq 1$, and wish to output a subgraph H containing as few edges as possible so that $d_H(u, v) \leq \alpha \cdot d_G(u, v)$ for all pairs of vertices $u, v \in V$. For odd

integers $\alpha = 2k-1$, for any graph G on n vertices there exists a α-spanner with $O(n^{1+1/k})$ edges, for any integer $k \geq 1$ [7]. Further, this is known to be optimal for $k \in \{1,2,3,5\}$ [51, 53], while for general k the best known bounds are $\Omega(n^{1+2/(3k-3)})$ for odd k and $\Omega(n^{1+2/(3k-2)})$ for even k [42, 43].

Under a standard conjecture of Erdős [31], this bound of $O(n^{1+1/k})$ is in fact optimal for every k. Recall that the girth of an unweighted graph is the minimum length cycle in the graph. Erdős's conjecture is that there exist graphs G with $\Omega(n^{1+1/k})$ edges for which the girth is $2k+2$. Note that given such a G, if one were to delete any edge $\{u,v\}$ in G, then the distance from u to v would increase from 1 to $2k+1$, and therefore G is the only $2k-1$-spanner of itself, giving the $\Omega(n^{1+1/k})$ edge lower bound. Notice that G is also the only $2k$-spanner of itself, and so the $\Omega(n^{1+1/k})$ lower bound also holds for even integers $\alpha = 2k$, which is also optimal since, as mentioned above, there always exist $(2k-1)$-spanners with $O(n^{1+1/k})$ edges.

1.1.2.1 Message-Passing Model

We show that for computing a multiplicative $(2k-1)$-spanner with s players, in the edge model with duplication on n-node graphs, there is an $\Omega(s \cdot OPT_k)$ communication lower bound, where OPT_k is the maximum size of a $(2k-1)$-spanner of any graph. Our lower bound is again based on a reduction from the multiplayer set disjointness communication problem. A greedy algorithm shows that this bound is optimal, that is, we provide a matching $\tilde{O}(s \cdot OPT_k)$ upper bound.

If instead each edge occurs on exactly one server, note that the additive 2-spanner algorithm already gives a separation in the s parameter by providing a $\tilde{O}(\sqrt{s}n^{3/2} + sn)$ algorithm. We show that this is optimal up to polylog factors by showing a lower bound of $\Omega(\sqrt{s}n^{3/2})$ for multiplicative 3-spanners. This then gives near optimal lower bounds for additive 2-spanners as well. Our lower bound here uses for the first time, to the best of our knowledge, the theory of *biregular bipartite cages*, which may be of wider interest. For $k \geq 3$, we again show that there is a separation in the s parameter between the models with and without edge duplication, by showing that carefully balancing the complexity of a lesser known variant of the classic algorithm of [13], the cluster-cluster joining variant, can be implemented to use only $\tilde{O}(ks^{1-2/k}n^{1+1/k} + snk)$ communication. We complement this result with a lower bound of $\Omega(s^{1/2-1/2k}n^{1+1/k} + sn)$ communication via a reduction from the edge model with duplication, essentially by splitting vertices to transform the input instance with duplication into one without duplication. This bound is off by a factor of $O(s^{1/2-3/2k})$. For $k = 3$, the exponent on s is exactly correct, giving a nearly tight characterization of $\tilde{\Theta}(s^{1/3}n^{4/3})$ communication for the problem of computing multiplicative 5-spanners.

1.1.2.2 Simultaneous Communication

In the simultaneous communication model, we show an upper bound of $\tilde{O}(sn^{1+1/k})$ in the with duplication model and a lower bound of $\Omega(sn^{1+1/k})$ without duplication model under the Erdős girth conjecture, showing that the complexity is $\tilde{\Theta}(sn^{1+1/k})$ in all cases. The upper bound simply comes from locally computing a multiplicative $(2k-1)$-spanner of size $\Theta(n^{1+1/k})$ at each server, while the lower bound comes from constructing s edge-disjoint graphs on n vertices and $\Omega(n^{1+1/k})$ edges, a constant fraction of which must be sent to the server in the simultaneous communication model, as we show.

1.1.2.3 Turnstile Streaming Model

Finally, we note that implementing the cluster-cluster joining algorithm of [13] in the turnstile streaming model gives an algorithm for computing a multiplicative $(2k-1)$-spanner with $(\lfloor k/2 \rfloor + 1)$ passes and $\tilde{O}(n^{1+1/k})$ space. Our algorithm follows the techniques of [3], but implements a different version of the Baswana-Sen algorithm than they do, which allows us to save on the number of passes. Previously, in the regime of a small constant number of passes, [40] gave an algorithm for computing multiplicative spanners with distortion 2^k in $\tilde{O}(n^{1+1/k})$ space with two passes. Our result improves upon this in the distortion for $k = 3$, achieving an optimal space-distortion tradeoff.

2 Preliminaries

We use $[n]$ to denote $\{1, \ldots, n\}$. We often use capital letters X, Y, ... for sets, vectors, or random variables, and lower case letters x, y, ... for specific values of the random variables X, Y, ... For a set S, we use $|S|$ to denote the size of S.

Let $G = (V, E)$ be an undirected graph, where V is the vertex set and E is the edgeset. Let $n = |V|$ and $m = |E|$ denote the number of vertices and the number of edges, respectively. For a pair of vertices u, v in G, the distance between u and v is denoted by $d_G(u, v)$, which indicates the length of the shortest path connecting u to v. The results in this paper are for unweighted graphs, thus the length of a path is equal to the number of edges is contains.

We will also assume in this paper that $s \ll n$, e.g. $s = O(n^{\varepsilon})$ for a small constant ε: this is typically the case in practice, as well as the interesting regime for most of our bounds.

As for messages and communication, we assume that all communication is measured in terms of bits. All logarithms in this paper are base 2.

We make use of the Set Disjointness problem in the message-passing model, see, e.g., [17].

▶ **Definition 1** ($\text{DISJ}_{n,s}$). *There are s players and each of them holds a set $X_i \subseteq [n]$, and the goal is to determine whether $\bigcap_{i=1}^{s} X_i$ is empty or not.*

Recently in [14], the authors obtained a tight lower bound for this problem.

▶ **Theorem 2** ([14], Theorem 1.1). *For every $\delta > 0$, $n \geq 1$ and $s = \Omega(\log n)$, the randomized communication complexity of set disjointness in the message-passing model is $\Omega(sn)$ bits. That is, for every randomized protocol which succeeds with probability at least $2/3$ on any given set of inputs, there exists a set of inputs and random coin tosses of the players which causes the sum of message lengths of the protocol to be $\Omega(sn)$ bits. Further, for any $s \geq 2$, the randomized communication complexity of set disjointness is $\Omega(n)$ ([37]).*

3 Additive Spanners

In this section we study how to compute additive spanners of graphs in the message-passing model. Recall the definition of additive spanners.

▶ **Definition 3** (Additive spanners). *Given a graph G, a subgraph H is an* additive β-spanner for G *if for all $u, v \in V$, $d_G(u, v) \leq d_H(u, v) \leq d_G(u, v) + \beta$, where $d_G(u, v)$ and $d_H(u, v)$ are lengths of the shortest paths in G and H, respectively.*

3.1 Additive 2-Spanners with Duplication

As a warmup, we first consider the case when $\beta = 2$, and edge duplication is allowed. We will show the following:

▶ **Theorem 4.** *The optimal communication cost of the additive 2-spanner problem with edge duplication in the message passing model is $\tilde{\Theta}(sn^{3/2})$ bits.*

The following lemma is well known and follows from Theorem 2 of [33].

▶ **Lemma 5.** *For every n, there is a family of graphs on n vertices with $\Theta(n^{3/2})$ edges and girth at least 6.*

It is easy to see that if a graph G has girth 6, then G is the only possible spanner for G. Thus, we can tell whether a subgraph H of G is all of G or not just by looking at a valid spanner of H. We use this fact to reduce the set disjointness problem to the problem of computing spanners. We state our reduction in the following general lemma:

▶ **Lemma 6.** *Let R be a binary relation between graphs and members of a set $\mathcal{P}$. Suppose there is a family of graphs $\{G_n\}_n$ such that G_n has n vertices and $f(n)$ edges, and:*
1. *p_n is the unique member of $\mathcal{P}$ with $(G_n, p_n) \in R$*
2. *for any proper subgraph H of G_n, $(H, p_n) \notin R$*

Then for a graph G on n vertices, the communication complexity in the edge duplication case of computing p such that $(G, p) \in R$ is $\Omega(sf(n))$ bits.

For concreteness, in this example we may think of $\mathcal{P}$ as the set of all graphs, and define R to be the set of pairs (G, S) such that S is an additive 2-spanner of G.

Proof. We reduce from the set disjointness problem in the message-passing model. Given an instance of set disjointness with s players each holding $X_i \subseteq [f(n)]$, we create a graph G_n on n vertices. We give player i the edge indexed by j if $j \notin X_i$. If the coordinator outputs $p = p_n$, we output that $\bigcap_i X_i \neq \varnothing$, otherwise we output that $\bigcap_i X_i = \varnothing$. The coordinator outputs p_n if and only if all the edges of G_n are present among the players, which is the case if and only if $\bigcap_i X_i \neq \varnothing$. Therefore this procedure correctly decides set disjointness. Theorem 2 implies a $\tilde{\Omega}(sf(n))$ bit lower bound for the communication cost of computing p. ◀

Together, these pieces yield the following:

Proof of Theorem 4. For the lower bound, we observe that for a graph G as in Lemma 5 removing any edge (u, v) increases the distance from u to v to at least 5, and thus the only additive-2 spanner of G is G itself. By Lemma 6 with $\mathcal{P}$ as the set of all graphs and R as the set of pairs (G, H) such that H is an additive 2-spanner of G, we immediately have that the communication cost of finding a subgraph that is an additive 2-spanner is $\tilde{\Omega}(s|E(G)|) = \tilde{\Omega}(sn^{3/2})$.

For the upper bound, one can show that the well known algorithm of [26] for computing additive 2-spanners can be implemented in the message passing model with $\tilde{O}(sn^{3/2})$ bits of communication, even in the case of edge duplication. See the proof of Theorem 5 of [56] for details. ◀

3.2 Additive k-Spanners with Duplication

Unfortunately, we are not able to design algorithms with improved communication over the above additive 2-spanners even if we allow for larger additive distortion, despite the existence of algorithms for additive 6-spanners that achieve $O(n^{4/3})$ edges [11, 54]. In this section, we show a lower bound of $\Omega(sn^{4/3-o(1)})$ on the communication of additive k-spanners via a similar argument to the lower bound in Theorem 4.

▶ **Theorem 7.** *The randomized communication complexity of the additive k-spanner problem with edge duplication is $\Omega(sn^{4/3-o(1)})$.*

Proof. The proof follows essentially from applying Lemma 6 on the extremal graph of [1], with minor modifications. The details are in the appendix of the full version of the paper. ◀

3.3 Additive 2-Spanners without Duplication

We next show how to improve the upper bound of Theorem 4 when edges are not duplicated across servers. We note that we can assume all servers know the degree of every vertex, since this involves exchanging at most n numbers per player or $O(ns \log n)$ bits of communication. This is negligible compared to the rest of the communication assuming $s \ll n$.

First we write down some simple lemmas that we will make use of multiple times. The proofs of these can be found in the full version.

▶ **Lemma 8.** *Let $\mathcal{C}$ be a collection of sets over a ground set $\mathcal{U}$ each of size at least t. If we sample $\frac{|\mathcal{U}|}{t} \log |\mathcal{C}/\delta|$ elements from $\mathcal{U}$ uniformly with replacement, with probability at least $1-\delta$ we sample at least one element from each set in $\mathcal{C}$.*

▶ **Lemma 9.** *The deterministic communication complexity of computing a BFS (breadth first search) tree from a given node in the message passing model (with our without duplication) is $\tilde{O}(sn)$.*

We are ready to state the main algorithm of this section:

▶ **Theorem 10.** *The randomized communication complexity of the additive-2 spanner problem without edge duplication is $\tilde{O}(\sqrt{s}n^{3/2})$.*

Algorithm 1 $+2$ spanner without edge duplication.

Input: $G = (V, E)$.
Output: H, $+2$ spanner of G.

1: $V_1 = \{x \in V : \text{degree of } x \leq \sqrt{sn}\}$.
2: Each player sends the coordinator all edges adjacent to V_1. The coordinator aggregates these and compiles the set $E_1 = \{(u, v) \in E : u \in V, v \in V_1\}$.
3: The coordinator samples $2 \log n \cdot \sqrt{\frac{n}{s}}$ vertices uniformly at random with replacement from V, and let $\mathcal{R}$ denote the sampled vertex set.
4: Grow a BFS tree T_x from each $x \in \mathcal{R}$, let $E(T_x)$ be its edge set.
5: $F = E_1 \cup \bigcup_{x \in \mathcal{R}} E(T_x)$.
6: **return** $H = (V, F)$.

Proof. First we will show this algorithm provides a $+2$ spanner of G with constant probability.

Consider the set $V \setminus V_1$ of vertices with degree $\geq \sqrt{sn}$. Let $\mathcal{E}$ denote the event that $\mathcal{R}$ contains at least one vertex from the neighborhood of every vertex in this set. Applying Lemma 8 with $\mathcal{U} = V$, with $\mathcal{C}$ as the collection of neighborhoods of vertices in $V \setminus V_1$, and with $t = \sqrt{sn}$, we have that $\mathcal{E}$ occurs with probability at least $1 - o(1)$.

Now for an arbitrary pair of vertices u, v, let us consider the shortest path P connecting them in G. Suppose an edge $(x, y) \in P$ is missing from E_1. This implies that both x and y are in $V \setminus V_1$ and have degree strictly larger than $\sqrt{sn}$. If $\mathcal{E}$ holds, then x has a neighbor w sampled in $\mathcal{R}$. Then:

$$\begin{aligned} d_H(u, v) &\leq d_H(u, w) + d_H(w, v) \leq d_G(u, w) + d_G(w, v) \\ &\leq d_G(u, x) + 1 + d_G(x, v) + 1 = d_G(u, v) + 2 \end{aligned} \tag{1}$$

Above the first and third line follow from the triangle inequality, the second holds since H includes a BFS-tree rooted at w, and the last line since x is on the shortest path between u and v.

Next we bound the communication. Line 2 requires $\tilde{O}(\sqrt{s}n^{3/2})$ communication since each edge is in $N(V_1)$ is sent exactly once. By Lemma 9, growing $\log n\sqrt{n/s}$ BFS trees on line 4 requires $\tilde{O}(sn \cdot \sqrt{n/s} = \sqrt{s}n^{3/2})$ communication as well. Thus the total communication is $\tilde{O}(\sqrt{s}n^{3/2})$. ◀

We will later show that this is nearly optimal by showing a lower bound of $\Omega(\sqrt{s}n^{3/2})$ for the weaker problem of computing a multiplicative 3-spanner in Theorem 16 later in the paper.

3.4 Additive k-Spanners without Duplication

We now study the additive spanner problem with larger distortion in the without duplication model. Unfortunately, we are not able to get tight tight results in this setting, and closing this gap remains and outstanding question.

A lower bound of $\Omega(n^{4/3-o(1)})$ on the communication complexity follows from the strong incompressibility result of additive spanners in Theorem 2 of [1], the proof for which can be found in the full version.

▶ **Theorem 11.** *The randomized communication complexity of the additive k-spanner problem without edge duplication is $\Omega(n^{4/3-o(1)} + sn)$.*

On the algorithms side, we generalize the additive 2-spanner algorithm, showing that the communication drops off by a factor of $\sqrt{k}$ for larger additive distortions k.

▶ **Theorem 12.** *The randomized communication complexity of the additive k-spanner problem without edge duplication is $\tilde{O}(\sqrt{s/k}n^{3/2} + snk)$.*

Algorithm 2 $+k$ spanner without edge duplication.

Input: $G = (V, E)$
Output: H, $+s$ spanner of G

1: $V_1 = \{x \in V : \text{degree of } x \leq \sqrt{sn/k}\}$, $E_1 = \{(u, v) \in E : u \in V, v \in V_1\}$
2: Uniformly sample $\tilde{O}(\sqrt{n/sk}) + \tilde{O}(k)$ vertices from V, and let $\mathcal{R}_1$ denote the set of sampled vertices.
3: Grow a BFS tree T_x from every $x \in \mathcal{R}_1$, $E_2 = \{e \in E : e \in T_x \text{ for some } x \in \mathcal{R}_1\}$.
4: Uniformly sample $\tilde{O}(\sqrt{kn/s})$ vertices from V, and let $\mathcal{R}_2$ denote the sampled vertices.
5: Grow a truncated BFS tree T_x from every $x \in \mathcal{R}_2$, such that $|T_x| = n/k$. (In the last level of building the tree, arbitrarily include edges until $|T_x| = n/k$.) Let $E_3 = \{e \in E : e \in T_x \text{ for some } x \in \mathcal{R}_2\}$
6: $F \leftarrow E_1 \cup E_2 \cup E_3$
7: **return** $H = (V, F)$

Proof. We will first show that Algorithm 2 gives an additive k-spanner with probability at least $1 - o(1)$, then argue that it achieves the stated communication complexity. We may assume that $k \geq 6$, since otherwise Theorem 10 directly implies the claim. For convenience, let $N_\ell(e)$ denote the set of vertices within ℓ hops of either of the endpoints of e. Suppose

we have added only the edges E_1 which are adjacent to vertices of degree at most $\sqrt{sn/k}$, which is the case at the end of line 3, and consider the shortest path P in G between an arbitrary pair of vertices u, v. Let D be the set of edges of P missing from E_1.

- **Case 1**: $|D| \geq k$
  Since P is a simple path and we have already included all edges adjacent to low-degree vertices, there are collectively at least $k\sqrt{sn/k}/2 = \Omega(\sqrt{snk})$ vertices in the union of the $N_1(e)$ for all e missing from P. Let $\mathcal{E}_1$ be the event that $\mathcal{R}_1$ contains a vertex from this neighborhood for every choice of u, v with at least k missing path edges. If $\mathcal{E}_1$ holds, since in line 5 we include a BFS tree from each sampled vertex, this implies that the returned H is a $+2$ spanner for each such pair of u, v by the same reasoning as in the proof of Theorem 10.
- **Case 2**: $|D| < k$
  In what follows, we will argue that our construction either bridges each missing $e = (u', v') \in D$ with a 2-hop path, or places the root of a full BFS within distance 3 of P. If all $e \in D$ are bridged by 2-hop paths, we will argue that these paths are contained in truncated BFS trees included in line 5. Since there are at most k edges missing from P, and since the distance between the endpoints of e changes from $d_G(u', v') = 1$ to $d_H(u', v') = 2$, we will have that $d_H(u, v) \leq d_G(u, v) + k$. On the other hand if there is a BFS tree center a within distance 3 of u', then by the triangle inequality

$$\begin{aligned} d_H(u, v) &\leq d_H(u, a) + d_H(a, v) \\ &\leq d_G(u, u') + d_G(u', a) + d_G(a, u') + d_G(u', v) \\ &\leq d_G(u, u') + d_G(u', v) + 3 + 3 = d_G(u, v) + 6 \end{aligned} \tag{2}$$

  and since we may assume that $k \geq 6$, we will again have that $d_H(u, v) \leq d_G(u, v) + k$. Let $\mathcal{E}_2$ denote the event that $\mathcal{R}_2$ samples a vertex u_e in $N_1(e)$ for every missing edge e. Furthermore, let $\mathcal{E}_3$ denote the event that $\mathcal{R}_1$ samples at least one vertex v_e from $N_2(u_e)$ for every edge e for which $|N_2(u_e)| \geq n/k$. If $\mathcal{E}_2$ and $\mathcal{E}_3$ both hold, then:
  - **Case a**: for all $e \in D$, we have $N_2(u_e) \leq n/k$
    By $\mathcal{E}_2$, there is a truncated BFS center in $N_1(e)$ for all $e \in D$ that reaches both endpoints of e. So all missing edges have 2-hop paths.
  - **Case b**: For some $e \in D$, we have $N_2(u_e) > n/k$
    By $\mathcal{E}_3$, there is a full BFS center v_e in $N_2(u_e)$, which is at a distance at most

$$d_G(P, v_e) \leq d_G(P, u_e) + d_G(u_e, v_e) \leq 1 + 2 = 3 \tag{3}$$

    of P.

  By the arguments above, this sub-case analysis implies that H is a $+s$ spanner for all u, v for which at most s edges are missing from P.

It remains to show that $\mathcal{E}_1$, $\mathcal{E}_2$ and $\mathcal{E}_3$ hold simultaneously with probability $1 - o(1)$. All three can be made to hold individually with probability $1 - o(1)$ by applying Lemma 8, and this will determine the $\tilde{O}$ factors in Algorithm 2. By a union bound all three events hold simultaneously with probability $1 - o(1)$.

We now consider the communication complexity. Identifying the vertices of degree at most $\sqrt{sn/k}$ and communicating their incident edges in line 2 requires $\tilde{O}(\sqrt{s/k}n^{3/2})$ communication. By Lemma 9 the full BFS trees constructed in line 3 require $\tilde{O}(\sqrt{s/k}n^{3/2} + snk)$ communication. Similarly, the truncated BFS trees found in step 8 require $\tilde{O}(sn/k)$ communication each, for a total of $\tilde{O}(\sqrt{s/k}n^{3/2})$. Adding, we obtain an upper bound of $\tilde{O}(\sqrt{s/k}n^{3/2} + snk)$. ◀

4 Multiplicative Spanners

In this section we study how to compute multiplicative spanners of graphs in the message-passing model. Recall the definition of multiplicative spanners.

▶ **Definition 13** (Multiplicative spanners). *Given a graph G, a subgraph H is a* multiplicative α-spanner for G *if for all vertex pairs $u, v \in V$, $d_H(u,v) \leq \alpha \cdot d_G(u,v)$. where $d_G(u,v)$ and $d_H(u,v)$ are the shortest path distances in G and H respectively.*

4.1 Multiplicative $(2k-1)$-Spanners with Duplication

One can show that the classic greedy algorithm for computing multiplicative $(2k-1)$-spanners can be implemented to match the a lower bound that follows the same techniques as Theorem 4:

▶ **Theorem 14.** *The communication cost of the multiplicative $(2k-1)$-spanner problem with edge duplication is $\tilde{O}(sn^{1+1/k})$. Under Erdős' girth conjecture [30], the bound is tight, in other words the cost is $\tilde{\Theta}(sn^{1+1/k})$.*

The details can be found in the full version.

4.2 Multiplicative $(2k-1)$-Spanners without Duplication

For $k = 2$, the additive 2-spanner algorithm of Theorem 10 immediately gives us a multiplicative $(2k-1) = 3$-spanner algorithm with $\tilde{O}(\sqrt{s}n^{3/2})$ communication. We show that this bound is in fact tight. We will use the following fact about bipartite biregular graphs follows from Theorem 2 of [57] by taking an appropriate subgraph of their construction:

▶ **Corollary 15.** *Let s, n be such that $\sqrt{sn}$ be a prime power and $\sqrt{n/s}$ is an integer. Then, there exists a bipartite biregular graph of girth 6 on $\Theta(n)$ vertices where one side has size $\Theta(n/s)$ with common degree $\sqrt{sn}$ and one side with size $\Theta(n)$ with common degree $\sqrt{n/s}$.*

Using this extremal graph, we obtain the following theorem:

▶ **Theorem 16.** *The randomized communication cost of the multiplicative 3-spanner problem without edge duplication is $\Omega(\sqrt{s}n^{3/2})$.*

Proof. Recall the graph Z from Corollary 15 and let U be the partite set with $\Theta(n/s)$ vertices and common degree $\sqrt{sn}$ and let V be the partite set with $\Theta(n)$ vertices and common degree $\sqrt{n/s}$. Note that this graph has $m := \Theta(n^{3/2}/\sqrt{s})$. We will reduce s player set disjointness on m elements to the problem of finding multiplicative 3-spanners without edge duplication.

Consider s copies of the vertex sets $U^1, U^2, \ldots, U^s$, each belonging to each of the s players, as well as one copy of the vertex set V belonging to the coordinator. Now given an instance of set disjointness with s players each holding a set $X_i \subseteq [m]$, we define our input graph G by giving the ith player the edge indexed by $j \in [m]$ if and only if $j \notin X_i$. That is, if $\{a, b\} \in Z$ is the edge indexed by j, then we give P^i the edge $\{(a,i), b\}$, where $(a,i) \in U^i$ is the copy of the vertex $a \in U$ and b is the single copy of the vertex $b \in V$ that belongs to the coordinator. Note that this graph consists of $\Theta(n)$ vertices for the one copy of V and $\Theta(s \cdot n/s) = \Theta(n)$ for the s copies of U, and $\Theta(\sqrt{s}n^{3/2})$ edges without duplication.

We now show that the s sets X_i simultaneously intersect if and only if there is an edge $\{a, b\} \in Z$ that is missing from all s copies of the graph Z in the spanner H. It's clear that if $j \in \bigcap_{i=1}^{s} X_i$, then the edge $\{a, b\} \in Z$ indexed by j cannot be in the spanner H. Now

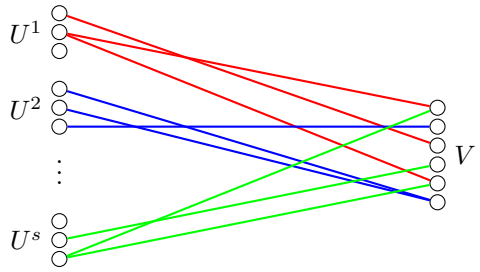


Figure 1 Our hard input instance.

suppose that no copy of the edge $\{a, b\} \in Z$ indexed by j is in the spanner H and suppose for contradiction that $j \notin X_i$ for some i, that is, the edge $\{a, b\} \in G$ belonged to some player P^i. We claim that there is no path of length at most 3 in the spanner H, contradicting that H is a multiplicative 3-spanner. Indeed, suppose for contradiction that a path $(a, i), v_1, (v_2, i'), b$ were in the spanner. Then $v_1 \neq b$ since we assumed that there were no copies of $\{a, b\}$ in the spanner, and similarly, $v_2 \neq a$. Then since $\{a, b\}$ is also an edge of Z, this implies that there is a 4-cycle a, v_1, v_2, b in Z, which contradicts that Z is of girth 6.

We thus conclude that the randomized communication complexity of this problem is

$$\Omega(sm) = \Omega\left(s\frac{n^{3/2}}{\sqrt{s}}\right) = \Omega(\sqrt{s}n^{3/2}), \tag{4}$$

as desired. ◀

▶ Remark 17. For bipartite biregular graphs of larger girth, the optimal size of the graph is unknown. However, a simple counting argument known as the *Moore bound* gives a lower bound on the number of vertices of a bipartite biregular graph of prescribed bidegree and girth [35, 32, 6], which shows limitations of the above technique for proving communication lower bounds for multiplicative spanners of larger distortion. More specifically, let $g = 2k + 2$ be the girth and let the two degrees be $\{d, sd\}$ (as we require one side of the bipartite graph to be of size $\Theta(n/s)$ in our proof technique). Then the Moore bound states that when k is odd, then the number of vertices is at least

$$n = \Omega\Big((sd)^{(k+1)/2}d^{(k-1)/2}\Big) = \Omega(s^{(k+1)/2}d^k) \tag{5}$$

which implies that we can't get a bound better than $\Omega(sdn) = \Omega(s^{1/2-1/2k}n^{1+1/k})$. We will in fact be able to show this bound under the Erdős girth conjecture for all k, as we show next. On the other hand, when k is even, then the Moore bound is

$$n = \Omega((sd)^{k/2}d^{k/2}) = \Omega(s^{k/2}d^k) \tag{6}$$

which implies that the best we can do is $\Omega(sdn) = \Omega(s^{1/2}n^{1+1/k})$, which is slightly better than the previous bound. However, it is known that these Moore bounds are not tight everywhere, and counterexamples exist in some limited parameter regimes, e.g. Theorem 4 of [24].

Finally, we note that more robust versions of the Moore bound have been shown for bipartite biregular graphs [35], where an analogue of the above bound holds even for irregular graphs.

For $k \geq 3$, one can implement the multiplicative spanner algorithm of [13] to get asymptotically better dependence on the number of servers s than the lower bound of Theorem 14. This separates the with edge duplication model from the without duplication model for all k, given the lower bound of Theorem 14.

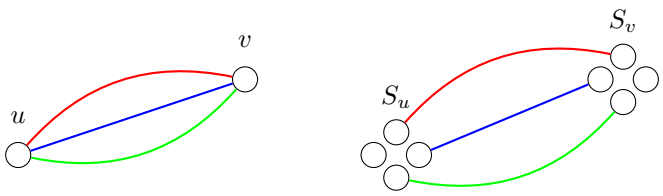


Figure 2 Converting an instance with edge duplication to one without.

▶ **Theorem 18.** *For $k \geq 3$, the randomized communication complexity of the multiplicative $(2k-1)$-spanner problem without edge duplication is $\tilde{O}(ks^{1-2/k}n^{1+1/k} + snk)$.*

Furthermore, this algorithm can be made deterministic with small modifications, essentially by having the players send $\tilde{O}(s^{1-2/k}n^{1+1/k})$ edges that allows for the coordinator to do the randomized part by themselves, which can then in turn be done deterministically by brute force.

▶ **Theorem 19.** *For $k \geq 3$, the deterministic communication complexity of the multiplicative $(2k-1)$-spanner problem without edge duplication is $\tilde{O}(ks^{1-2/k}n^{1+1/k} + snk)$.*

The details for both of these algorithms can be found in the full version.

We also show that in this model, a polynomial dependence on the parameter s is necessary. Specifically, we prove a $\Omega(s^{1/2-1/2k}n^{1+1/k})$ lower bound via a reduction from the lower bound for the multiplicative $(2k-1)$-spanner problem *with* duplication. The lower bound matches the algorithm of Theorem 18 exactly for $k = 3$ up to polylog factors, giving a communication complexity of $\tilde{\Theta}(s^{1/3}n^{4/3})$ in this case. For general k, the bounds are off by a factor of $\tilde{O}(s^{1/2-3/2k})$. Interestingly, this technique is not able to get us tight results for $k = 2$, giving a lower bound of $\Omega(s^{1/4}n^{3/2})$ instead.

▶ **Theorem 20.** *Under Erdős' girth conjecture, the randomized communication cost of the multiplicative $(2k-1)$-spanner problem without edge duplication is $\Omega(s^{1/2-1/2k}n^{1+1/k} + sn)$.*

Figure 2 illustrates the main idea of the proof: we can split vertices into $\sqrt{s}$ different vertices so that an instance with edge duplication on n vertices can be converted into one with $\sqrt{s}n$ vertices without duplication. Proofs for the lower and upper bounds are detailed in the full version.

4.3 Simultaneous Communication of Multiplicative $(2k-1)$-Spanners

We now prove our results for simultaneous communication for multiplicative $(2k-1)$-spanners.

Our algorithm comes from observing that for multiplicative spanners, each server can just locally compute a multiplicative $(2k-1)$-spanner of size $O(n^{1+1/k})$ and send it to the server for a $\tilde{O}(sn^{1+1/k})$ communication algorithm, which turns out to be optimal.

▶ **Theorem 21.** *The deterministic simultaneous communication complexity of multiplicative $(2k-1)$-spanners problem with duplication is $\tilde{O}(sn^{1+1/k})$.*

To prove the lower bound, we will make use of the following lemma.

▶ **Lemma 22.** *Let $s = o(n^{1/3-1/3k})$. Then under the Erdős girth conjecture, there exist s pairwise edge-disjoint graphs $E_1, E_2, \ldots, E_s$ on n vertices and $\Theta(n^{1+1/k})$ edges, each of girth $2k+2$.*

Proof. Under the Erdős girth conjecture, there exists a graph G on n vertices with $\Theta(n^{1+1/k})$ edges and girth $2k+2$. We now choose the s pairwise edge-disjoint graphs on n vertices as follows. First draw a random permutation of G for each server P^j for $j \in [s]$ by drawing a random permutation $\pi_j : [n] \to [n]$ of the vertices and giving the server P^j the edge set $\{\{\pi_j(u), \pi_j(v)\} : \{u, v\} \in E(G)\}$. To get pairwise edge-disjoint graphs, we now just delete any shared edges to produce our final edges E_j for $j \in [s]$.

Note that any subgraph of G also has girth at least $2k-1$, so E_j has girth at least $2k+2$ for all $j \in [s]$. It remains to show that in our parameter regime, this yields graphs of the desired size. Fix two distinct players P^i, P^j and edges $e_1 \in E_i$ and $e_2 \in E_j$. Then, the probability that e_1 collides with e_2 is $(n-2)!/n! = 1/n(n-1)$ and thus the expected number of edges shared between P^i and P^j is $n^{1+1/k}n^{1+1/k}/n(n-1) = \Theta(n^{2/k})$. By Markov's inequality, we delete $O(s^2n^{2/k})$ edges between these two players with probability at least $1-O(s^{-2})$. By the union bound, this is true simultaneously for all pairs of players with positive probability. In this event, each player deleted at most $s \cdot O(s^2n^{2/k}) = O(s^3n^{2/k}) = o(n^{1-1/k}n^{2/k}) = o(n^{1+1/k})$ edges, so each player still has a graph of size $\Theta(n^{1+1/k})$. ◀

Our lower bound now comes from either giving each of the players the above graphs with probability $1/2$, or giving only one player one of the above graphs with probability $1/2$. The intuition is as follows. When a player is the only one with a graph, then they must send their entire graph, while this is not the case when everyone has a graph. However, the players do not know whether everyone else has a graph or not, and therefore must always send their input graph whenever they get one.

▶ **Theorem 23.** *The randomized simultaneous communication complexity of multiplicative $(2k-1)$-approximate distance oracle problem without duplication is $\Omega(sn^{1+1/k})$.*

The above intuition is formalized in the full version.

4.4 Multiplicative $(2k-1)$-Spanners in the Dynamic Streaming Model

Finally, we note that implementing the Baswana-Sen cluster-cluster joining algorithm [13] in the turnstile streaming model gives a $(\lfloor k/2 \rfloor + 1)$-pass algorithm.

▶ **Theorem 24.** *There exists an algorithm for constructing a multiplicative $(2k-1)$-spanner using $\tilde{O}(n^{1+1/k})$ space and $\lfloor k/2 \rfloor + 1$ passes in the dynamic streaming model.*

The space-distortion tradeoff here is optimal under the Erdős girth conjecture, as graphs given by this conjecture must output themselves as spanners, which takes $\Omega(n^{1+1/k})$ bits of space.

5 Conclusions

We initiated the study of communication versus spanner quality in the message-passing model of communication, in which the edges of a graph are arbitrarily distributed, with or without duplication, across two or more players, and the players wish to execute a low communication protocol to compute a spanner. We believe there are several surprising aspects of these problems illustrated by our work, illustrating separations between models with and without edge duplication.

One open question is whether it is possible to obtain an additive spanner with constant distortion with $O(n^{4/3})$ communication for constant s. We show it is possible to obtain $O(n^{3/2})$ communication and constant distortion, but in the non-distributed setting it is

possible to obtain an additive 6-spanner with $O(n^{4/3})$ edges. Since known constructions involve computing many partial breadth-first search trees, we are not able to implement them in the message-passing model, nor are we able to exploit any of the literature for computing distributed BFS trees (see, e.g., [8]), without spending $\Omega(n^2)$ communication in the message-passing model. Yet another question is to extend our techniques to other notions of spanners, such as distance preservers [21] or mixed additive and multiplicative spanners [29]; see also the $(k, k-1)$ spanners in [11].

References

1 Amir Abboud and Greg Bodwin. The 4/3 Additive Spanner Exponent Is Tight. *J. ACM*, 64(4):28:1–28:20, September 2017.

2 Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. Analyzing graph structure via linear measurements. In Yuval Rabani, editor, *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 459–467. SIAM, 2012. doi:10.1137/1.9781611973099.40.

3 Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. Graph sketches: sparsification, spanners, and subgraphs. In Michael Benedikt, Markus Krötzsch, and Maurizio Lenzerini, editors, *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2012, Scottsdale, AZ, USA, May 20-24, 2012*, pages 5–14. ACM, 2012. doi:10.1145/2213556.2213560.

4 Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. Spectral Sparsification in Dynamic Graph Streams. In Prasad Raghavendra, Sofya Raskhodnikova, Klaus Jansen, and José D. P. Rolim, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques - 16th International Workshop, APPROX 2013, and 17th International Workshop, RANDOM 2013, Berkeley, CA, USA, August 21-23, 2013. Proceedings*, volume 8096 of *Lecture Notes in Computer Science*, pages 1–10. Springer, 2013. doi:10.1007/978-3-642-40328-6_1.

5 Donald Aingworth, Chandra Chekuri, Piotr Indyk, and Rajeev Motwani. Fast estimation of diameter and shortest paths (without matrix multiplication). *SIAM Journal on Computing*, 28(4):1167–1181, 1999.

6 Gabriela Araujo-Pardo, Alejandra Ramos-Rivera, and Robert Jajcay. Bipartite Biregular Cages and Block Designs. *arXiv preprint*, 2019. arXiv:1907.11568.

7 Baruch Awerbuch. Complexity of Network Synchronization. *J. ACM*, 32(4):804–823, 1985. doi:10.1145/4221.4227.

8 Baruch Awerbuch. Distributed Shortest Paths Algorithms (Extended Abstract). In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing, May 14-17, 1989, Seattle, Washigton, USA*, pages 490–500, 1989.

9 László Babai, Anna Gál, Peter G. Kimmel, and Satyanarayana V. Lokam. Communication Complexity of Simultaneous Messages. *SIAM J. Comput.*, 33(1):137–166, 2003. doi:10.1137/S0097539700375944.

10 Surender Baswana. Streaming algorithm for graph spanners - single pass and constant processing time per edge. *Inf. Process. Lett.*, 106(3):110–114, 2008. doi:10.1016/j.ipl.2007.11.001.

11 Surender Baswana, Telikepalli Kavitha, Kurt Mehlhorn, and Seth Pettie. New constructions of (alpha, beta)-spanners and purely additive spanners. In *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2005, Vancouver, British Columbia, Canada, January 23-25, 2005*, pages 672–681, 2005.

12 Surender Baswana and Sandeep Sen. Approximate distance oracles for unweighted graphs in $\tilde{O}(n^2)$ time. In *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2004, New Orleans, Louisiana, USA, January 11-14, 2004*, pages 271–280, 2004.

13 Surender Baswana and Sandeep Sen. A simple and linear time randomized algorithm for computing sparse spanners in weighted graphs. *Random Struct. Algorithms*, 30(4):532–563, 2007. doi:10.1002/rsa.20130.

14 Mark Braverman, Faith Ellen, Rotem Oshman, Toniann Pitassi, and Vinod Vaikuntanathan. A Tight Bound for Set Disjointness in the Message-Passing Model. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 668–677, 2013.

15 Keren Censor-Hillel and Michal Dory. Distributed Spanner Approximation. In Calvin Newport and Idit Keidar, editors, *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing, PODC 2018, Egham, United Kingdom, July 23-27, 2018*, pages 139–148. ACM, 2018. doi:10.1145/3212734.3212758.

16 Keren Censor-Hillel, Telikepalli Kavitha, Ami Paz, and Amir Yehudayoff. Distributed construction of purely additive spanners. In *International Symposium on Distributed Computing*, pages 129–142. Springer, 2016.

17 Arkadev Chattopadhyay and Toniann Pitassi. The story of set disjointness. *ACM SIGACT News*, 41(3):59–85, 2010.

18 Shiri Chechik. New Additive Spanners. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6-8, 2013*, pages 498–512, 2013.

19 Edith Cohen. Fast Algorithms for Constructing t-Spanners and Paths with Stretch t. *SIAM J. Comput.*, 28(1):210–236, 1998. doi:10.1137/S0097539794261295.

20 Edith Cohen. Polylog-time and near-linear work approximation scheme for undirected shortest paths. *J. ACM*, 47(1):132–166, 2000. doi:10.1145/331605.331610.

21 Don Coppersmith and Michael Elkin. Sparse sourcewise and pairwise distance preservers. *SIAM Journal on Discrete Mathematics*, 20(2):463–501, 2006.

22 Lenore Cowen. Compact Routing with Minimum Stretch. *J. Algorithms*, 38(1):170–183, 2001. doi:10.1006/jagm.2000.1134.

23 Lenore Cowen and Christopher G. Wagner. Compact roundtrip routing in directed networks. *J. Algorithms*, 50(1):79–95, 2004. doi:10.1016/j.jalgor.2003.08.001.

24 D De Caen and László A Székely. On dense bipartite graphs of girth eight and upper bounds for certain configurations in planar point–line systems. *Journal of Combinatorial Theory, Series A*, 77(2):268–278, 1997.

25 Michael Dinitz and Yasamin Nazari. Approximating k-spanners in the LOCAL model. *CoRR*, abs/1703.07417, 2017. arXiv:1703.07417.

26 Dorit Dor, Shay Halperin, and Uri Zwick. All-pairs almost shortest paths. *SIAM Journal on Computing*, 29(5):1740–1759, 2000.

27 Michael Elkin. Computing almost shortest paths. *ACM Trans. Algorithms*, 1(2):283–323, 2005. doi:10.1145/1103963.1103968.

28 Michael Elkin and Ofer Neiman. Efficient Algorithms for Constructing Very Sparse Spanners and Emulators. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 652–669, 2017.

29 Michael Elkin and David Peleg. (1+epsilon, beta)-Spanner Constructions for General Graphs. *SIAM J. Comput.*, 33(3):608–631, 2004.

30 P Erdős. On extremal problems of graphs and generalized graphs. *Israel Journal of Mathematics*, 2(3):183–190, 1964.

31 P Erdős. On some extremal problems in graph theory. *Israel Journal of Mathematics*, 3(2):113–116, 1965.

32 Slobodan Filipovski, Alejandra Ramos Rivera, and Robert Jajcay. On biregular bipartite graphs of small excess. *Discrete Mathematics*, 342(7):2066–2076, 2019. doi:10.1016/j.disc.2019.04.004.

33 Zoltán Füredi. Quadrilateral-free graphs with maximum number of edges. *preprint*, 1994.

34 Ofer Grossman and Merav Parter. Improved Deterministic Distributed Construction of Spanners. In *31st International Symposium on Distributed Computing, DISC 2017, October 16-20, 2017, Vienna, Austria*, pages 24:1–24:16, 2017.

35 Shlomo Hoory. The Size of Bipartite Graphs with a Given Girth. *J. Comb. Theory, Ser. B*, 86(2):215–220, 2002. doi:10.1006/jctb.2002.2123.

36 Zengfeng Huang, Bozidar Radunovic, Milan Vojnovic, and Qin Zhang. Communication Complexity of Approximate Matching in Distributed Graphs. In Ernst W. Mayr and Nicolas Ollinger, editors, *32nd International Symposium on Theoretical Aspects of Computer Science, STACS 2015, March 4-7, 2015, Garching, Germany*, volume 30 of *LIPIcs*, pages 460–473. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015. doi:10.4230/LIPIcs.STACS.2015.460.

37 Bala Kalyanasundaram and Georg Schnitger. The Probabilistic Communication Complexity of Set Intersection. *SIAM J. Discrete Math.*, 5(4):545–557, 1992.

38 Michael Kapralov, Yin Tat Lee, Cameron Musco, Christopher Musco, and Aaron Sidford. Single Pass Spectral Sparsification in Dynamic Streams. *SIAM J. Comput.*, 46(1):456–477, 2017. doi:10.1137/141002281.

39 Michael Kapralov, Navid Nouri, Aaron Sidford, and Jakab Tardos. Dynamic Streaming Spectral Sparsification in Nearly Linear Time and Space. *CoRR*, abs/1903.12150, 2019. arXiv:1903.12150.

40 Michael Kapralov and David P. Woodruff. Spanners and sparsifiers in dynamic streams. In Magnús M. Halldórsson and Shlomi Dolev, editors, *ACM Symposium on Principles of Distributed Computing, PODC '14, Paris, France, July 15-18, 2014*, pages 272–281. ACM, 2014. doi:10.1145/2611462.2611497.

41 Hartmut Klauck, Danupon Nanongkai, Gopal Pandurangan, and Peter Robinson. Distributed Computation of Large-scale Graph Problems. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 391–410, 2015.

42 Felix Lazebnik, Vasiliy A Ustimenko, and Andrew J Woldar. A new series of dense graphs of high girth. *Bulletin of the American mathematical society*, 32(1):73–79, 1995.

43 Felix Lazebnik, Vasiliy A. Ustimenko, and Andrew J. Woldar. A characterization of the components of the graphs D(k, q). *Discrete Mathematics*, 157(1-3):271–283, 1996. doi:10.1016/S0012-365X(96)83019-6.

44 Merav Parter, Ronitt Rubinfeld, Ali Vakilian, and Anak Yodpinyanee. Local Computation Algorithms for Spanners. In Avrim Blum, editor, *10th Innovations in Theoretical Computer Science Conference, ITCS 2019, January 10-12, 2019, San Diego, California, USA*, volume 124 of *LIPIcs*, pages 58:1–58:21. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2019. doi:10.4230/LIPIcs.ITCS.2019.58.

45 Merav Parter and Eylon Yogev. Congested Clique Algorithms for Graph Spanners. In Ulrich Schmid and Josef Widder, editors, *32nd International Symposium on Distributed Computing, DISC 2018, New Orleans, LA, USA, October 15-19, 2018*, volume 121 of *LIPIcs*, pages 40:1–40:18. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018. doi:10.4230/LIPIcs.DISC.2018.40.

46 David Peleg and Jeffrey D. Ullman. An Optimal Synchronizer for the Hypercube. *SIAM J. Comput.*, 18(4):740–747, 1989.

47 David Peleg and Eli Upfal. A Tradeoff between Space and Efficiency for Routing Tables (Extended Abstract). In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, pages 43–52, 1988.

48 Jeff M. Phillips, Elad Verbin, and Qin Zhang. Lower bounds for number-in-hand multiparty communication complexity, made easy. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 486–501, 2012.

49 Mikkel Thorup and Uri Zwick. Compact routing schemes. In *SPAA*, pages 1–10, 2001. doi:10.1145/378580.378581.

50 Mikkel Thorup and Uri Zwick. Approximate distance oracles. *J. ACM*, 52(1):1–24, 2005. doi:10.1145/1044731.1044732.

51 Jacques Tits. Sur la trialité et certains groupes qui s' en déduisent. *Publications Mathématiques de l'IHÉS*, 2:13–60, 1959.

52 Omri Weinstein and David P. Woodruff. The Simultaneous Communication of Disjointness with Applications to Data Streams. In Magnús M. Halldórsson, Kazuo Iwama, Naoki Kobayashi, and Bettina Speckmann, editors, *Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part I*, volume 9134 of *Lecture Notes in Computer Science*, pages 1082–1093. Springer, 2015. doi:10.1007/978-3-662-47672-7_88.

53 Rephael Wenger. Extremal graphs with no C^4's, C^6's, or C^{10}'s. *J. Comb. Theory, Ser. B*, 52(1):113–116, 1991. doi:10.1016/0095-8956(91)90097-4.

54 David P Woodruff. Additive spanners in nearly quadratic time. In *International Colloquium on Automata, Languages, and Programming*, pages 463–474. Springer, 2010.

55 David P. Woodruff and Qin Zhang. Tight bounds for distributed functional monitoring. In *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 - 22, 2012*, pages 941–960, 2012.

56 David P. Woodruff and Qin Zhang. When Distributed Computation Is Communication Expensive. In *Distributed Computing - 27th International Symposium, DISC 2013, Jerusalem, Israel, October 14-18, 2013. Proceedings*, pages 16–30, 2013.

57 Yuansheng Yang and Weifa Liang. The minimum number of vertices with girth 6 and degree set D=$\{r, m\}$. *Discrete Mathematics*, 269(1-3):249–258, 2003. doi:10.1016/S0012-365X(02)00758-6.

Computational Hardness of Certifying Bounds on Constrained PCA Problems

Afonso S. Bandeira
Dept. of Mathematics, ETH Zurich, Switzerland
https://people.math.ethz.ch/~abandeira/
bandeira@math.ethz.ch

Dmitriy Kunisky
Dept. of Mathematics, Courant Institute of Mathematical Sciences, New York University, USA
http://www.kunisky.com/
kunisky@cims.nyu.edu

Alexander S. Wein
Dept. of Mathematics, Courant Institute of Mathematical Sciences, New York University, USA
https://cims.nyu.edu/~aw128/
awein@cims.nyu.edu

Abstract

Given a random $n \times n$ symmetric matrix $\boldsymbol{W}$ drawn from the Gaussian orthogonal ensemble (GOE), we consider the problem of certifying an upper bound on the maximum value of the quadratic form $\boldsymbol{x}^\top \boldsymbol{W} \boldsymbol{x}$ over all vectors $\boldsymbol{x}$ in a constraint set $\mathcal{S} \subset \mathbb{R}^n$. For a certain class of normalized constraint sets we show that, conditional on a certain complexity-theoretic conjecture, no polynomial-time algorithm can certify a better upper bound than the largest eigenvalue of $\boldsymbol{W}$. A notable special case included in our results is the hypercube $\mathcal{S} = \{\pm 1/\sqrt{n}\}^n$, which corresponds to the problem of certifying bounds on the Hamiltonian of the Sherrington-Kirkpatrick spin glass model from statistical physics. Our results suggest a striking gap between optimization and certification for this problem.

Our proof proceeds in two steps. First, we give a reduction from the detection problem in the *negatively-spiked Wishart model* to the above certification problem. We then give evidence that this Wishart detection problem is computationally hard below the classical spectral threshold, by showing that no low-degree polynomial can (in expectation) distinguish the spiked and unspiked models. This method for predicting computational thresholds was proposed in a sequence of recent works on the sum-of-squares hierarchy, and is conjectured to be correct for a large class of problems. Our proof can be seen as constructing a distribution over symmetric matrices that appears computationally indistinguishable from the GOE, yet is supported on matrices whose maximum quadratic form over $\boldsymbol{x} \in \mathcal{S}$ is much larger than that of a GOE matrix.

2012 ACM Subject Classification Theory of computation → Computational complexity and cryptography

Keywords and phrases Certification, Sherrington-Kirkpatrick model, spiked Wishart model, low-degree likelihood ratio

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.78

Funding *Afonso S. Bandeira*: Part of this work was done while with the Courant Institute of Mathematical Sciences and the Center for Data Science at New York University and supported by NSF grants DMS-1712730, DMS-1719545, and by a grant from the Sloan Foundation.
Dmitriy Kunisky: Partially supported by NSF grants DMS-1712730 and DMS-1719545.
Alexander S. Wein: Partially supported by NSF grant DMS-1712730 and by the Simons Collaboration on Algorithms and Geometry.

Acknowledgements We thank Andrea Montanari and Samuel B. Hopkins for insightful discussions.

1 Introduction

An important phenomenon in the study of the computational aspects of random problems is the appearance of *statistical-to-computational gaps*, wherein a problem may be solved by an inefficient algorithm – typically a brute-force search – but empirical evidence, heuristic formal calculations, and negative results for classes of powerful algorithms all suggest that the same problem cannot be solved by any algorithm running in polynomial time. Many examples of this phenomenon arise from Bayesian estimation tasks, in which the goal is to recover a planted *signal* from noisy observations. Bayesian problems exhibiting statistical-to-computational gaps in certain regimes include graph problems such as community detection [16], estimation for models of structured matrices and tensors [42, 30], statistical problems arising from imaging and microscopy tasks [53, 10], and many others. A different family of examples comes from random optimization problems that are *signal-free*, where there is no "planted" structure to recover; rather, the task is simply to optimize a random objective function as effectively as possible. Notable instances of problems of this kind that exhibit statistical-to-computational gaps include finding a large clique in a random graph [33], finding a submatrix of large entries of a random matrix [25], or finding an approximate solution to a random constraint satisfaction problem [1].

In this paper, we study a problem from the latter class, namely the problem of maximizing the quadratic form $\boldsymbol{x}^\top \boldsymbol{W}\boldsymbol{x}$ over a constraint set $\boldsymbol{x} \in \mathcal{S} \subset \mathbb{R}^n$, where $\boldsymbol{W}$ is a random matrix drawn from the Gaussian orthogonal ensemble,[1] $\boldsymbol{W} \sim \mathsf{GOE}(n)$. Unlike previous works that have studied whether an efficient algorithm can *optimize* and find $\boldsymbol{x} = \boldsymbol{x}(\boldsymbol{W})$ that achieves a large objective value, we study whether an efficient algorithm can *certify* an upper bound on the objective over all $\boldsymbol{x} \in \mathcal{S}$. In the notable case of the *Sherrington-Kirkpatrick (SK) Hamiltonian* [59, 49], where $\mathcal{S} = \{\pm 1/\sqrt{n}\}^n$, while there is an efficient algorithm believed to optimize arbitrarily close to the true maximum [46], we give evidence – based on the *low-degree likelihood ratio* recently studied in the sum-of-squares literature [9, 31, 29, 28], which we describe in Section 2.4 – that there is *no* efficient algorithm to certify an upper bound that improves on a simple spectral certificate. Thus, the certification task for this problem appears to exhibit a statistical-to-computational gap, while the optimization task does not.

1.1 Computational tasks in random optimization problems

To formalize the above discussion, consider a generic random optimization problem:

$$\begin{array}{ll} \text{maximize} & f_\omega(\boldsymbol{x}) \\ \text{subject to} & \boldsymbol{x} \in \mathcal{S} \\ \text{where} & \omega \sim \mathbb{P}, \end{array} \tag{1}$$

for $\mathbb{P}$ a probability distribution over some set of problem instances Ω. We will contrast two important computational tasks in this setting. The first, most obvious task is that of *optimization*, producing an algorithm that computes $\mathsf{alg}_{\mathsf{opt}} : \Omega \to \mathcal{S}$ such that $f_\omega(\mathsf{alg}_{\mathsf{opt}}(\omega))$ is as large as possible (say, in expectation, or with high probability as the size of the problem diverges).

Another task is that of *certification*, producing instead an algorithm that computes a scalar $\mathsf{alg}_{\mathsf{cert}} : \Omega \to \mathbb{R}$, such that *for all* $\omega \in \Omega$ and all $\boldsymbol{x} \in \mathcal{S}$ we have $f_\omega(\boldsymbol{x}) \le \mathsf{alg}_{\mathsf{cert}}(\omega)$. The main challenge specific to certification is that $\mathsf{alg}_{\mathsf{cert}}$ must produce a valid upper bound on

[1] Gaussian orthogonal ensemble (GOE): $\boldsymbol{W}$ is symmetric with $W_{ij} = W_{ji} \sim \mathcal{N}(0, 1/n)$ for $i \neq j$ and $W_{ii} \sim \mathcal{N}(0, 2/n)$, all independent.

f_ω for *every* possible instance $\omega \in \Omega$, no matter how unlikely ω is to occur under $\mathbb{P}$. Subject to this requirement, we seek to minimize $\mathsf{alg}_{\mathsf{cert}}(\omega)$ (again, in a suitable probabilistic sense when $\omega \sim \mathbb{P}$). *Convex relaxations* are a common approach to certification, where $\mathcal{S}$ is relaxed to a convex superset $\mathcal{S}' \supset \mathcal{S}$ over which it is possible to optimize exactly and efficiently using convex optimization. Often, such algorithms admit an alternative interpretation of *proving* a bound on $f_\omega(\boldsymbol{x})$ in some limited proof system (see, e.g., [27] for such discussion of sum-of-squares algorithms).

If $\boldsymbol{x}^\star = \boldsymbol{x}^\star(\omega)$ is the true maximizer of f_ω, then for any pair of optimization and certification algorithms as above, we have

$$f_\omega(\mathsf{alg}_{\mathsf{opt}}(\omega)) \leq f_\omega(\boldsymbol{x}^\star) \leq \mathsf{alg}_{\mathsf{cert}}(\omega). \tag{2}$$

Thus, in the case of a maximization problem, optimization algorithms approximate the true value $f_\omega(\boldsymbol{x}^\star)$ from below, while certification algorithms approximate it from above. We are then interested in how tight either inequality can be for random problems of growing dimension. Of course, we can achieve "perfect" optimization and certification (equality on either side of (2)) by exhaustive search over all $\boldsymbol{x} \in \mathcal{S}$, but we are interested in whether this is still possible when we restrict our attention to computationally-efficient algorithms.

To make these definitions concrete, we review an instance of each type of algorithm for the problem of optimizing the Sherrington-Kirkpatrick Hamiltonian.

▶ **Example 1.1.** The "SK problem" is the random optimization problem

$$\begin{array}{ll} \text{maximize} & \boldsymbol{x}^\top \boldsymbol{W} \boldsymbol{x} \\ \text{subject to} & \boldsymbol{x} \in \{\pm 1/\sqrt{n}\}^n \\ \text{where} & \boldsymbol{W} \sim \mathsf{GOE}(n). \end{array} \tag{3}$$

Here, two related *spectral algorithms* give simple examples of algorithms for both optimization and certification. For certification, writing $\lambda_{\max}$ for the largest eigenvalue of $\boldsymbol{W}$, we may use the bound

$$\boldsymbol{x}^\top \boldsymbol{W} \boldsymbol{x} \leq \lambda_{\max} \cdot \|\boldsymbol{x}\|^2 = \lambda_{\max} \approx 2 \tag{4}$$

for all $\boldsymbol{x} \in \{\pm 1/\sqrt{n}\}^n$, whereby $\lambda_{\max}$ is a certifiable upper bound on (3). From classical random matrix theory (see, e.g., [5]), it is known that $\lambda_{\max} \approx 2$ as $n \to \infty$.

For optimization, for $\boldsymbol{v}_{\max}$ the eigenvector of $\lambda_{\max}$, we may take $\boldsymbol{x} = \boldsymbol{x}(\boldsymbol{W}) := \mathrm{sgn}(\boldsymbol{v}_{\max})/\sqrt{n}$ where sgn denotes the $\{\pm 1\}$-valued sign function, applied entrywise. The vector $\boldsymbol{v}_{\max}$ is distributed as an uniform random unit vector in $\mathbb{R}^n$, so the quality of this solution may be computed as

$$\boldsymbol{x}^\top \boldsymbol{W} \boldsymbol{x} = \lambda_{\max} \cdot \langle \boldsymbol{x}, \boldsymbol{v}_{\max} \rangle^2 + O\left(\frac{1}{\sqrt{n}}\right) = \lambda_{\max} \cdot \frac{\|\boldsymbol{v}_{\max}\|_1^2}{n} + O\left(\frac{1}{\sqrt{n}}\right) \approx \frac{4}{\pi} \approx 1.2732 \tag{5}$$

with high probability as $n \to \infty$. (The error in the first equation is obtained as $\sum_i \lambda_i \langle \boldsymbol{v}_i, \boldsymbol{x} \rangle^2 \approx \frac{1}{n} \mathrm{Tr}(\boldsymbol{W})(1 - \langle \boldsymbol{v}_{\max}, \boldsymbol{x} \rangle^2)$, where the sum is over all eigenvectors $\boldsymbol{v}_i$ except $\boldsymbol{v}_{\max}$. This analysis appeared in [3], an early rigorous mathematical work on the SK model.)

On the other hand, deep results of statistical physics imply that the true optimal value approaches

$$\boldsymbol{x}^{\star\top} \boldsymbol{W} \boldsymbol{x}^\star \approx 2\mathsf{P}_* \approx 1.5264 \tag{6}$$

as $n \to \infty$, where the constant P_* is expressed via the celebrated Parisi formula for the free energy of the SK model [50, 49, 62]. The approximate value we give above was estimated with numerical experiments in previous works (see, e.g., [51, 15]).

Thus, neither for optimization nor for certification does the naive spectral algorithm achieve the optimal value, which suggests the question: can more sophisticated algorithms improve on the spectral algorithm for either task? For optimization, the recent result of [46] implies, assuming a widely-believed conjecture from statistical physics, that for any $\varepsilon > 0$ there exists a polynomial-time optimization algorithm achieving with high probability a value of $2\mathsf{P}_* - \varepsilon$ on the SK problem.[2] On the other hand, there are few results addressing certification in the SK problem. The only previous work we are aware of in this direction is [48], where a simple semidefinite programming relaxation (which coincides with degree-2 sum-of-squares) is shown to achieve the same value as the spectral certificate (4). More recently (after the initial appearance of this paper) the same was shown for degree-4 sum-of-squares [39, 45].

1.2 Our contributions

The main result of this paper, which we now state informally, gives formal evidence that for the SK certification problem, the simple spectral certificate (4) is optimal among efficient algorithms. See Corollary 3.8 for the precise statement.

▶ **Theorem 1.2** (Informal). *Conditional on the correctness of the low-degree likelihood ratio method (see Section 2.4), for any $\varepsilon > 0$, there is no polynomial-time algorithm that certifies the upper bound $2 - \varepsilon$ on the SK problem* (3) *with probability $1 - o(1)$ as $n \to \infty$.*

In fact, we expect that there is not even a subexponential-time algorithm; see Remark 3.6. Theorem 1.2 reveals a striking gap between optimization and certification: it is possible to efficiently give a tight lower bound on the maximum objective value by exhibiting a solution $\boldsymbol{x}$, but it seems impossible to efficiently give a tight upper bound. In other words, an algorithm can efficiently find a near-optimal solution, but cannot be sure that it has done so. The same result also holds for a wide variety of constraints other than $\boldsymbol{x} \in \{\pm 1/\sqrt{n}\}^n$ (see Corollary 3.8). Due to the high-dimensional setting of the problem, we expect the value of an optimal certification algorithm to concentrate tightly; thus we expect Theorem 1.2 to still hold if $1 - o(1)$ is replaced by any positive constant.

Our result has important consequences for convex programming. A natural approach for optimizing the SK problem (3) would be to use a convex programming relaxation such as a semidefinite program based on the sum-of-squares hierarchy [60, 52, 41] (see [55] for a survey). Such a method would relax the constraints of the SK problem to weaker ones for which the associated optimization problem can be solved efficiently. One can either hope that the relaxation is *tight* and gives a valid solution $\boldsymbol{x} \in \{\pm 1/\sqrt{n}\}^n$ (with high probability), or use a *rounding* procedure to extract a valid solution from the relaxation. The optimal value of any convex relaxation of (3) provides an upper bound on the optimal value of (3) and therefore gives a certification algorithm. Thus Theorem 1.2 implies that (conditional on the correctness of the low-degree likelihood ratio method) no polynomial-time convex relaxation of (3) can have value $\leq 2 - \varepsilon$ (resolving a question posed by [32]) and in particular cannot be tight.[3] As a result, we expect that natural relax-and-round approaches for optimization should fail to find a solution of value close to $2\mathsf{P}_*$. This would suggest a fundamental weakness of convex programs: even the most powerful convex programs (such as sum-of-squares relaxations)

[2] The work [46] builds on that of [2, 61], and these works taken together formalize the heuristic idea from statistical physics that optimization is tractable for certain optimization problems exhibiting *full replica symmetry breaking*.

[3] Our results suggest that $\Omega(n^{1-o(1)})$ rounds of sum-of-squares are required to certify a value $2 - \varepsilon$; see Remark 3.6.

seem to fail to optimize (3), even though other methods succeed (namely, the message-passing algorithm of [46]).[4] An explanation for this suboptimality is that convex relaxations are actually solving a fundamentally harder problem: certification.

1.3 Related work

A related example of an optimization–certification gap comes from random constraint satisfaction problems (CSPs).

▶ **Example 1.3.** In random MAX-3SAT, the decision variable is a boolean vector $\boldsymbol{x} \in \{0,1\}^n$, and the optimization task is to maximize the number of satisfied *clauses* $C_1, \ldots, C_m$, each of which is a boolean expression of the form $C_i = a_{i_1} \vee a_{i_2} \vee a_{i_3}$ where each a_{i_j} is chosen uniformly among the x_i and their boolean negations. Let $s_{\boldsymbol{C}}(\boldsymbol{x})$ denote the number of clauses satisfied by $\boldsymbol{x}$.

If $m/n \to \infty$ as $n \to \infty$, the optimal value $\max_{\boldsymbol{x}} s_{\boldsymbol{C}}(\boldsymbol{x})$ is $(7/8 + o(1))m$ with probability $1 - o(1)$ [11]. This is achieved by the trivial optimization algorithm that chooses a uniformly random assignment $\boldsymbol{x}$. On the other hand, sum-of-squares lower bounds suggest that it is hard to certify even $s_{\boldsymbol{C}}(\boldsymbol{x}) < m$ unless $m \gg n^{3/2}$ [26, 58, 38]. Along similar lines, a well-known conjecture of Feige asserts this cannot be certified (by *any* certification algorithm) in polynomial time for m/n an arbitrarily large constant [22].

As in the SK problem, there is an efficient algorithm for near-perfect optimization, while there does not seem to be such an algorithm for near-perfect certification. However, here the optimization algorithm is trivial (a random guess), so arguably a more natural optimization task would be to achieve the best possible advantage over random guessing, assessing the quality of a solution on a finer scale.

Prior work has used *sum-of-squares lower bounds* to argue for hardness of certification in problems such as random CSPs [26, 58, 38], planted clique [19, 44, 9], tensor injective norm [30, 29], graph coloring [8], community detection in hypergraphs [37], and others. These results prove that the sum-of-squares hierarchy (at some degree) fails to certify. If sum-of-squares fails at every constant degree (e.g., [9, 38, 29]), this suggests that all polynomial-time algorithms should also fail. In our case, it appears difficult to prove such sum-of-squares lower bounds for the SK problem, although recent work (appearing after the initial version of this paper) has shown lower bounds at degree 4 [39, 45]. We instead take a new approach based on a related heuristic for computational hardness, which we explain in Section 2.4. One advantage of this approach over sum-of-squares is that it is substantially simpler. Perhaps the prior work that is closest to our approach is [63], which also gives a reduction from a hypothesis testing problem to a certification problem.

Overview of techniques

The proof of Theorem 1.2 has two parts. First, we give a reduction from hypothesis testing in the *negatively-spiked Wishart model* [34, 6, 7, 54] to the SK certification problem. We then use a method introduced in the sum-of-squares literature based on the *low-degree likelihood ratio* [9, 31, 29, 28] to give formal evidence that detection in that negatively-spiked Wishart model is computationally hard.

[4] In contrast, simple rounded convex relaxations are believed to approximate many similar problems optimally in the worst-case (rather than average-case) setting [36].

In the spiked Wishart model, we observe either N i.i.d. samples $\boldsymbol{y}_1, \dots, \boldsymbol{y}_N \sim \mathcal{N}(0, \boldsymbol{I}_n)$, or N i.i.d. samples $\boldsymbol{y}_1, \dots, \boldsymbol{y}_N \sim \mathcal{N}(0, \boldsymbol{I}_n + \beta \boldsymbol{x}\boldsymbol{x}^\top)$ where the "spike" $\boldsymbol{x} \in \{\pm 1/\sqrt{n}\}^n$ is a uniformly random hypercube vector, and $\beta \in [-1, \infty)$. The goal is to distinguish between these two cases with probability $1 - o(1)$ as $n \to \infty$. In the *negatively-spiked* ($\beta < 0$) case with $\beta \approx -1$, this task amounts to deciding whether there is a hypercube vector $\boldsymbol{x} \in \{\pm 1/\sqrt{n}\}^n$ that is nearly orthogonal to all of the samples $\boldsymbol{y}_i$. When $N = \Theta(n)$, a simple spectral method succeeds when $\beta^2 > n/N$ [6, 7], and we expect the problem to be computationally hard when $\beta^2 < n/N$.

Let us now intuitively explain the relation between the negatively-spiked Wishart model and the SK certification problem. Suppose we want to certify that

$$\mathsf{SK}(\boldsymbol{W}) := \max_{\boldsymbol{x} \in \{\pm 1/\sqrt{n}\}^n} \boldsymbol{x}^\top \boldsymbol{W} \boldsymbol{x} \leq 2 - \varepsilon$$

where $\boldsymbol{W} \sim \mathsf{GOE}(n)$, for some small constant $\varepsilon > 0$. Since the eigenvalues of $\boldsymbol{W}$ approximately follow the semicircle distribution on $[-2, 2]$ [64], we need to certify that the top δn-dimensional eigenspace of $\boldsymbol{W}$ does not (approximately) contain a hypercube vector, for a small $\delta > 0$ depending on ε. In particular, we need to distinguish a uniformly random δn-dimensional subspace (the distribution of the actual top eigenspace of $\boldsymbol{W} \sim \mathsf{GOE}(n)$) from a δn-dimensional subspace that contains a hypercube vector. Equivalently, taking orthogonal complements, we need to distinguish a uniformly random $(1-\delta)n$-dimensional subspace from a $(1-\delta)n$-dimensional subspace that is orthogonal to a hypercube vector. This is the problem of detection in the negatively-spiked Wishart model with $\beta \approx -1$ and $N = (1-\delta)n$, and these parameters lie in the "hard regime" $\beta^2 < n/N$.

Formally, we construct a distribution $\mathcal{D}(n)$ over $n \times n$ symmetric matrices with $\mathsf{SK}(\boldsymbol{W}) \geq 2 - \varepsilon/2$ when $\boldsymbol{W} \sim \mathcal{D}(n)$. This $\mathcal{D}(n)$ also has the property that, conditional on the hardness of the above detection problem, it is computationally hard to distinguish $\boldsymbol{W} \sim \mathcal{D}(n)$ from $\boldsymbol{W} \sim \mathsf{GOE}(n)$. The existence of such $\mathcal{D}(n)$ implies hardness of certification for the SK problem, because if an algorithm could certify that $\mathsf{SK}(\boldsymbol{W}) \leq 2 - \varepsilon$ when $\boldsymbol{W} \sim \mathsf{GOE}(n)$, then it could distinguish $\mathcal{D}(n)$ from $\mathsf{GOE}(n)$.

Borrowing terminology from [65, 66], we refer to this idea of "planting" a hidden solution (in our case, a hypercube vector $\boldsymbol{x}$) in such a way that it is difficult to detect, as *quiet planting*[5]. Our quiet planting scheme $\mathcal{D}(n)$ draws $\boldsymbol{W} \sim \mathsf{GOE}(n)$ and then rotates the top eigenspace of $\boldsymbol{W}$ to align with a random hypercube vector $\boldsymbol{x}$, while leaving the eigenvalues of $\boldsymbol{W}$ unchanged. (The more straightforward planting scheme, $\boldsymbol{W} + (2 - \varepsilon/2)\boldsymbol{x}\boldsymbol{x}^\top$ with $\boldsymbol{W} \sim \mathsf{GOE}(n)$, is not quiet because it changes the largest eigenvalue of $\boldsymbol{W}$ [23].) The question of how to design optimal quiet planting schemes in general remains an interesting open problem.

The final ingredient in our proof is to give formal evidence (in the form of the low-degree likelihood ratio) that detection in the spiked Wishart model is computationally hard below the spectral threshold. This consists of a calculation involving the projection of the likelihood ratio (between the "null" and "planted" distributions) onto the subspace of low-degree polynomials. This method suggests that the correct strategy for quiet planting is to match the low-degree moments of the distributions $\mathcal{D}(n)$ and $\mathsf{GOE}(n)$. We discuss the details of this method further in Section 2.4.

Our results on hardness in the spiked Wishart model may be of independent interest: our low-degree calculations suggest that, for a large class of spike priors, no polynomial-time algorithm can successfully distinguish the spiked and unspiked models below the classical

[5] However, our notion of quiet planting is not quite the same as that of [65, 66].

spectral threshold [6, 7], both in the negatively-spiked and positively-spiked regimes. (For positive spikes there was existing evidence for this based on failure of approximate message passing [24]; no such evidence was known for negative spikes.)

2 Background

2.1 Probability Theory

Our asymptotic notation (e.g., $O(\cdot), o(\cdot)$) always pertains to the limit $n \to \infty$. Parameters of the problem (e.g., $\beta, \gamma, \mathcal{X}, \mathcal{S}$) are held fixed as $n \to \infty$. Thus the constants hidden by $O(\cdot)$ and $o(\cdot)$ do not depend on n but may depend on the other parameters. When A_n is a sequence of events in probability spaces with measures $\mathbb{P}_n$, we say A_n holds *with high probability* if $\mathbb{P}_n[A_n] = 1 - o(1)$.

▶ **Definition 2.1.** *A real-valued random variable π with $\mathbb{E}[\pi] = 0$ is* subgaussian *if there exists $\sigma^2 \geq 0$ (the* variance proxy*) such that, for all $t \in \mathbb{R}$, $M(t) := \mathbb{E}[\exp(t\pi)] \leq \exp(\sigma^2 t^2/2)$.*

The name *subgaussian* refers to the fact that if $\pi \sim \mathcal{N}(0, \sigma^2)$, then $M(t) = \exp(\sigma^2 t^2/2)$. A random variable with law $\mathcal{N}(0, \sigma^2)$ is therefore subgaussian. Any bounded centered random variable is also subgaussian: if $\pi \in [a, b]$ almost surely, then π is subgaussian with $\sigma^2 = \frac{1}{4}(b-a)^2$ (see, e.g., [56]).

We next give some background facts from random matrix theory (see, e.g., [5]).

▶ **Definition 2.2.** *The* Gaussian orthogonal ensemble $\mathsf{GOE}(n)$ *is a probability distribution over symmetric matrices $\boldsymbol{W} \in \mathbb{R}^{n \times n}$, under which $W_{ii} \sim \mathcal{N}(0, 2/n)$ and $W_{ij} \sim \mathcal{N}(0, 1/n)$ when $i \neq j$, where the entries W_{ij} are independent for distinct pairs (i, j) with $i \leq j$.*

Our scaling of the entries of $\mathsf{GOE}(n)$ ensures a spectrum of constant width.

▶ **Proposition 2.3.** *Let $\boldsymbol{W}_n \sim \mathsf{GOE}(n)$. Then, almost surely, $\lambda_{\min}(\boldsymbol{W}_n) \to -2$ and $\lambda_{\max}(\boldsymbol{W}_n) \to 2$ as $n \to \infty$. In particular, for any $\varepsilon > 0$, $\|\boldsymbol{W}_n\| \leq 2 + \varepsilon$ with high probability. Also, the empirical distribution of eigenvalues of $\boldsymbol{W}_n$ converges weakly to a semicircle distribution supported on $[-2, 2]$.*

2.2 Constrained PCA

▶ **Definition 2.4.** *A* constraint set *is a sequence $\mathcal{S} = (\mathcal{S}_n)_{n \in \mathbb{N}}$ where $\mathcal{S}_n \subset \mathbb{R}^n$. The* constrained principal component analysis (PCA) problem with constraint set $\mathcal{S}$*, denoted* $\mathsf{PCA}(\mathcal{S})$*, is*

$$\begin{array}{ll} \textit{maximize} & \boldsymbol{x}^\top \boldsymbol{W} \boldsymbol{x} \\ \textit{subject to} & \boldsymbol{x} \in \mathcal{S}_n \\ \textit{where} & \boldsymbol{W} \sim \mathsf{GOE}(n). \end{array}$$

We will work only with constraint sets supported on vectors of approximately unit norm. General problems of this kind have been studied previously in, e.g., [20].

▶ **Example 2.5.** Problems that may be described in the constrained PCA framework include:

- the Sherrington-Kirkpatrick (SK) spin glass model: $\mathcal{S}_n = \{\pm 1/\sqrt{n}\}^n$ [59, 49],
- the Wigner sparse PCA null model: $\mathcal{S}_n = \{\boldsymbol{x} \in \mathbb{R}^n \ : \ \|\boldsymbol{x}\| = 1, \|\boldsymbol{x}\|_0 \leq \rho\}$ [17, 43],
- the spherical $2p$-spin spin glass model: $\mathcal{S}_{pn} = \{\boldsymbol{x}^{\otimes p} \ : \ \boldsymbol{x} \in \mathbb{R}^n, \|\boldsymbol{x}\| = 1\}$ [14, 13],
- the positive PCA null model: $\mathcal{S}_n = \{\boldsymbol{x} \in \mathbb{R}^n \ : \ x_i \geq 0, \|\boldsymbol{x}\| = 1\}$ [47].

Our results apply to the first two examples: the SK model, and sparse PCA when $\rho = \Theta(n)$.

▶ **Definition 2.6.** *Let f be a (randomized) algorithm*[6] *that takes a square matrix $\boldsymbol{W}$ as input and outputs a number $f(\boldsymbol{W}) \in \mathbb{R}$. We say that f* certifies *a value B on $\mathsf{PCA}(\mathcal{S})$ if*
1. *for* any *symmetric matrix $\boldsymbol{W} \in \mathbb{R}^{n \times n}$, $\max_{\boldsymbol{x} \in \mathcal{S}_n} \boldsymbol{x}^\top \boldsymbol{W} \boldsymbol{x} \le f(\boldsymbol{W})$, and*
2. *if $\boldsymbol{W}_n \sim \mathsf{GOE}(n)$ then $f(\boldsymbol{W}_n) \le B + o(1)$ with high probability.*

2.3 Spiked Wishart Models

▶ **Definition 2.7.** *A* normalized spike prior *is a sequence $\mathcal{X} = (\mathcal{X}_n)_{n \in \mathbb{N}}$ where $\mathcal{X}_n$ is a probability distribution over $\mathbb{R}^n$, such that if $\boldsymbol{x} \sim \mathcal{X}_n$ then $\|\boldsymbol{x}\| \to 1$ in probability as $n \to \infty$.*

▶ **Definition 2.8** (Spiked Wishart model)**.** *Let $\mathcal{X}$ be a normalized spike prior, let $\gamma > 0$, and let $\beta \in [-1, \infty)$. Let $N = \lceil n/\gamma \rceil$. We define two probability distributions over $(\mathbb{R}^n)^N$:*
1. *Under $\mathbb{Q}$, the* null *model, draw $\boldsymbol{y}_i \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_n)$ independently for $i \in [N]$.*
2. *Under $\mathbb{P}$, the* planted *model, draw $\boldsymbol{x} \sim \mathcal{X}_n$. If $\beta\|\boldsymbol{x}\|^2 \ge -1$, then draw $\boldsymbol{y}_i \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_n + \beta \boldsymbol{x}\boldsymbol{x}^\top)$ independently for $i \in [N]$. Otherwise, draw $\boldsymbol{y}_i \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_n)$ independently for $i \in [N]$.*

Taken together, $\mathbb{P}$ and $\mathbb{Q}$ form the spiked Wishart model $(\mathbb{P}, \mathbb{Q}) =: \mathsf{Wishart}(n, \gamma, \beta, \mathcal{X})$*. For fixed γ and β we denote the sequence $(\mathsf{Wishart}(n, \gamma, \beta, \mathcal{X}))_{n \in \mathbb{N}}$ by $\mathsf{Wishart}(\gamma, \beta, \mathcal{X})$.*

Several remarks on this definition are in order. First, we make the explicit choice $N = \lceil n/\gamma \rceil$ for concreteness, but our results apply to any choice of $N = N(n)$ for which $n/N \to \gamma$ as $n \to \infty$. Second, often the Wishart model is described in terms of the distribution of the sample covariance matrix $\frac{1}{N}\sum_{i=1}^N \boldsymbol{y}_i \boldsymbol{y}_i^\top$. We instead work directly with the samples $\boldsymbol{y}_i$ so as not to restrict ourselves to algorithms that only use the sample covariance matrix. (This modification only makes our results on computational hardness of detection more general.) Finally, the definition of $\mathbb{P}$ has two cases to ensure that the covariance matrix $\boldsymbol{I}_n + \beta \boldsymbol{x}\boldsymbol{x}^\top$ is positive semidefinite. We will work in the setting $\beta > -1$ where the first case ($\beta\|\boldsymbol{x}\|^2 \ge -1$) occurs with high probability.

We consider the algorithmic task of distinguishing between $\mathbb{P}$ and $\mathbb{Q}$ in the following sense.

▶ **Definition 2.9.** *For sequences of distributions $\mathbb{P} = (\mathbb{P}_n)_{n \in \mathbb{N}}$ and $\mathbb{Q} = (\mathbb{Q}_n)_{n \in \mathbb{N}}$ over measurable spaces $(\Omega_n, \mathcal{F}_n)_{n \in \mathbb{N}}$, an algorithm $f_n : \Omega_n \to \{0, 1\}$ achieves* strong detection *between $\mathbb{P}$ and $\mathbb{Q}$ if*

$$\mathbb{Q}_n[f_n(\boldsymbol{y}) = 0] = 1 - o(1) \qquad \text{and} \qquad \mathbb{P}_n[f_n(\boldsymbol{y}) = 1] = 1 - o(1).$$

The celebrated BBP transition [6] implies a spectral algorithm for strong detection in the spiked Wishart model whenever $\beta^2 > \gamma$.

▶ **Theorem 2.10** ([6, 7])**.** *Let $\mathcal{X}$ be any normalized spike prior. If $\beta^2 > \gamma$ then there exists a polynomial-time algorithm for strong detection in $\mathsf{Wishart}(\gamma, \beta, \mathcal{X})$.*

The algorithm thresholds the largest eigenvalue (if $\beta > 0$) or smallest eigenvalue (if $\beta < 0$) of the sample covariance matrix $\frac{1}{N}\sum_{i=1}^N \boldsymbol{y}_i \boldsymbol{y}_i^\top$. This eigenvalue converges almost surely to a limiting value which is different under $\mathbb{P}$ and $\mathbb{Q}$.

[6] We allow f to be randomized; i.e., it may use randomness in its computations, but the output B must be an upper bound almost surely. We do not expect certification algorithms to require randomness, but it may be convenient, e.g., to randomly initialize an iterative optimization procedure.

We will give evidence that (see Corollary 3.5) if $\mathcal{X}$ has i.i.d. subgaussian entries with suitable scaling, then no polynomial-time algorithm achieves strong detection below the BBP threshold ($\beta^2 < \gamma$). Exponential-time strong detection is possible below the BBP threshold for some priors, such as i.i.d. Rademacher when $\beta < -0.84$ [54]. Very sparse priors with $\boldsymbol{x}$ supported on $O(\sqrt{n})$ entries give rise to the *sparse PCA* regime, where polynomial-time strong detection is possible below the BBP threshold [35, 4, 18]; our results will not apply in this setting (although see [29, 21] for some related work that addresses this regime).

2.4 The Low-Degree Likelihood Ratio

Inspired by the sum-of-squares hierarchy (e.g., [60, 52, 41]) and in particular the pseudo-calibration approach [9], recent works [31, 29, 28] have proposed a strikingly simple method for predicting computational hardness of Bayesian inference problems. This method recovers widely-conjectured computational thresholds for high-dimensional inference problems such as planted clique [9, 28], community detection [31, 28], sparse PCA [21], tensor PCA [29, 28, 40], and the spiked Wigner matrix model [40]. We now give an overview of this method (see also [40] for a survey).

Consider the problem of distinguishing two simple hypotheses $\mathbb{P}_n$ and $\mathbb{Q}_n$ which are probability distributions on some domain $\Omega_n = \mathbb{R}^{d(n)}$ (where typically the dimension $d(n)$ grows with n). One example is the spiked Wishart model $\mathsf{Wishart}(\gamma, \beta, \mathcal{X})$ for some fixed choice of the parameters $\beta, \gamma, \mathcal{X}$. The idea is to take low-degree polynomials as a proxy for polynomial-time algorithms and consider whether there are such polynomials that can distinguish $\mathbb{P}_n$ from $\mathbb{Q}_n$.

▶ **Definition 2.11.** *Let $D : \mathbb{N} \to \mathbb{N}$. We say that distinguishing $\mathbb{P}_n$ from $\mathbb{Q}_n$ is $D(n)$-low-degree easy if there exists a sequence of nonzero polynomials $f_n \in \mathbb{R}[y_1, \ldots, y_{d(n)}]$ with $\deg f_n \leq D(n)$ such that*

$$\lim_{n\to\infty} \frac{\mathbb{E}_{\boldsymbol{y}\sim\mathbb{P}_n} f_n(\boldsymbol{y})}{\sqrt{\mathbb{E}_{\boldsymbol{y}\sim\mathbb{Q}_n} f_n(\boldsymbol{y})^2}} = +\infty, \tag{7}$$

and $D(n)$-low-degree hard otherwise.

We view $\mathbb{Q}_n$ as the "null" distribution, which is often i.i.d. Gaussian (as in the Wishart model) or i.i.d. Rademacher (± 1-valued). $\mathbb{Q}_n$ induces an inner product on L^2 functions $f : \Omega_n \to \mathbb{R}$ given by $\langle f, g\rangle_{L^2(\mathbb{Q}_n)} = \mathbb{E}_{\boldsymbol{y}\sim\mathbb{Q}_n}[f(\boldsymbol{y})g(\boldsymbol{y})]$, and a norm $\|f\|^2_{L^2(\mathbb{Q}_n)} = \langle f, f\rangle_{L^2(\mathbb{Q}_n)}$. For $D \in \mathbb{N}$, let $\mathbb{R}[\boldsymbol{y}]_{\leq D}$ denote the polynomials $\Omega_n \to \mathbb{R}$ of degree at most D. For $f : \Omega_n \to \mathbb{R}$, let $f^{\leq D}$ denote the orthogonal projection (with respect to $\langle\cdot,\cdot\rangle_{L^2(\mathbb{Q}_n)}$) of f onto $\mathbb{R}[\boldsymbol{y}]_{\leq D}$. The following relates low-degree hardness to the *low-degree likelihood ratio*.

▶ **Theorem 2.12** ([31])**.** *Let $\mathbb{P}_n$ and $\mathbb{Q}_n$ be probability distributions on Ω_n for each $n \in \mathbb{N}$. Suppose $\mathbb{P}_n$ is absolutely continuous with respect to $\mathbb{Q}_n$, so that the likelihood ratio $L_n = \frac{d\mathbb{P}_n}{d\mathbb{Q}_n}$ is defined. Then*

$$\max_{f\in\mathbb{R}[\boldsymbol{y}]_{\leq D}\setminus\{0\}} \frac{\mathbb{E}_{\boldsymbol{y}\sim\mathbb{P}_n} f(\boldsymbol{y})}{\sqrt{\mathbb{E}_{\boldsymbol{y}\sim\mathbb{Q}_n} f(\boldsymbol{y})^2}} = \|L_n^{\leq D}\|_{L^2(\mathbb{Q}_n)}. \tag{8}$$

Proof. The objective can be rewritten as $\langle f, L_n\rangle_{L^2(\mathbb{Q}_n)}/\|f\|_{L^2(\mathbb{Q}_n)}$, so by basic Hilbert space theory, the maximum is attained by taking $f = L_n^{\leq D}$. ◀

▶ **Corollary 2.13.** *In the setting of Theorem 2.12,*

1. *if $\|L_n^{\leq D(n)}\|_{L^2(\mathbb{Q}_n)} = O(1)$, then distinguishing $\mathbb{P}_n$ from $\mathbb{Q}_n$ is $D(n)$-low-degree hard;*
2. *if $\|L_n^{\leq D(n)}\|_{L^2(\mathbb{Q}_n)} = \omega(1)$, then distinguishing $\mathbb{P}_n$ from $\mathbb{Q}_n$ is $D(n)$-low-degree easy.*

We take $O(\log n)$-degree polynomials $\Omega_n \to \mathbb{R}$ as a proxy for functions computable in polynomial-time. One justification for this is that many polynomial-time algorithms compute the leading eigenvalue of a matrix $\boldsymbol{M}$ whose entries are constant-degree polynomials in the data; in fact, there is formal evidence that such *low-degree spectral methods* are as powerful as the sum-of-squares hierarchy [29]. Typically, $O(\log n)$ rounds of power iteration are sufficient to compute the leading eigenvalue accurately, which amounts to evaluating the $O(\log n)$-degree polynomial $\mathrm{Tr}(\boldsymbol{M}^{2q})$ for some $q = O(\log n)$. This argument can be made formal: if $\|L_n^{\leq D}\|_{L^2(\mathbb{Q}_n)} = O(1)$ then all low-degree spectral methods must fail in a certain sense [40]. This motivates the following informal conjecture, which is based on [31, 29, 28], particularly Conjecture 2.2.4 of [28].

▶ **Conjecture 2.14** (Informal). *For "nice" distributions $\mathbb{P}_n$ and $\mathbb{Q}_n$, if distinguishing $\mathbb{P}_n$ and $\mathbb{Q}_n$ is $\log^{1+\Omega(1)}(n)$-low-degree hard, then there is no randomized polynomial-time algorithm for strong detection between $\mathbb{P}$ and $\mathbb{Q}$.*

This conjecture is useful because the norm of the low-degree likelihood ratio, $\|L_n^{\leq D}\|_{L^2(\mathbb{Q}_n)}$, can be computed (or at least bounded) for various distributions such as the stochastic block model [31] and the spiked tensor model [29, 28]. More generally, Hypothesis 2.1.5 of [28] conjectures that degree-D polynomials are a proxy for time-$n^{\widetilde{\Theta}(D)}$ algorithms.

▶ Remark 2.15. We do not expect the converse of Conjecture 2.14 to hold. If detection is $O(\log n)$-low-degree easy then we expect an $n^{O(\log n)}$-time algorithm but not necessarily a polynomial-time algorithm, because not every $O(\log n)$-degree polynomial can be evaluated in polynomial time.

Conjecture 2.14 is informal in that we do not specify what is meant by "nice" distributions. See Conjecture 2.2.4 of [28] for a precise variant of Conjecture 2.14; however, this variant uses the more refined notion of *coordinate degree* and so does not apply to the calculations we will perform. Roughly speaking, "nice" distributions $\mathbb{P}$ and $\mathbb{Q}$ should satisfy the following:

1. $\mathbb{Q}$ should be a product distribution, e.g., i.i.d. Gaussian or i.i.d. Rademacher;
2. $\mathbb{P}$ should be sufficiently symmetric with respect to permutations of its coordinates; and
3. we should be able to add a small amount of noise to $\mathbb{P}$, ruling out distributions with brittle algebraic structure (such as random satisfiable instances of XOR-SAT, which can be identified using Gaussian elimination [12]).

We refer the reader to [31, 28, 40] for further details and evidence in favor of Conjecture 2.14.

3 Main Results

3.1 Spiked Wishart Models

We now study the low-degree hardness of the spiked Wishart model. The following technical definitions will be important to specify the priors to which our results apply.

▶ **Definition 3.1.** *Let $\beta \in (-1, \infty)$ and let $\mathcal{X}$ be a normalized spike prior. We say that $\mathcal{X}$ is* β-good *if when $\boldsymbol{x} \sim \mathcal{X}_n$ then $\beta\|\boldsymbol{x}\|^2 > -1$ almost surely.*

We consider spike priors having i.i.d. entries, and will sometimes need to slightly modify the spike prior to ensure that it is β-good and has bounded norm.

▶ **Definition 3.2.** *Let π be a probability distribution over $\mathbb{R}$ such that $\mathbb{E}[\pi] = 0$ and $\mathbb{E}[\pi^2] = 1$. Let $\mathsf{iid}(\pi/\sqrt{n})$ denote the normalized spike prior $\mathcal{X} = (\mathcal{X}_n)$ that draws each entry of $\boldsymbol{x}$ independently from $\frac{1}{\sqrt{n}}\pi$. (We do not allow π to depend on n.)*

▶ **Definition 3.3.** *For a normalized spike prior $\mathcal{X}$, let the β-truncation* $\mathrm{trunc}_\beta(\mathcal{X})$ *of $\mathcal{X}$ denote the following normalized spike prior. To sample $\boldsymbol{x}$ from* $(\mathrm{trunc}_\beta(\mathcal{X}))_n$*, first sample $\boldsymbol{x}' \sim \mathcal{X}_n$. Then, let $\boldsymbol{x} = \boldsymbol{x}'$ if $\beta\|\boldsymbol{x}'\|^2 > -1$ and $\|\boldsymbol{x}'\|^2 \le 2$, and let $\boldsymbol{x} = \boldsymbol{0}$ otherwise.*

If $\beta > -1$ then since $\mathcal{X}$ is normalized ($\|\boldsymbol{x}'\| \to 1$ in probability), the first case of Definition 3.3 occurs with high probability. The upper bound $\|\boldsymbol{x}'\| \le 2$ is for technical convenience, and the constant 2 is not essential. Note also that the β-truncation of an i.i.d. prior is no longer i.i.d.

▶ **Theorem 3.4.** *Fix constants $\gamma > 0$ and $\beta > -1$.*

1. *Suppose $\beta^2 < \gamma$. Let $\mathcal{X} = \mathrm{trunc}_\beta(\mathsf{iid}(\pi/\sqrt{n}))$ where π is subgaussian with $\mathbb{E}[\pi] = 0$ and $\mathbb{E}[\pi^2] = 1$. For any $D = o(n/\log n)$, distinguishing $\mathbb{P}_n$ from $\mathbb{Q}_n$ is $D(n)$-low-degree hard.*
2. *Suppose $\beta^2 > \gamma$. Let $\mathcal{X} = \mathsf{iid}(\pi/\sqrt{n})$ be β-good with π symmetric about zero, $\mathbb{E}[\pi] = 0$, and $\mathbb{E}[\pi^2] = 1$. For any $D = \omega(1)$, distinguishing $\mathbb{P}_n$ from $\mathbb{Q}_n$ is $D(n)$-low-degree easy.*

We prove Theorem 3.4 in Section 5. Part 1 of Theorem 3.4, combined with Conjecture 2.14, suggests that for i.i.d. subgaussian priors, strong detection is hard below the BBP threshold.

▶ **Corollary 3.5.** *Suppose Conjecture 2.14 holds for the spiked Wishart model. Fix constants $\gamma > 0$ and $\beta > -1$. Let π be subgaussian with $\mathbb{E}[\pi] = 0$ and $\mathbb{E}[\pi^2] = 1$. Let $\mathcal{X}$ be either $\mathsf{iid}(\pi/\sqrt{n})$ or $\mathrm{trunc}_\beta(\mathsf{iid}(\pi/\sqrt{n}))$. If $\beta^2 < \gamma$, then there is no randomized polynomial-time algorithm for strong detection in $\mathsf{Wishart}(\gamma, \beta, \mathcal{X})$.*

Proof. The case $\mathcal{X} = \mathrm{trunc}_\beta(\mathsf{iid}(\pi/\sqrt{n}))$ follows immediately from Part 1 of Theorem 3.4. If strong detection is impossible for $\mathcal{X} = \mathrm{trunc}_\beta(\mathsf{iid}(\pi/\sqrt{n}))$, then strong detection is also impossible for $\mathcal{X} = \mathsf{iid}(\pi/\sqrt{n})$, as these two spike priors differ with probability $o(1)$ (under the natural coupling). ◀

▶ Remark 3.6. We make some technical remarks regarding Theorem 3.4 and Corollary 3.5.

1. Even if Conjecture 2.14 does not hold, note that Theorem 3.4 still implies unconditional lower bounds against low-degree polynomials in the sense of Definition 2.11.
2. Part 2 of Theorem 3.4 serves only to check that we do not predict computational hardness when $\beta^2 > \gamma$ (as polynomial-time strong detection is possible in this regime; see Theorem 2.10). The assumption that π is symmetric about zero should not be essential.
3. In Part 1 of Theorem 3.4 and in Corollary 3.5, the requirement that $\mathcal{X}$ be a β-truncated i.i.d. prior can be relaxed. We only require that $\mathcal{X}$ is the β-truncation of a normalized prior admitting a *local Chernoff bound* (see Definition 5.11).
4. Part 1 of Theorem 3.4 holds for any $D = o(n/\log n)$, much larger than the $D = \log^{1+\Omega(1)}(n)$ required by Conjecture 2.14. Since Hypothesis 2.1.5 of [28] conjectures that degree-D polynomials are a proxy for $n^{\widetilde{\Theta}(D)}$-time algorithms [28], this suggests that the conclusion of Corollary 3.5 also holds for $2^{n^{1-\delta}}$-time algorithms, for any $\delta > 0$. In other words, strong detection requires nearly-exponential time.

3.2 Constrained PCA

The following result gives a reduction from strong detection in the spiked Wishart model to certification in the constrained PCA problem.

▶ **Theorem 3.7.** *Let $\mathcal{S}$ be a constraint set and let $\mathcal{X}$ be a normalized spike prior such that if $\boldsymbol{x} \sim \mathcal{X}_n$ then $\boldsymbol{x} \in \mathcal{S}_n$ with high probability. Suppose there exists $\varepsilon > 0$ and a randomized polynomial-time algorithm that certifies the value $2 - \varepsilon$ on $\mathsf{PCA}(\mathcal{S})$. Then there exist $\gamma > 1$ and $\beta \in (-1, 0)$ such that there is a randomized polynomial-time algorithm for strong detection in $\mathsf{Wishart}(\gamma, \beta, \mathcal{X})$.*

We give the proof in Section 4. Note for the parameters γ, β above, $\beta^2 < \gamma$ (the "hard regime").

▶ **Corollary 3.8.** *Suppose Conjecture 2.14 holds for the spiked Wishart model. Let π be subgaussian with $\mathbb{E}[\pi] = 0$ and $\mathbb{E}[\pi^2] = 1$. Let $\mathcal{S}$ be a constraint set such that, if $\boldsymbol{x} \sim \mathsf{iid}(\pi/\sqrt{n})$, then $\boldsymbol{x} \in \mathcal{S}_n$ with high probability. Then, for any $\varepsilon > 0$, there is no randomized polynomial-time algorithm to certify the value $2 - \varepsilon$ on $\mathsf{PCA}(\mathcal{S})$.*

Proof. The result is immediate from Theorem 3.4 and Theorem 3.7. ◀

In particular, we obtain the hardness of improving on the spectral certificate in the SK model.

▶ **Corollary 3.9.** *If Conjecture 2.14 holds for the spiked Wishart model, then for any $\varepsilon > 0$, there is no randomized polynomial-time algorithm to certify the value $2 - \varepsilon$ on the SK problem* (3)*.*

Proof. Apply Corollary 3.8 with π having the Rademacher distribution and $\mathcal{S}_n = \{\pm 1/\sqrt{n}\}^n$. ◀

4 Proof of Reduction from Spiked Wishart to Constrained PCA

Our proof will rely on the following crucial invariance property of $\mathsf{GOE}(n)$.

▶ **Proposition 4.1.** *For any orthogonal matrix $\boldsymbol{Q} \in \mathcal{O}(n)$, if $\boldsymbol{W} \sim \mathsf{GOE}(n)$, then the law of $\boldsymbol{Q}\boldsymbol{W}\boldsymbol{Q}^\top$ is also $\mathsf{GOE}(n)$.*

Proof of Theorem 3.7. Let $\mathcal{S}$ be a constraint set and let $\mathcal{X}$ be a normalized spike prior such that, if $\boldsymbol{x} \sim \mathcal{X}_n$, then $\boldsymbol{x} \in \mathcal{S}_n$ with high probability. Suppose that for some $\varepsilon > 0$ there is a randomized polynomial-time algorithm f that certifies the value $2 - \varepsilon$ on $\mathsf{PCA}(\mathcal{S})$. We will show that this implies that there is a polynomial-time algorithm for strong detection in $\mathsf{Wishart}(\gamma, \beta, \mathcal{X})$ with certain parameters $\gamma > 1$ and $\beta \in (-1, 0)$ (depending on ε). Note that these parameters lie in the "hard" regime $\beta^2 < \gamma$.

Our algorithm for detection in the Wishart model is as follows. Fix $\gamma > 1$, to be chosen later. Since $n/N \to \gamma$ we have $n > N$ (for sufficiently large n). Given samples $\boldsymbol{y}_1, \dots, \boldsymbol{y}_N \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_n + \beta \boldsymbol{x}\boldsymbol{x}^\top)$, let $V = \mathrm{span}\{\boldsymbol{y}_1, \dots, \boldsymbol{y}_N\} \subseteq \mathbb{R}^n$ and let $V^\perp$ be its orthogonal complement. We sample $\boldsymbol{W} \in \mathbb{R}^{n \times n}$ having the distribution $\mathsf{GOE}(n)$ conditioned on the event that the span of the top $n - N$ eigenvectors of $\boldsymbol{W}$ is $V^\perp$. Concretely, we can obtain a sample in the following way. Let $\boldsymbol{v}_1, \dots, \boldsymbol{v}_N$ be a uniformly random orthonormal basis for V and let $\boldsymbol{v}_{N+1}, \dots, \boldsymbol{v}_n$ be a uniformly random orthonormal basis for $V^\perp$. Sample $\boldsymbol{W}' \sim \mathsf{GOE}(n)$ and let $\lambda_1 < \cdots < \lambda_n$ be the eigenvalues of $\boldsymbol{W}'$. Then, let $\boldsymbol{W} := \sum_{i=1}^n \lambda_i \boldsymbol{v}_i \boldsymbol{v}_i^\top$. Finally, run the certification algorithm f for $\mathsf{PCA}(\mathcal{S})$ on $\boldsymbol{W}$. The detection algorithm $\widetilde{f} : (\mathbb{R}^n)^N \to \{0, 1\}$ then thresholds $f(\boldsymbol{W})$:

$$\widetilde{f}(\boldsymbol{y}_1, \dots, \boldsymbol{y}_N) = \begin{cases} 0 \text{ (report } \mathbb{Q}_n) & \text{if } f(\boldsymbol{W}) \le 2 - \varepsilon/2, \\ 1 \text{ (report } \mathbb{P}_n) & \text{if } f(\boldsymbol{W}) > 2 - \varepsilon/2. \end{cases} \tag{9}$$

We now prove that $\widetilde{f}$ indeed achieves strong detection in $\mathsf{Wishart}(\gamma, \beta, \mathcal{X})$. First, if the samples $\boldsymbol{y}_i$ are drawn from the null model $\mathbb{Q}_n$, then V is a uniformly random N-dimensional subspace of $\mathbb{R}^n$, so by Proposition 4.1 the law of $\boldsymbol{W}$ constructed above is $\mathsf{GOE}(n)$. Thus $f(\boldsymbol{W}) \le 2 - \varepsilon/2$ with high probability by assumption, and therefore $\widetilde{f}(\boldsymbol{y}_1, \dots, \boldsymbol{y}_N) = 0$ with high probability, i.e., our algorithm correctly reports that the samples were drawn from the null model.

Next, suppose the samples $\boldsymbol{y}_i$ are drawn from the planted model $\mathbb{P}_n$ with planted spike $\boldsymbol{x} \sim \mathcal{X}_n$. We will choose $\gamma > 1$ and $\beta \in (-1, 0)$ so that $\boldsymbol{x}^\top \boldsymbol{W} \boldsymbol{x} \geq 2 - \varepsilon/3$ with high probability. Since $\boldsymbol{x} \in \mathcal{S}_n$ with high probability, this would imply $f(\boldsymbol{W}) \geq 2 - \varepsilon/3$, so we will have $\widetilde{f}(\boldsymbol{y}_1, \dots, \boldsymbol{y}_N) = 1$ with high probability, i.e., our algorithm will correctly report that the samples were drawn from the planted model.

It remains to show that $\boldsymbol{x}^\top \boldsymbol{W} \boldsymbol{x} \geq 2 - \varepsilon/3$. Let $\lambda_1 < \cdots < \lambda_n$ be the eigenvalues of $\boldsymbol{W}$ and let $\boldsymbol{v}_1, \dots, \boldsymbol{v}_n$ be the corresponding (unit-norm) eigenvectors. By Proposition 2.3, with high probability, for all $i \in [n]$, $\lambda_i \in [-2 - o(1), 2 + o(1)]$. Furthermore, by the semicircle law [64], with high probability, $\lambda_{N+1} \geq 2 - g(\gamma)$ where $g(\gamma) > 0$ is a function satisfying $g(\gamma) \to 0$ as $\gamma \to 1^+$ (recalling that $n/N \to \gamma$). Letting $\|\boldsymbol{x}\|_V$ denote the norm of the orthogonal projection of $\boldsymbol{x}$ onto V, we have, with high probability,

$$\begin{aligned}
\boldsymbol{x}^\top \boldsymbol{W} \boldsymbol{x} &= \boldsymbol{x}^\top \left(\sum_{i=1}^n \lambda_i \boldsymbol{v}_i \boldsymbol{v}_i^\top \right) \boldsymbol{x} \\
&= \sum_{i=1}^n \lambda_i \langle \boldsymbol{x}, \boldsymbol{v}_i \rangle^2 \\
&\geq \lambda_1 \|\boldsymbol{x}\|_V^2 + \lambda_{N+1} \|\boldsymbol{x}\|_{V^\perp}^2 \\
&\geq (-2 - o(1)) \|\boldsymbol{x}\|_V^2 + (2 - g(\gamma))(\|\boldsymbol{x}\|^2 - \|\boldsymbol{x}\|_V^2) \\
&= (2 - g(\gamma)) \|\boldsymbol{x}\|^2 + (-4 + g(\gamma) - o(1)) \|\boldsymbol{x}\|_V^2 \\
&\geq 2 - g(\gamma) - 4\|\boldsymbol{x}\|_V^2 - o(1). \qquad (10)
\end{aligned}$$

Thus we need to upper bound $\|\boldsymbol{x}\|_V^2$. Let $\boldsymbol{P}_V$ denote the orthogonal projection matrix onto V. Since V is the span of $\{\boldsymbol{y}_1, \dots, \boldsymbol{y}_N\}$, we have $\boldsymbol{P}_V \preceq \frac{1}{\mu} \boldsymbol{Y}$ where

$$\boldsymbol{Y} = \frac{1}{N} \sum_{i=1}^N \boldsymbol{y}_i \boldsymbol{y}_i^\top$$

and μ is the smallest nonzero eigenvalue of $\boldsymbol{Y}$. (Here $\preceq$ denotes Loewner order.) Since $\boldsymbol{Y}$ is a spiked Wishart matrix, it follows from Theorem 1.2 of [7] that its smallest nonzero eigenvalue converges almost surely to $(1 - \sqrt{\gamma})^2$ as $n \to \infty$. Thus we have $\mu = (1 - \sqrt{\gamma})^2 + o(1)$. Therefore,

$$\|\boldsymbol{x}\|_V^2 = \|\boldsymbol{P}_V \boldsymbol{x}\|^2 = \boldsymbol{x}^\top \boldsymbol{P}_V \boldsymbol{x} \leq \frac{1}{\mu} \boldsymbol{x}^\top \boldsymbol{Y} \boldsymbol{x} = \frac{1}{\mu N} \sum_{i=1}^N \langle \boldsymbol{x}, \boldsymbol{y}_i \rangle^2.$$

We have $\boldsymbol{y}_i \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_n + \beta \boldsymbol{x}\boldsymbol{x}^\top)$ and so $\langle \boldsymbol{x}, \boldsymbol{y}_i \rangle \sim \mathcal{N}(0, \boldsymbol{x}^\top (\boldsymbol{I}_n + \beta \boldsymbol{x}\boldsymbol{x}^\top) \boldsymbol{x}) = \mathcal{N}(0, \|\boldsymbol{x}\|^2 + \beta \|\boldsymbol{x}\|^4)$. Therefore, letting $a_N^2 = \sum_{i=1}^N g_i^2$ for g_i i.i.d. standard gaussian random variables, so that a_N^2 has the χ^2 distribution with N degrees of freedom, we have conditional on $\boldsymbol{x}$ the distributional equality

$$\boldsymbol{x}^\top \boldsymbol{Y} \boldsymbol{x} \stackrel{(d)}{=} (\|\boldsymbol{x}\|^2 + \beta \|\boldsymbol{x}\|^4) \frac{a_N^2}{N}.$$

Standard concentration inequalities imply $a_N^2 / N \in [1 - o(1), 1 + o(1)]$ with high probability, and therefore $\boldsymbol{x}^\top \boldsymbol{Y} \boldsymbol{x} = 1 + \beta + o(1)$ with high probability. Thus, with high probability, we find

$$\|\boldsymbol{x}\|_V^2 = \frac{1 + \beta}{(1 - \sqrt{\gamma})^2} + o(1). \qquad (11)$$

Finally, we choose $\gamma > 1$ close enough to 1 so that $g(\gamma) \le \varepsilon/8$. By (11), we can also choose $\beta \in (-1, 0)$ close enough to -1 so that $\|\boldsymbol{x}\|_V^2 \le \varepsilon/32$ with high probability. Combining these, from (10) it follows that $\boldsymbol{x}^\top \boldsymbol{W}\boldsymbol{x} \ge 2 - \varepsilon/4 - o(1) \ge 2 - \varepsilon/3$ with high probability, completing the proof. ◀

▶ Remark 4.2. We remark that we have ignored issues of numerical precision by assuming a model of computation where eigendecomposition computations can be done exactly in polynomial time. However, we believe all the operations we have used are stable, so that our reduction should also hold for weaker models of computation. (In particular, if we want to compute polynomially-many bits of precision of the $\mathsf{PCA}(\mathcal{S})$ instance, this should require only polynomially-many bits of precision in the eigendecomposition computation.)

5 Proofs for Spiked Wishart Models

5.1 Preliminaries

Spiked Wishart model statistics

The following formulae pertaining to the spiked Wishart model are derived in [54]. (Recall that in the spiked Wishart model, the parameter N is determined by n and γ as $N = \lceil n/\gamma \rceil$.)

▶ **Proposition 5.1.** *Suppose $\gamma > 0$, $\beta \in [-1, \infty)$, and $\mathcal{X}$ is a β-good normalized spike prior. Then, the likelihood ratio of the null and planted probability distributions of Definition 2.8 is*

$$L_{n,\gamma,\beta,\mathcal{X}}(\boldsymbol{y}_1, \dots, \boldsymbol{y}_N) := \frac{d\mathbb{P}_n}{d\mathbb{Q}_n}(\boldsymbol{y}_1, \dots, \boldsymbol{y}_N) = \mathop{\mathbb{E}}_{\boldsymbol{x} \sim \mathcal{X}_n} \left[\left(1 + \beta \|\boldsymbol{x}\|^2\right)^{-N/2} \prod_{i=1}^{N} \exp\left(\frac{1}{2} \frac{\beta}{1 + \beta\|\boldsymbol{x}\|^2} \langle \boldsymbol{x}, \boldsymbol{y}_i \rangle^2 \right) \right]. \quad (12)$$

If furthermore $\|\boldsymbol{x}\|^2 < 1/|\beta|$ almost surely when $\boldsymbol{x} \sim \mathcal{X}_n$, then the second moment of the likelihood ratio is given by

$$\mathop{\mathbb{E}}_{\boldsymbol{y} \sim \mathbb{Q}_n} \left(L_{n,\gamma,\beta,\mathcal{X}}(\boldsymbol{y}_1, \dots, \boldsymbol{y}_N)\right)^2 = \mathop{\mathbb{E}}_{\boldsymbol{x}^1, \boldsymbol{x}^2 \sim \mathcal{X}_n} \left[(1 - \beta^2 \langle \boldsymbol{x}^1, \boldsymbol{x}^2 \rangle^2)^{-N/2} \right] \quad (13)$$

where $\boldsymbol{x}^1, \boldsymbol{x}^2$ are drawn independently from $\mathcal{X}_n$.

Hermite polynomials

We recall the classical one-dimensional Hermite polynomials.

▶ **Definition 5.2.** *The polynomials $h_k \in \mathbb{R}[x]$ for $k \ge 0$ are defined by the recursion*

$$h_0(x) = 1,$$
$$h_{k+1}(x) = x h_k(x) - h_k'(x),$$

and we define normalized versions

$$\widehat{h}_k(x) = \frac{1}{\sqrt{k!}} h_k(x).$$

▶ **Proposition 5.3.** *The $\widehat{h}_k$ are an orthonormal polynomial system for the standard Gaussian measure:*

$$\mathop{\mathbb{E}}_{g \sim \mathcal{N}(0,1)} \left[\widehat{h}_k(g) \widehat{h}_\ell(g) \right] = \delta_{k\ell}.$$

Similarly, we define the product Hermite polynomials. It is helpful to first define some notations for vectors of indices, which will also be used in the later derivations.

▶ **Definition 5.4.** *Let* $\mathbb{N} = \{n \in \mathbb{Z} : n \geq 0\}$. *For* $\boldsymbol{\alpha} \in \mathbb{N}^n$ *and* $\boldsymbol{x} \in \mathbb{R}^n$, *let*

$$|\boldsymbol{\alpha}| := \sum_{i=1}^{n} \alpha_i,$$
$$\boldsymbol{\alpha}! := \prod_{i=1}^{n} \alpha_i!,$$
$$\boldsymbol{x}^{\boldsymbol{\alpha}} := \prod_{i=1}^{n} x_i^{\alpha_i}.$$

▶ **Definition 5.5.** *For* $\boldsymbol{\alpha} \in \mathbb{N}^n$ *and* $\boldsymbol{x} \in \mathbb{R}^n$,

$$H_{\boldsymbol{\alpha}}(\boldsymbol{x}) := \prod_{i=1}^{n} h_{\alpha_i}(x_i),$$
$$\widehat{H}_{\boldsymbol{\alpha}}(\boldsymbol{x}) := \prod_{i=1}^{n} \widehat{h}_{\alpha_i}(x_i) = \frac{1}{\sqrt{\boldsymbol{\alpha}!}} H_{\boldsymbol{\alpha}}(\boldsymbol{x}).$$

▶ **Proposition 5.6.** *The* $\widehat{H}_{\boldsymbol{\alpha}}$ *are an orthonormal polynomial system for the product measure of* n *standard Gaussian measures:*

$$\mathop{\mathbb{E}}_{\boldsymbol{g} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_n)} \left[\widehat{H}_{\boldsymbol{\alpha}}(\boldsymbol{g}) \widehat{H}_{\boldsymbol{\beta}}(\boldsymbol{g}) \right] = \delta_{\boldsymbol{\alpha}\boldsymbol{\beta}}.$$

Combinatorics

We will also need the (ordinary) generating function of the central binomial coefficients.

▶ **Proposition 5.7.** *For any* $x \in \mathbb{R}$ *with* $|x| < \frac{1}{4}$,

$$(1 - 4x)^{-1/2} = \sum_{k \geq 0} \binom{2k}{k} x^k.$$

5.2 Norm of the Low-Degree Projection

In this section, we describe the formulas for the norm of the low-degree likelihood ratio in the spiked Wishart model, $\|L_{n,\gamma,\beta,\mathcal{X}}^{\leq D}\|_{L^2(\mathbb{Q}_n)}$. The full calculations are given in Appendix A.

The following result, the main technical one of this portion of the argument, computes the norm of the projection of the likelihood ratio onto a single Hermite polynomial.

▶ **Lemma 5.8.** *Let* $\boldsymbol{\alpha} \in (\mathbb{N}^n)^N$, *and let* $\boldsymbol{\alpha}_i \in \mathbb{N}^n$ *denote the* i*th component. Let* $|\boldsymbol{\alpha}| = \sum_{i=1}^{N} |\boldsymbol{\alpha}_i|$. *Suppose* $\gamma > 0$, $\beta \in [-1, \infty)$, *and* $\mathcal{X}$ *is a* β*-good normalized spike prior. Then,*

$$\langle L_{n,\gamma,\beta,\mathcal{X}}, \widehat{H}_{\boldsymbol{\alpha}} \rangle_{L^2(\mathbb{Q}_n)}^2 = \begin{cases} \beta^{|\boldsymbol{\alpha}|} \cdot \prod_{i=1}^{N} \frac{(|\boldsymbol{\alpha}_i|-1)!!^2}{\boldsymbol{\alpha}_i!} \cdot \left(\mathbb{E}_{\boldsymbol{x} \sim \mathcal{X}_n} \boldsymbol{x}^{\sum_{i=1}^{N} \boldsymbol{\alpha}_i} \right)^2 & \text{if } |\boldsymbol{\alpha}_i| \text{ even for all } i \in [N], \\ 0 & \text{otherwise,} \end{cases} \tag{14}$$

where when $\beta = 0$ *and* $\boldsymbol{\alpha} = \boldsymbol{0}$ *we interpret* $0^0 = 1$.

Note in particular that the quantity in question does not depend on the sign of β; thus the calculation of the norm of the low-degree projection of the likelihood ratio will not distinguish between the positively and negatively spiked Wishart models. Interestingly, in our proof, which involves generalized Hermite polynomials that form families of orthogonal polynomials with respect to Gaussian measures of different variances, this corresponds to the fact that an "umbral" analogue of the Hermite polynomials corresponding to a fictitious Gaussian measure with *negative* variance satisfies many of the same identities as the ordinary Hermite polynomials.

Combining these quantities, we may give a simple description of the norm of the low-degree projection of the likelihood ratio.

▶ **Lemma 5.9.** *Suppose $\gamma > 0$, $\beta \in [-1, \infty)$, and $\mathcal{X}$ is a β-good normalized spike prior. Define*

$$\varphi_N(x) := (1-4x)^{-N/2}, \tag{15}$$

$$\varphi_{N,k}(x) := \sum_{d=0}^{k} x^d \sum_{\substack{d_1,\dots,d_N \\ \sum d_i = d}} \prod_{i=1}^{N} \binom{2d_i}{d_i}, \tag{16}$$

so that $\varphi_{N,k}(x)$ is the Taylor series of φ_N around $x = 0$ truncated to degree k (as may be justified by Proposition 5.7). Then,

$$\|L^{\le D}_{n,\gamma,\beta,\mathcal{X}}\|^2_{L^2(\mathbb{Q}_n)} = \mathop{\mathbb{E}}_{\boldsymbol{x}^1,\boldsymbol{x}^2 \sim \mathcal{X}_n} \left[\varphi_{N,\lfloor D/2 \rfloor} \left(\frac{\beta^2 \langle \boldsymbol{x}^1, \boldsymbol{x}^2 \rangle^2}{4} \right) \right] \tag{17}$$

where $\boldsymbol{x}^1, \boldsymbol{x}^2$ are drawn independently from $\mathcal{X}$.

▶ Remark 5.10. The squared norm of the low-degree likelihood ratio (17) is closely related via Taylor expansion to the squared norm (or second moment) of the full likelihood ratio (13), which is recovered by taking $D \to \infty$ while n and N remain fixed.

5.3 Asymptotics as $n \to \infty$

In this section, we use the formula from Lemma 5.9 to prove Part 1 of Theorem 3.4 (the case $\beta^2 < \gamma$). The proof of Part 2 ($\beta^2 > \gamma$) is deferred to Appendix B.4.

The following concentration result is the key property that we require from the spike prior $\mathcal{X}$.

▶ **Definition 5.11.** *A normalized spike prior $\mathcal{X}$ admits a* local Chernoff bound *if for every $\eta > 0$ there exist $\delta > 0$ and $C > 0$ such that, for all n,*

$$\Pr\left\{ |\langle \boldsymbol{x}^1, \boldsymbol{x}^2 \rangle| \ge t \right\} \le C \exp\left(-\frac{1}{2}(1-\eta) n t^2 \right) \quad \text{for all } t \in [0, \delta] \tag{18}$$

where $\boldsymbol{x}^1, \boldsymbol{x}^2$ are drawn independently from $\mathcal{X}_n$.

▶ **Proposition 5.12.** *If π is subgaussian with $\mathbb{E}[\pi] = 0$ and $\mathbb{E}[\pi^2] = 1$ then $\mathsf{iid}(\pi/\sqrt{n})$ and $\mathrm{trunc}_\beta(\mathsf{iid}(\pi/\sqrt{n}))$ (for any $\beta > -1$) each admit a local Chernoff bound.*

We defer the proof to Appendix B.1.

Proof of Theorem 3.4 (Part 1). Let $\beta^2 < \gamma$. We decompose the norm of the low-degree likelihood ratio into two parts, which we will bound separately:

$$\|L_{n,\gamma,\beta,\mathcal{X}}^{\leq D}\|_{L^2(\mathbb{Q}_n)}^2 = \mathop{\mathbb{E}}_{\boldsymbol{x}^1,\boldsymbol{x}^2\sim\mathcal{X}_n}\left[\varphi_{N,\lfloor D/2\rfloor}\left(\frac{\beta^2}{4}\langle \boldsymbol{x}^1,\boldsymbol{x}^2\rangle^2\right)\right] = R_1 + R_2$$

where

$$R_1 := \mathop{\mathbb{E}}_{\boldsymbol{x}^1,\boldsymbol{x}^2\sim\mathcal{X}_n}\left[\mathbb{1}_{|\langle \boldsymbol{x}^1,\boldsymbol{x}^2\rangle|\leq\varepsilon}\,\varphi_{N,\lfloor D/2\rfloor}\left(\frac{\beta^2}{4}\langle \boldsymbol{x}^1,\boldsymbol{x}^2\rangle^2\right)\right],$$

$$R_2 := \mathop{\mathbb{E}}_{\boldsymbol{x}^1,\boldsymbol{x}^2\sim\mathcal{X}_n}\left[\mathbb{1}_{|\langle \boldsymbol{x}^1,\boldsymbol{x}^2\rangle|>\varepsilon}\,\varphi_{N,\lfloor D/2\rfloor}\left(\frac{\beta^2}{4}\langle \boldsymbol{x}^1,\boldsymbol{x}^2\rangle^2\right)\right].$$

Here $\varepsilon > 0$ is a small constant to be chosen later. We call R_1 the *small deviations* and call R_2 the *large deviations*.

The following two lemmas bound these two terms, respectively. First, we bound the large deviations.

▶ **Lemma 5.13** (Large Deviations). *Let $\beta^2 < \gamma$. Suppose $\mathcal{X}$ is a β-good normalized spike prior that admits a local Chernoff bound. Suppose that for any n, $\boldsymbol{x} \sim \mathcal{X}_n$ satisfies $\|\boldsymbol{x}\|^2 \leq 2$ almost surely. If $D = o(n/\log n)$ and $\varepsilon > 0$ is any constant, then $R_2 = o(1)$.*

We give a proof summary, with the full proof deferred to Appendix B.2. Since $\|\boldsymbol{x}^1\|^2 \leq 2$ and $\|\boldsymbol{x}^2\|^2 \leq 2$,

$$R_2 \leq \Pr\left\{|\langle \boldsymbol{x}^1,\boldsymbol{x}^2\rangle| > \varepsilon\right\}\varphi_{N,\lfloor D/2\rfloor}(\beta^2).$$

By the local Chernoff bound, $\Pr\left\{|\langle \boldsymbol{x}^1,\boldsymbol{x}^2\rangle| > \varepsilon\right\}$ decays exponentially in n. To complete the proof, we use elementary combinatorial bounds to control the polynomial expression (16) for $\varphi_{N,\lfloor D/2\rfloor}(\beta^2)$. Its growth is roughly of order $O(n^D)$, which is counteracted by the exponential decay of $\Pr\left\{|\langle \boldsymbol{x}^1,\boldsymbol{x}^2\rangle| > \varepsilon\right\}$ so long as $D = o(n/\log n)$.

Next, we bound the small deviations. For this part of the argument, it is irrelevant that the likelihood ratio is truncated to its low-degree component, and we essentially reuse an existing argument for the full likelihood ratio from [54].

▶ **Lemma 5.14** (Small Deviations). *Let $\beta^2 < \gamma$. Suppose $\mathcal{X}$ is a β-good normalized spike prior that admits a local Chernoff bound. Let $D = D(n)$ be any function of n. If $\varepsilon > 0$ is a sufficiently small constant then $R_1 = O(1)$.*

We again give a proof summary, with the full proof deferred to Appendix B.3. As mentioned above, unlike in the proof of Lemma 5.13, here we simply bound $\varphi_{N,\lfloor D/2\rfloor} \leq \varphi_N$ in the expression for R_1. To bound the resulting expression, we borrow an argument from [54]. This step crucially uses the local Chernoff bound, and amounts to showing that the exponential decay from the Chernoff bound sufficiently counteracts the exponential growth of the likelihood ratio term $\varphi_N(\frac{\beta^2}{4}\langle \boldsymbol{x}^1,\boldsymbol{x}^2\rangle^2)$ when $\langle \boldsymbol{x}^1,\boldsymbol{x}^2\rangle$ is small.

Combining Proposition 5.12 with Lemmas 5.13 and 5.14 completes the proof of Part 1 of Theorem 3.4. (The proof of Part 2 is deferred to Appendix B.4.) ◀

References

1 Dimitris Achlioptas and Amin Coja-Oghlan. Algorithmic barriers from phase transitions. In *Foundations of Computer Science, 2008. FOCS'08. IEEE 49th Annual IEEE Symposium on*, pages 793–802. IEEE, 2008.

2 Louigi Addario-Berry and Pascal Maillard. The algorithmic hardness threshold for continuous random energy models. *arXiv preprint*, 2018. arXiv:1810.05129.

3 Michael Aizenman, Joel L Lebowitz, and David Ruelle. Some rigorous results on the Sherrington-Kirkpatrick spin glass model. *Communications in mathematical physics*, 112(1):3–20, 1987.

4 Arash A Amini and Martin J Wainwright. High-dimensional analysis of semidefinite relaxations for sparse principal components. In *2008 IEEE International Symposium on Information Theory*, pages 2454–2458. IEEE, 2008.

5 Greg W Anderson, Alice Guionnet, and Ofer Zeitouni. An introduction to random matrices, 2010.

6 Jinho Baik, Gérard Ben Arous, and Sandrine Péché. Phase transition of the largest eigenvalue for nonnull complex sample covariance matrices. *The Annals of Probability*, 33(5):1643–1697, 2005.

7 Jinho Baik and Jack W Silverstein. Eigenvalues of large sample covariance matrices of spiked population models. *Journal of multivariate analysis*, 97(6):1382–1408, 2006.

8 Jess Banks, Robert Kleinberg, and Cristopher Moore. The Lovász Theta Function for Random Regular Graphs and Community Detection in the Hard Regime. *arXiv preprint*, 2017. arXiv:1705.01194.

9 Boaz Barak, Samuel B Hopkins, Jonathan Kelner, Pravesh Kothari, Ankur Moitra, and Aaron Potechin. A nearly tight sum-of-squares lower bound for the planted clique problem. In *Foundations of Computer Science (FOCS), 2016 IEEE 57th Annual Symposium on*, pages 428–437. IEEE, 2016.

10 Nicolas Boumal, Tamir Bendory, Roy R Lederman, and Amit Singer. Heterogeneous multireference alignment: A single pass approach. In *Information Sciences and Systems (CISS), 2018 52nd Annual Conference on*, pages 1–6. IEEE, 2018.

11 Amin Coja-Oghlan, Andreas Goerdt, and André Lanka. Strong refutation heuristics for random k-SAT. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 310–321. Springer, 2004.

12 Nadia Creignou and Hervé Daude. Satisfiability threshold for random XOR-CNF formulas. *Discrete Applied Mathematics*, 96:41–53, 1999.

13 A Crisanti, H Horner, and H-J Sommers. The spherical p-spin interaction spin-glass model. *Zeitschrift für Physik B Condensed Matter*, 92(2):257–271, 1993.

14 A Crisanti and HJ Sommers. The spherical p-spin interaction spin glass model: the statics. *Phys. B*, 87:341, 1992.

15 Andrea Crisanti and Tommaso Rizzo. Analysis of the infinity-replica symmetry breaking solution of the Sherrington-Kirkpatrick model. *Physical Review E*, 65(4):046137, 2002.

16 Aurelien Decelle, Florent Krzakala, Cristopher Moore, and Lenka Zdeborová. Asymptotic analysis of the stochastic block model for modular networks and its algorithmic applications. *Physical Review E*, 84(6):066106, 2011.

17 Yash Deshpande and Andrea Montanari. Information-theoretically optimal sparse PCA. *arXiv preprint*, 2014. arXiv:1402.2238.

18 Yash Deshpande and Andrea Montanari. Sparse PCA via covariance thresholding. In *Advances in Neural Information Processing Systems*, pages 334–342, 2014.

19 Yash Deshpande and Andrea Montanari. Improved sum-of-squares lower bounds for hidden clique and hidden submatrix problems. In *Conference on Learning Theory*, pages 523–562, 2015.

20 Yash Deshpande, Andrea Montanari, and Emile Richard. Cone-constrained principal component analysis. In *Advances in Neural Information Processing Systems*, pages 2717–2725, 2014.

21 Yunzi Ding, Dmitriy Kunisky, Alexander S Wein, and Afonso S Bandeira. Subexponential-Time Algorithms for Sparse PCA. *arXiv preprint*, 2019. arXiv:1907.11635.

22 Uriel Feige. Relations between average case complexity and approximation complexity. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 534–543. ACM, 2002.

23 Delphine Féral and Sandrine Péché. The largest eigenvalue of rank one deformation of large Wigner matrices. *Communications in mathematical physics*, 272(1):185–228, 2007.

24 Alyson K Fletcher and Sundeep Rangan. Iterative reconstruction of rank-one matrices in noise. *Information and Inference: A Journal of the IMA*, 7(3):531–562, 2018.

25 David Gamarnik and Quan Li. Finding a large submatrix of a Gaussian random matrix. *The Annals of Statistics*, 46(6A):2511–2561, 2018.

26 Dima Grigoriev. Linear lower bound on degrees of Positivstellensatz calculus proofs for the parity. *Theoretical Computer Science*, 259(1-2):613–622, 2001.

27 Dima Grigoriev and Nicolai Vorobjov. Complexity of Null-and Positivstellensatz proofs. *Annals of Pure and Applied Logic*, 113(1-3):153–160, 2001.

28 Samuel Hopkins. *Statistical Inference and the Sum of Squares Method*. PhD thesis, Cornell University, 2018.

29 Samuel B Hopkins, Pravesh K Kothari, Aaron Potechin, Prasad Raghavendra, Tselil Schramm, and David Steurer. The power of sum-of-squares for detecting hidden structures. In *Foundations of Computer Science (FOCS), 2017 IEEE 58th Annual Symposium on*, pages 720–731. IEEE, 2017.

30 Samuel B Hopkins, Jonathan Shi, and David Steurer. Tensor principal component analysis via sum-of-squares proofs. In *Conference on Learning Theory*, pages 956–1006, 2015.

31 Samuel B Hopkins and David Steurer. Bayesian estimation from few samples: community detection and related problems. *arXiv preprint*, 2017. `arXiv:1710.00264`.

32 Vishesh Jain, Frederic Koehler, and Andrej Risteski. Mean-field approximation, convex hierarchies, and the optimality of correlation rounding: a unified perspective. *arXiv preprint*, 2018. `arXiv:1808.07226`.

33 Mark Jerrum. Large cliques elude the Metropolis process. *Random Structures & Algorithms*, 3(4):347–359, 1992.

34 Iain M. Johnstone. On the distribution of the largest eigenvalue in principal components analysis. *The Annals of Statistics*, 29(2):295–327, 2001.

35 Iain M Johnstone and Arthur Yu Lu. Sparse principal components analysis. *Unpublished manuscript*, 7, 2004.

36 Subhash Khot, Guy Kindler, Elchanan Mossel, and Ryan O'Donnell. Optimal inapproximability results for MAX-CUT and other 2-variable CSPs? *SIAM Journal on Computing*, 37(1):319–357, 2007.

37 Chiheon Kim, Afonso S Bandeira, and Michel X Goemans. Community detection in hypergraphs, spiked tensor models, and Sum-of-Squares. In *Sampling Theory and Applications (SampTA), 2017 International Conference on*, pages 124–128. IEEE, 2017.

38 Pravesh K Kothari, Ryuhei Mori, Ryan O'Donnell, and David Witmer. Sum of squares lower bounds for refuting any CSP. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 132–145. ACM, 2017.

39 Dmitriy Kunisky and Afonso S Bandeira. A Tight Degree 4 Sum-of-Squares Lower Bound for the Sherrington-Kirkpatrick Hamiltonian. *arXiv preprint*, 2019. `arXiv:1907.11686`.

40 Dmitriy Kunisky, Alexander S Wein, and Afonso S Bandeira. Notes on Computational Hardness of Hypothesis Testing: Predictions using the Low-Degree Likelihood Ratio. *arXiv preprint*, 2019. `arXiv:1907.11636`.

41 Jean B Lasserre. An explicit exact SDP relaxation for nonlinear 0-1 programs. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 293–303. Springer, 2001.

42 Thibault Lesieur, Florent Krzakala, and Lenka Zdeborová. MMSE of probabilistic low-rank matrix estimation: Universality with respect to the output channel. *arXiv preprint*, 2015. `arXiv:1507.03857`.

43 Thibault Lesieur, Florent Krzakala, and Lenka Zdeborová. Phase transitions in sparse PCA. In *2015 IEEE International Symposium on Information Theory (ISIT)*, pages 1635–1639. IEEE, 2015.

44 Raghu Meka, Aaron Potechin, and Avi Wigderson. Sum-of-squares lower bounds for planted clique. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 87–96. ACM, 2015.

45 Sidhanth Mohanty, Prasad Raghavendra, and Jeff Xu. Lifting Sum-of-Squares Lower Bounds: Degree-2 to Degree-4, 2019. `arXiv:1911.01411`.

46 Andrea Montanari. Optimization of the Sherrington-Kirkpatrick Hamiltonian. *arXiv preprint*, 2018. `arXiv:1812.10897`.

47 Andrea Montanari and Emile Richard. Non-negative principal component analysis: Message passing algorithms and sharp asymptotics. *IEEE Transactions on Information Theory*, 62(3):1458–1484, 2016.

48 Andrea Montanari and Subhabrata Sen. Semidefinite programs on sparse random graphs and their application to community detection. *arXiv preprint*, 2015. `arXiv:1504.05910`.

49 Dmitry Panchenko. *The Sherrington-Kirkpatrick model*. Springer Science & Business Media, 2013.

50 Giorgio Parisi. Infinite number of order parameters for spin-glasses. *Physical Review Letters*, 43(23):1754, 1979.

51 Giorgio Parisi. A sequence of approximated solutions to the SK model for spin glasses. *Journal of Physics A: Mathematical and General*, 13(4):L115, 1980.

52 Pablo A Parrilo. *Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization*. PhD thesis, California Institute of Technology, 2000.

53 Amelia Perry, Alexander S Wein, Afonso S Bandeira, and Ankur Moitra. Message-Passing Algorithms for Synchronization Problems over Compact Groups. *Communications on Pure and Applied Mathematics*, 71(11):2275–2322, 2018.

54 Amelia Perry, Alexander S Wein, Afonso S Bandeira, and Ankur Moitra. Optimality and sub-optimality of PCA I: Spiked random matrix models. *The Annals of Statistics*, 46(5):2416–2451, 2018.

55 Prasad Raghavendra, Tselil Schramm, and David Steurer. High-dimensional estimation via sum-of-squares proofs. *arXiv preprint*, 2018. `arXiv:1807.11419`.

56 Philippe Rigollet and Jan-Christian Hütter. High-dimensional statistics. *Lecture notes*, 2018.

57 Steven Roman. *The umbral calculus*. Springer, 2005.

58 Grant Schoenebeck. Linear level Lasserre lower bounds for certain k-CSPs. In *2008 49th Annual IEEE Symposium on Foundations of Computer Science*, pages 593–602. IEEE, 2008.

59 David Sherrington and Scott Kirkpatrick. Solvable model of a spin-glass. *Physical review letters*, 35(26):1792, 1975.

60 Naum Z Shor. Class of global minimum bounds of polynomial functions. *Cybernetics*, 23(6):731–734, 1987.

61 Eliran Subag. Following the ground-states of full-RSB spherical spin glasses. *arXiv preprint*, 2018. `arXiv:1812.04588`.

62 Michel Talagrand. The Parisi formula. *Annals of mathematics*, pages 221–263, 2006.

63 Tengyao Wang, Quentin Berthet, and Yaniv Plan. Average-case hardness of RIP certification. In *Advances in Neural Information Processing Systems*, pages 3819–3827, 2016.

64 Eugene P Wigner. Characteristic vectors of bordered matrices with infinite dimensions I. In *The Collected Works of Eugene Paul Wigner*, pages 524–540. Springer, 1993.

65 Lenka Zdeborova and Florent Krzakala. Hiding quiet solutions in random constraint satisfaction problems. *Physical Review Letters*, 102(LA-UR-08-08090; LA-UR-08-8090), 2008.

66 Lenka Zdeborová and Florent Krzakala. Quiet planting in the locked constraint satisfaction problems. *SIAM Journal on Discrete Mathematics*, 25(2):750–770, 2011.

A Proofs for Computing the Low-Degree Likelihood Ratio

A.1 Generalized and Umbral Hermite Polynomials

We introduce some calculations with a useful generalization of the Hermite polynomials. While the usual Hermite polynomials are a family of orthogonal polynomials for the Gaussian measure with variance 1, and a straightforward generalization yields orthogonal polynomials for the Gaussian measure with any positive variance, we will use the surprising further generalization to fictitious Gaussian measures with *negative* variance, as described by the so-called *umbral calculus*. We follow the presentation of [57] (specifically, Section 2.1 of Chapter 4).

▶ **Definition A.1.** *For any $v \in \mathbb{R}$, the* Hermite polynomials with variance v *are defined by the recursion*

$$h_0(x; v) = 1, \tag{19}$$
$$h_{k+1}(x; v) = x h_k(x; v) - v \partial_x [h_k](x; v). \tag{20}$$

The next facts are useful for translating between different versions of the basic recursion and other properties of the Hermite polynomials.

▶ **Proposition A.2** (Differentiation Identity)**.** *For any $v, x \in \mathbb{R}$,*

$$\partial_x [h_k](x; v) = k h_{k-1}(x; v). \tag{21}$$

▶ **Proposition A.3** (Alternate Recursion)**.** *For any $v \in \mathbb{R}$, the Hermite polynomials are equivalently defined by the recursion*

$$h_0(x; v) = 1, \tag{22}$$
$$h_{k+1}(x; v) = x h_k(x; v) - v k h_{k-1}(x; v). \tag{23}$$

The following is yet another common way of defining the Hermite polynomials, in terms of the derivatives of the corresponding Gaussian density (or, in the negative variance case, a suitable generalization thereof).

▶ **Proposition A.4** (Rodrigues Formula)**.** *Let $v \in \mathbb{R}$ with $v \neq 0$. Then,*

$$\frac{d^k}{dx^k}\left[\exp\left(-\frac{1}{2v}x^2\right)\right] = (-v)^{-k} h_k(x; v) \exp\left(-\frac{1}{2v}x^2\right). \tag{24}$$

The next fact shows how the generalized Hermite polynomials transform under scaling.

▶ **Proposition A.5** (Scaling Identity)**.** *Let $v, w, x \in \mathbb{R}$, then*

$$h_k(wx; v) = w^k h_k\left(x; \frac{v}{w^2}\right). \tag{25}$$

Finally, the following is a generalized version of Gaussian integration by parts. We only provide the version of this identity for the standard Hermite polynomials, which is the only one we will use, but analogous statements hold for the generalized and umbral Hermite polynomials.

▶ **Proposition A.6** (Integration by Parts)**.** *Let $f \in \mathcal{C}^k(\mathbb{R})$ have $|f^{(i)}(x)| \leq e^{Cx}$ for all $i \in \{0, 1, \ldots, k\}$ and some $C > 0$. Then,*

$$\mathop{\mathbb{E}}_{g \sim \mathcal{N}(0,1)} [h_k(g; 1) f(g)] = \mathop{\mathbb{E}}_{g \sim \mathcal{N}(0,1)} \left[f^{(k)}(g)\right]. \tag{26}$$

While the above results are standard, we now give two results we will use in our calculation that do not seem to appear explicitly in the previous literature, although they are straightforward to obtain from the preceding facts. First, we will use the following slightly more general version of the Rodgrigues formula (Proposition A.4) in our calculations.

▶ **Proposition A.7** (Multidimensional Rodrigues Formula). *Let $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{\alpha} \in \mathbb{N}^n$, and $v \in \mathbb{R}$ with $v \neq 0$. Then,*

$$\partial_{\boldsymbol{y}}^{\boldsymbol{\alpha}} \left[\exp\left(-\frac{1}{2v} \langle \boldsymbol{x}, \boldsymbol{y} \rangle^2 \right) \right] = (-v)^{-|\boldsymbol{\alpha}|} \boldsymbol{x}^{\boldsymbol{\alpha}} h_{|\boldsymbol{\alpha}|}(\langle \boldsymbol{x}, \boldsymbol{y} \rangle; v) \exp\left(-\frac{1}{2v} \langle \boldsymbol{x}, \boldsymbol{y} \rangle^2 \right). \tag{27}$$

Proof. We proceed by induction on $|\boldsymbol{\alpha}|$. Clearly the result holds for $\boldsymbol{\alpha} = \mathbf{0}$. Suppose the result holds for all $|\boldsymbol{\alpha}'| \leq k$, and $|\boldsymbol{\alpha}| = k + 1 > 0$. Let $\boldsymbol{\alpha}'$ having $|\boldsymbol{\alpha}'| = k$ differ from $\boldsymbol{\alpha}$ only in coordinate i, so that $\alpha'_i = \alpha_i - 1$ and $\alpha'_j = \alpha_j$ for all $j \neq i$. Then,

$$\begin{aligned}
&\partial_{\boldsymbol{y}}^{\boldsymbol{\alpha}} \left[\exp\left(-\frac{1}{2v} \langle \boldsymbol{x}, \boldsymbol{y} \rangle^2 \right) \right] \\
&\quad = \partial_{y_i} \left[\partial_{\boldsymbol{y}}^{\boldsymbol{\alpha}'} \left[\exp\left(-\frac{1}{2v} \langle \boldsymbol{x}, \boldsymbol{y} \rangle^2 \right) \right] \right] \\
&\quad = (-v)^{-k} \boldsymbol{x}^{\boldsymbol{\alpha}'} \partial_{y_i} \left[h_k(\langle \boldsymbol{x}, \boldsymbol{y} \rangle; v) \exp\left(-\frac{1}{2v} \langle \boldsymbol{x}, \boldsymbol{y} \rangle^2 \right) \right] \\
&\quad = (-v)^{-k} \boldsymbol{x}^{\boldsymbol{\alpha}'} \left(x_i \partial_x[h_k](\langle \boldsymbol{x}, \boldsymbol{y} \rangle; v) - v^{-1} \langle \boldsymbol{x}, \boldsymbol{y} \rangle x_i h_k(\langle \boldsymbol{x}, \boldsymbol{y} \rangle; v) \right) \exp\left(-\frac{1}{2v} \langle \boldsymbol{x}, \boldsymbol{y} \rangle^2 \right) \\
&\quad = (-v)^{-(k+1)} \boldsymbol{x}^{\boldsymbol{\alpha}} \left(\langle \boldsymbol{x}, \boldsymbol{y} \rangle h_k(\langle \boldsymbol{x}, \boldsymbol{y} \rangle; v) - v \partial_x[h_k](\langle \boldsymbol{x}, \boldsymbol{y} \rangle; v) \right) \exp\left(-\frac{1}{2v} \langle \boldsymbol{x}, \boldsymbol{y} \rangle^2 \right) \\
&\quad = (-v)^{-(k+1)} \boldsymbol{x}^{\boldsymbol{\alpha}} h_{k+1}(\langle \boldsymbol{x}, \boldsymbol{y} \rangle; v) \exp\left(-\frac{1}{2v} \langle \boldsymbol{x}, \boldsymbol{y} \rangle^2 \right),
\end{aligned}$$

completing the proof. ◀

Second, we will need the following calculation evaluating the expectation of any Hermite polynomial under any centered Gaussian measure.

▶ **Proposition A.8** (Expectation Under Mismatched Variance). *Let $v \in \mathbb{R}$ and $k \geq 0$. Then,*

$$\mathbb{E}_{g \sim \mathcal{N}(0, \sigma^2)} \left[h_k(g; v) \right] = \begin{cases} 0 & \text{if } k \text{ odd,} \\ (k-1)!!(\sigma^2 - v)^{k/2} & \text{if } k \text{ even.} \end{cases}$$

Proof. The result for odd k holds since $h_k(\cdot\,; v)$ is an odd function for any $v \in \mathbb{R}$ in this case. For even k, we argue by induction on k. The result clearly holds for $k = 0$. If the result holds for a given k, then we may compute

$$\begin{aligned}
\mathbb{E}_{g \sim \mathcal{N}(0,\sigma^2)}[h_{k+2}(g; v)] &= \mathbb{E}_{g \sim \mathcal{N}(0,1)}[h_{k+2}(\sigma g)] \\
&= \sigma \mathbb{E}_{g \sim \mathcal{N}(0,1)}[g h_{k+1}(\sigma g; v)] - v(k+1) \mathbb{E}_{g \sim \mathcal{N}(0,1)}[h_k(\sigma g)] \\
&= \sigma^2 \mathbb{E}_{g \sim \mathcal{N}(0,1)}[\partial_x[h_{k+1}](\sigma g; v)] - v(k+1) \mathbb{E}_{g \sim \mathcal{N}(0,1)}[h_k(\sigma g)] \\
&= (k+1)\sigma^2 \mathbb{E}_{g \sim \mathcal{N}(0,1)}[h_k(\sigma g; v)] - v(k+1) \mathbb{E}_{g \sim \mathcal{N}(0,1)}[h_k(\sigma g)] \\
&= (k+1)(\sigma^2 - v) \mathbb{E}_{g \sim \mathcal{N}(0,1)}[h_k(\sigma g)] \\
&= (k+1)(\sigma^2 - v) \mathbb{E}_{g \sim \mathcal{N}(0,\sigma^2)}[h_k(g)]
\end{aligned}$$

completing the proof. ◀

We note two interesting features of this result. First, it generalizes two simple cases, on the one hand $v = \sigma^2$ where the expectation is zero unless $k = 0$, as may be seen from the orthogonality relations, and on the other $v = 0$ where it recovers the moments of a Gaussian measure. Second, the quantities appearing on the right-hand side formally resemble the moments of a Gaussian measure of suitable variance, but the formula in fact still holds for $\sigma^2 < v$, in which case case these quantities may be viewed as the moments of a fictitious Gaussian measure of negative variance (the same as inspired the umbral Hermite polynomials).

A.2 Individual Hermite Components of the Likelihood Ratio

Proof of Lemma 5.8. By Proposition A.6, we find

$$\begin{aligned}
&\langle L_{n,\gamma,\beta,\mathcal{X}}, \widehat{H}_{\boldsymbol{\alpha}} \rangle^2 \\
&= \frac{1}{\boldsymbol{\alpha}!} \left(\mathop{\mathbb{E}}_{\boldsymbol{y} \sim \mathbb{Q}_n} \partial_{\boldsymbol{y}}^{\boldsymbol{\alpha}} L_{n,\gamma,\beta,\mathcal{X}}(\boldsymbol{y}_1, \dots, \boldsymbol{y}_N) \right)^2 \\
&= \frac{1}{\boldsymbol{\alpha}!} \left(\mathop{\mathbb{E}}_{\boldsymbol{x} \sim \mathcal{X}_n, \boldsymbol{y} \sim \mathbb{Q}_n} \left(1 + \beta \|\boldsymbol{x}\|^2\right)^{-N/2} \partial_{\boldsymbol{y}}^{\boldsymbol{\alpha}} \prod_{i=1}^N \exp\left(\frac{1}{2} \frac{\beta}{1 + \beta \|\boldsymbol{x}\|^2} \langle \boldsymbol{x}, \boldsymbol{y}_i \rangle^2 \right) \right)^2 \\
&= \frac{1}{\prod_{i=1}^N \boldsymbol{\alpha}_i!} \left(\mathop{\mathbb{E}}_{\boldsymbol{x} \sim \mathcal{X}_n} \left(1 + \beta \|\boldsymbol{x}\|^2\right)^{-N/2} \prod_{i=1}^N \mathop{\mathbb{E}}_{\boldsymbol{y} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_n)} \left[\partial_{\boldsymbol{y}}^{\boldsymbol{\alpha}_i} \exp\left(\frac{1}{2} \frac{\beta}{1 + \beta \|\boldsymbol{x}\|^2} \langle \boldsymbol{x}, \boldsymbol{y} \rangle^2 \right) \right] \right)^2
\end{aligned} \tag{28}$$

where the $\boldsymbol{\alpha}_i \in \mathbb{N}^n$ are the components of $\boldsymbol{\alpha}$ corresponding to $\boldsymbol{y}_i$, for each $i \in [N]$. When $\beta = 0$, our result follows from the above, giving $\langle L_{n,\gamma,\beta,\mathcal{X}}, \widehat{H}_{\boldsymbol{\alpha}} \rangle^2 = \delta_{0,|\boldsymbol{\alpha}|}$. (Indeed, in this case the null and planted models are identical, so $L_{n,\gamma,\beta,\mathcal{X}} = 1$ is a constant, which is compatible with the above.) Let us suppose $\beta \neq 0$ below.

In this case, using Proposition A.7, we have

$$\begin{aligned}
&\partial_{\boldsymbol{y}}^{\boldsymbol{\alpha}_i} \exp\left(\frac{1}{2} \frac{\beta}{1 + \beta \|\boldsymbol{x}\|^2} \langle \boldsymbol{x}, \boldsymbol{y} \rangle^2 \right) \\
&\quad = \left(\frac{1 + \beta \|\boldsymbol{x}\|^2}{\beta} \right)^{-|\boldsymbol{\alpha}_i|} \boldsymbol{x}^{\boldsymbol{\alpha}_i} h_{|\boldsymbol{\alpha}_i|}\left(\langle \boldsymbol{x}, \boldsymbol{y} \rangle; -\frac{1 + \beta \|\boldsymbol{x}\|^2}{\beta} \right) \exp\left(\frac{1}{2} \frac{\beta}{1 + \beta \|\boldsymbol{x}\|^2} \langle \boldsymbol{x}, \boldsymbol{y} \rangle^2 \right).
\end{aligned} \tag{29}$$

(Note that the sign of the spike, or equivalently the sign of β, is the opposite of the sign of the variance of the Hermite polynomials that appear; thus, it is the negatively spiked case that corresponds to the more natural positive variance Hermite polynomials.) Since when $\boldsymbol{y} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_n)$ then $\langle \boldsymbol{x}, \boldsymbol{y} \rangle \sim \mathcal{N}(0, \|\boldsymbol{x}\|^2)$, we find

$$\begin{aligned}
\langle L_{n,\gamma,\beta,\mathcal{X}}, \widehat{H}_{\boldsymbol{\alpha}} \rangle^2 = &\frac{\beta^{2|\boldsymbol{\alpha}|}}{\prod_{i=1}^N \boldsymbol{\alpha}_i!} \left(\mathop{\mathbb{E}}_{\boldsymbol{x} \sim \mathcal{X}_n} \frac{\boldsymbol{x}^{\sum_{i=1}^N \boldsymbol{\alpha}_i}}{(1 + \beta \|\boldsymbol{x}\|^2)^{|\boldsymbol{\alpha}| + N/2}} \right. \\
&\left. \prod_{i=1}^N \mathop{\mathbb{E}}_{g \sim \mathcal{N}(\boldsymbol{0}, \|\boldsymbol{x}\|^2)} h_{|\boldsymbol{\alpha}_i|}\left(g; -\frac{1 + \beta \|\boldsymbol{x}\|^2}{\beta} \right) \exp\left(\frac{1}{2} \frac{\beta}{1 + \beta \|\boldsymbol{x}\|^2} g^2 \right) \right)^2.
\end{aligned} \tag{30}$$

We next focus on the innermost expectation. We may rewrite:

$$\begin{aligned}
&\mathop{\mathbb{E}}_{g\sim\mathcal{N}(\mathbf{0},\|\boldsymbol{x}\|^2)} h_{|\boldsymbol{\alpha}_i|}\left(g; -\frac{1+\beta\|\boldsymbol{x}\|^2}{\beta}\right)\exp\left(\frac{1}{2}\frac{\beta}{1+\beta\|\boldsymbol{x}\|^2}g^2\right)\\
&= \frac{1}{\sqrt{2\pi\|\boldsymbol{x}\|^2}}\int_{-\infty}^{\infty} h_{|\boldsymbol{\alpha}_i|}\left(g; -\frac{1+\beta\|\boldsymbol{x}\|^2}{\beta}\right)\exp\left(-\frac{1}{2}\left(\frac{1}{\|\boldsymbol{x}\|^2}-\frac{\beta}{1+\beta\|\boldsymbol{x}\|^2}\right)g^2\right)dg\\
&= \frac{(1+\beta\|\boldsymbol{x}\|^2)^{1/2}}{\sqrt{2\pi\|\boldsymbol{x}\|^2(1+\beta\|\boldsymbol{x}\|^2)}}\int_{-\infty}^{\infty} h_{|\boldsymbol{\alpha}_i|}\left(g; -\frac{1+\beta\|\boldsymbol{x}\|^2}{\beta}\right)\exp\left(-\frac{1}{2\|\boldsymbol{x}\|^2(1+\beta\|\boldsymbol{x}\|^2)}g^2\right)dg\\
&= (1+\beta\|\boldsymbol{x}\|^2)^{1/2}\mathop{\mathbb{E}}_{g\sim\mathcal{N}(0,\|\boldsymbol{x}\|^2(1+\beta\|\boldsymbol{x}\|^2))} h_{|\boldsymbol{\alpha}_i|}\left(g; -\frac{1+\beta\|\boldsymbol{x}\|^2}{\beta}\right). \qquad (31)
\end{aligned}$$

By Proposition A.8, this quantity will be zero unless $|\boldsymbol{\alpha}_i|$ is even, and thus $\langle L_{n,\gamma,\beta,\mathcal{X}}, \widehat{H}_{\boldsymbol{\alpha}}\rangle^2$ will be zero unless $|\boldsymbol{\alpha}_i|$ is even for all i. In this case, by Proposition A.8,

$$\mathop{\mathbb{E}}_{g\sim\mathcal{N}(\mathbf{0},\|\boldsymbol{x}\|^2)} h_{|\boldsymbol{\alpha}_i|}\left(g; -\frac{1+\beta\|\boldsymbol{x}\|^2}{\beta}\right)\exp\left(\frac{1}{2}\frac{\beta}{1+\beta\|\boldsymbol{x}\|^2}g^2\right) = (|\boldsymbol{\alpha}_i|-1)!!\frac{(1+\beta\|\boldsymbol{x}\|^2)^{|\boldsymbol{\alpha}_i|+1/2}}{\beta^{|\boldsymbol{\alpha}_i|/2}}. \qquad (32)$$

Substituting into (30), we find many cancellations after which we are left with

$$\langle L_{n,\gamma,\beta,\mathcal{X}}, \widehat{H}_{\boldsymbol{\alpha}}\rangle^2 = \frac{\prod_{i=1}^N (|\boldsymbol{\alpha}_i|-1)!!^2}{\prod_{i=1}^N \boldsymbol{\alpha}_i!}\beta^{|\boldsymbol{\alpha}|}\left(\mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathcal{X}_n} \boldsymbol{x}^{\sum_{i=1}^N \boldsymbol{\alpha}_i}\right)^2, \qquad (33)$$

the final result. ◀

A.3 Norm of the Low-Degree Likelihood Ratio

Proof of Lemma 5.9. Recall that

$$\|L^{\le D}_{n,\gamma,\beta,\mathcal{X}}\|^2_{L^2(\mathbb{Q}_n)} = \sum_{\substack{\boldsymbol{\alpha}\in(\mathbb{N}^n)^N\\ |\boldsymbol{\alpha}|\le D}} \langle L_{n,\gamma,\beta,\mathcal{X}}, \widehat{H}_{\boldsymbol{\alpha}}\rangle^2. \qquad (34)$$

We substitute in the result of Lemma 5.8, which, after introducing independent replicas $\boldsymbol{x}^1, \boldsymbol{x}^2 \sim \mathcal{X}_n$, may be rewritten as

$$\begin{aligned}
\|L^{\le D}_{n,\gamma,\beta,\mathcal{X}}\|^2_{L^2(\mathbb{Q}_n)} &= \mathop{\mathbb{E}}_{\boldsymbol{x}^1,\boldsymbol{x}^2\sim\mathcal{X}_n} \sum_{\substack{\boldsymbol{\alpha}_i\in\mathbb{N}^n, i\in[N]\\ |\boldsymbol{\alpha}_i| \text{even}\\ \sum_{i=1}^N|\boldsymbol{\alpha}_i|\le D}} \prod_{i=1}^N \frac{(|\boldsymbol{\alpha}_i|-1)!!^2}{\boldsymbol{\alpha}_i!}\beta^{|\boldsymbol{\alpha}_i|}(\boldsymbol{x}^1)^{\boldsymbol{\alpha}_i}(\boldsymbol{x}^2)^{\boldsymbol{\alpha}_i}\\
&= \mathop{\mathbb{E}}_{\boldsymbol{x}^1,\boldsymbol{x}^2\sim\mathcal{X}_n} \sum_{d=0}^D \beta^d \sum_{\substack{d_1,\dots,d_N \text{ even}\\ \sum d_i = d}} \left(\prod_{i=1}^N \frac{(d_i-1)!!^2}{d_i!}\right) \sum_{\substack{\boldsymbol{\alpha}_i\in\mathbb{N}^n, i\in[N]\\ |\boldsymbol{\alpha}_i|=d_i}} \prod_{i=1}^N \binom{d_i}{\boldsymbol{\alpha}_i} \prod_{j=1}^n (x_j^1 x_j^2)^{\boldsymbol{\alpha}_i(j)}.
\end{aligned}$$

By the multinomial theorem,

$$\langle \boldsymbol{x}^1, \boldsymbol{x}^2\rangle^{d_i} = \sum_{\substack{\boldsymbol{\alpha}\in\mathbb{N}^n\\ |\boldsymbol{\alpha}|=d_i}} \binom{d_i}{\boldsymbol{\alpha}} \prod_{j=1}^n (x_j^1 x_j^2)^{\boldsymbol{\alpha}(j)},$$

and therefore

$$\begin{aligned}\langle \boldsymbol{x}^1, \boldsymbol{x}^2\rangle^{\sum_{i=1}^N d_i} &= \prod_{i=1}^N \langle \boldsymbol{x}^1, \boldsymbol{x}^2\rangle^{d_i} \\ &= \prod_{i=1}^N \sum_{\substack{\boldsymbol{\alpha}\in\mathbb{N}^n \\ |\boldsymbol{\alpha}|=d_i}} \binom{d_i}{\boldsymbol{\alpha}} \prod_{j=1}^n (x_j^1 x_j^2)^{\boldsymbol{\alpha}(j)} \\ &= \sum_{\substack{\boldsymbol{\alpha}_i\in\mathbb{N}^n, i\in[N] \\ |\boldsymbol{\alpha}_i|=d_i}} \prod_{i=1}^N \binom{d_i}{\boldsymbol{\alpha}_i} \prod_{j=1}^n (x_j^1 x_j^2)^{\boldsymbol{\alpha}_i(j)}.\end{aligned}$$

In our case, this shows

$$\begin{aligned}\|L_{n,\gamma,\beta,\mathcal{X}}^{\leq D}\|_{L^2(\mathbb{Q}_n)}^2 &= \mathop{\mathbb{E}}_{\boldsymbol{x}^1,\boldsymbol{x}^2\sim\mathcal{X}_n} \sum_{\substack{0\leq d\leq D \\ d \text{ even}}} \beta^d \sum_{\substack{d_1,\dots,d_N \text{ even} \\ \sum d_i = d}} \left(\prod_{i=1}^N \frac{(d_i-1)!!^2}{d_i!}\right)\left(\prod_{i=1}^N \langle \boldsymbol{x}^1, \boldsymbol{x}^2\rangle^{d_i}\right) \\ &= \mathop{\mathbb{E}}_{\boldsymbol{x}^1,\boldsymbol{x}^2\sim\mathcal{X}_n} \sum_{\substack{0\leq d\leq D \\ d \text{ even}}} \beta^d \langle \boldsymbol{x}^1, \boldsymbol{x}^2\rangle^d \sum_{\substack{d_1,\dots,d_N \text{ even} \\ \sum d_i = d}} \prod_{i=1}^N \frac{d_i!}{d_i!!^2} \\ &= \mathop{\mathbb{E}}_{\boldsymbol{x}^1,\boldsymbol{x}^2\sim\mathcal{X}_n} \sum_{\substack{0\leq d\leq D \\ d \text{ even}}} 2^{-d}\beta^d \langle \boldsymbol{x}^1, \boldsymbol{x}^2\rangle^d \sum_{\substack{d_1,\dots,d_N \text{ even} \\ \sum d_i = d}} \prod_{i=1}^N \binom{d_i}{d_i/2}\end{aligned}$$

where we have used the identities $n! = n!! \cdot (n-1)!!$ and $(2n)!! = 2^n \cdot n!$. We now pass to a notation making the restriction to even degrees clearer:

$$\|L_{n,\gamma,\beta,\mathcal{X}}^{\leq D}\|_{L^2(\mathbb{Q}_n)}^2 = \mathop{\mathbb{E}}_{\boldsymbol{x}^1,\boldsymbol{x}^2\sim\mathcal{X}_n} \sum_{0\leq d\leq \lfloor D/2\rfloor} \left(\frac{\beta^2\langle \boldsymbol{x}^1, \boldsymbol{x}^2\rangle^2}{4}\right)^d \sum_{\substack{d_1,\dots,d_N \\ \sum d_i = d}} \prod_{i=1}^N \binom{2d_i}{d_i}.$$

The remaining function may be understood in terms of the generating function of the central binomial coefficients: using Proposition 5.7, we have that for any $x \in (-\frac{1}{4}, \frac{1}{4})$,

$$\varphi_N(x) := (1-4x)^{-N/2} = \left(\sum_{d\geq 0} \binom{2d}{d} x^d\right)^N = \sum_{d\geq 0} x^d \sum_{\substack{d_1,\dots,d_N \\ \sum d_i = d}} \prod_{i=1}^N \binom{2d_i}{d_i}.$$

Writing $\varphi_{N,k}(x)$ for the truncation of this Taylor series to degree k, we see that

$$\|L_{n,\gamma,\beta,\mathcal{X}}^{\leq D}\|_{L^2(\mathbb{Q}_n)}^2 = \mathop{\mathbb{E}}_{\boldsymbol{x}^1,\boldsymbol{x}^2\sim\mathcal{X}_n} \left[\varphi_{N,\lfloor D/2\rfloor}\left(\frac{\beta^2\langle \boldsymbol{x}^1, \boldsymbol{x}^2\rangle^2}{4}\right)\right],$$

the final result. ◀

B Proofs for Bounding the Low-Degree Likelihood Ratio

B.1 Local Chernoff Bound

Proof of Proposition 5.12. It is sufficient to show that $\mathsf{iid}(\pi/\sqrt{n})$ admits a local Chernoff bound. Since π is subgaussian, $\pi^2 - \mathbb{E}[\pi^2]$ is subexponential (see, e.g., [56]), i.e., the moment-generating function $M(t) = \mathbb{E}[\exp(t(\pi^2 - \mathbb{E}[\pi^2]))]$ satisfies $M(t) \leq \exp(\frac{t^2}{2s^2})$ for all $|t| \leq s$ for

a suitable choice of a constant $s > 0$. In particular, $\mathbb{E}[\exp(t\pi^2)] < \infty$ for all $|t| \le s$ for this choice of $s > 0$.

Let $\Pi = \pi\pi'$, the product of two independent copies of π. Let σ^2 be the variance proxy of π (see Definition 2.1). The moment-generating function of Π is

$$M(t) = \mathbb{E}[\exp(t\Pi)] = \mathbb{E}_\pi \mathbb{E}_{\pi'}[\exp(t\pi\pi')] \le \mathbb{E}_\pi \left[\exp\left(\sigma^2 t^2 \pi^2/2\right)\right] < \infty$$

provided $\frac{1}{2}\sigma^2 t^2 < s$, i.e., $|t| < \sqrt{2s/\sigma^2}$. Thus $M(t)$ exists in an open interval containing $t = 0$, which implies $M'(0) = \mathbb{E}[\Pi] = 0$ and $M''(0) = \mathbb{E}[\Pi^2] = 1$ (this is the defining property of the moment-generating function: its derivatives at $t = 0$ are the moments of Π).

Let $\eta > 0$ and $f(t) = \exp\left(\frac{t^2}{2(1-\eta)}\right)$. Since $M(0) = 1, M'(0) = 0, M''(0) = 1$ and $f(0) = 1, f'(0) = 0, f''(0) = \frac{1}{1-\eta} > 1$, there exists $\delta > 0$ such that for all $t \in [-\delta, \delta]$, $M(t)$ exists and $M(t) \le f(t)$.

We now apply the standard Chernoff bound argument to $\langle \boldsymbol{x}^1, \boldsymbol{x}^2 \rangle = \frac{1}{n}\sum_{i=1}^n \Pi_i$, where $\Pi_1, \ldots, \Pi_n$ are i.i.d. copies of Π. For any $\lambda > 0$,

$$\begin{aligned}
\Pr\left\{\langle \boldsymbol{x}^1, \boldsymbol{x}^2 \rangle \ge t\right\} &= \Pr\left\{\exp(\lambda\langle \boldsymbol{x}^1, \boldsymbol{x}^2 \rangle) \ge \exp(\lambda t)\right\} \\
&\le \exp(-\lambda t)\mathbb{E}[\exp(\lambda\langle \boldsymbol{x}^1, \boldsymbol{x}^2 \rangle)] && \text{(by Markov's inequality)} \\
&= \exp(-\lambda t)\mathbb{E}[\exp(\lambda n^{-1}\sum_{i=1}^n \Pi_i)] \\
&= \exp(-\lambda t)[M(\lambda/n)]^n \\
&\le \exp(-\lambda t)[f(\lambda/n)]^n && \text{(provided } \lambda/n \le \delta\text{)} \\
&\le \exp(-\lambda t)\exp\left(\frac{\lambda^2}{2(1-\eta)n}\right).
\end{aligned}$$

Taking $\lambda = (1-\eta)nt$,

$$\Pr\left\{\langle \boldsymbol{x}^1, \boldsymbol{x}^2 \rangle \ge t\right\} \le \exp\left(-(1-\eta)nt^2 + \frac{1}{2}(1-\eta)nt^2\right) = \exp\left(-\frac{1}{2}(1-\eta)nt^2\right)$$

as desired. This holds provided $\lambda/n \le \delta$, i.e., $t \le \delta/(1-\eta)$. The same argument (with $-\Pi$ instead of Π) holds for the other tail bound $\Pr\left\{\langle \boldsymbol{x}^1, \boldsymbol{x}^2 \rangle \le -t\right\}$. ◀

B.2 Bounding the Large Deviations

Proof of Lemma 5.13. Recall that

$$\varphi_{N,\lfloor D/2 \rfloor}(x) = \sum_{d=0}^{\lfloor D/2 \rfloor} x^d \sum_{\substack{d_1,\ldots,d_N \\ \sum d_i = d}} \prod_{i=1}^N \binom{2d_i}{d_i}.$$

Note that the first sum above has $\lfloor D/2 \rfloor + 1$ terms and the second sum has at most $N^d \le N^{\lfloor D/2 \rfloor} \le N^{D/2}$ terms. It is combinatorially clear that $\binom{2d_i}{d_i}\binom{2d_j}{d_j} \le \binom{2(d_i+d_j)}{d_i+d_j}$, and therefore

$$\prod_{i=1}^N \binom{2d_i}{d_i} \le \binom{2\sum_{i=1}^N d_i}{\sum_{i=1}^N d_i} = \binom{2d}{d} \le (2d)^d \le D^{D/2}.$$

Since $\|\boldsymbol{x}^1\|^2 \le 2$ and $\|\boldsymbol{x}^2\|^2 \le 2$ we have $\frac{1}{4}\beta^2\langle \boldsymbol{x}^1, \boldsymbol{x}^2 \rangle^2 \le \beta^2$. Since $d \le D/2$ we have $(\beta^2)^d \le (1+\beta^2)^{D/2}$, and therefore

$$\varphi_{N,\lfloor D/2 \rfloor}\left(\frac{\beta^2}{4}\langle \boldsymbol{x}^1, \boldsymbol{x}^2 \rangle^2\right) \le (D/2+1)(1+\beta^2)^{D/2} N^{D/2} D^{D/2} \le (1+\beta^2)^{D/2} D N^{D/2} D^{D/2}.$$

Combining these bounds,

$$\begin{aligned} R_2 &= \mathop{\mathbb{E}}_{\boldsymbol{x}^1,\boldsymbol{x}^2\sim\mathcal{X}_n}\left[\mathbb{1}_{|\langle \boldsymbol{x}^1,\boldsymbol{x}^2\rangle|>\varepsilon}\,\varphi_{N,\lfloor D/2\rfloor}\left(\frac{\beta^2}{4}\langle \boldsymbol{x}^1,\boldsymbol{x}^2\rangle^2\right)\right] \\ &\leq \Pr\left\{|\langle \boldsymbol{x}^1,\boldsymbol{x}^2\rangle|>\varepsilon\right\}(1+\beta^2)^{D/2}DN^{D/2}D^{D/2}. \end{aligned}$$

Since R_2 increases as ε decreases, we can assume without loss of generality that ε is small enough that we may apply the local Chernoff bound (18):

$$\begin{aligned} &\leq \exp\left(-\frac{1}{3}n\varepsilon^2\right)(1+\beta^2)^{D/2}DN^{D/2}D^{D/2} \\ &= \exp\left(-\frac{1}{3}n\varepsilon^2+\frac{D}{2}\log(1+\beta^2)+\log D+\frac{D}{2}\log N+\frac{D}{2}\log D\right) \\ &= o(1) \end{aligned}$$

provided $D=o(n/\log n)$, completing the proof. ◀

B.3 Bounding the Small Deviations

Proof of Lemma 5.14. We use the argument from Appendix K of [54]. Since the Taylor series for $\varphi_N(x)$ has nonnegative coefficients, we have $\varphi_{N,\lfloor D/2\rfloor}(x)\leq\varphi_N(x)$ for all $x\in[0,1/4)$. Taking $\varepsilon<1/|\beta|$, we have

$$\begin{aligned} R_1 &= \mathop{\mathbb{E}}_{\boldsymbol{x}^1,\boldsymbol{x}^2\sim\mathcal{X}_n}\left[\mathbb{1}_{|\langle \boldsymbol{x}^1,\boldsymbol{x}^2\rangle|\leq\varepsilon}\,\varphi_{N,\lfloor D/2\rfloor}\left(\frac{\beta^2}{4}\langle \boldsymbol{x}^1,\boldsymbol{x}^2\rangle^2\right)\right] \\ &\leq \mathop{\mathbb{E}}_{\boldsymbol{x}^1,\boldsymbol{x}^2\sim\mathcal{X}_n}\left[\mathbb{1}_{|\langle \boldsymbol{x}^1,\boldsymbol{x}^2\rangle|\leq\varepsilon}\,\varphi_{N}\left(\frac{\beta^2}{4}\langle \boldsymbol{x}^1,\boldsymbol{x}^2\rangle^2\right)\right] \\ &= \mathop{\mathbb{E}}_{\boldsymbol{x}^1,\boldsymbol{x}^2\sim\mathcal{X}_n}\left[\mathbb{1}_{|\langle \boldsymbol{x}^1,\boldsymbol{x}^2\rangle|\leq\varepsilon}\,\left(1-\beta^2\langle \boldsymbol{x}^1,\boldsymbol{x}^2\rangle^2\right)^{-N/2}\right] \\ &= \mathop{\mathbb{E}}_{\boldsymbol{x}^1,\boldsymbol{x}^2\sim\mathcal{X}_n}\left[\mathbb{1}_{|\langle \boldsymbol{x}^1,\boldsymbol{x}^2\rangle|\leq\varepsilon}\,\exp\left(-\frac{N}{2}\log\left(1-\beta^2\langle \boldsymbol{x}^1,\boldsymbol{x}^2\rangle^2\right)\right)\right]. \end{aligned}$$

By the convexity of $t\mapsto-\log(1-\beta^2 t)$, we have $-\log(1-\beta^2 t)\leq-\frac{t}{\varepsilon^2}(1-\beta^2\varepsilon^2)$ for all $t\in[0,\varepsilon^2]$. Letting $c:=-\frac{N}{2\varepsilon^2}\log(1-\beta^2\varepsilon^2)>0$, we proceed bounding

$$\begin{aligned} &\leq \mathop{\mathbb{E}}_{\boldsymbol{x}^1,\boldsymbol{x}^2\sim\mathcal{X}_n}\left[\mathbb{1}_{|\langle \boldsymbol{x}^1,\boldsymbol{x}^2\rangle|\leq\varepsilon}\,\exp\left(c\langle \boldsymbol{x}^1,\boldsymbol{x}^2\rangle^2\right)\right] \\ &= \int_0^\infty \Pr\left\{\mathbb{1}_{|\langle \boldsymbol{x}^1,\boldsymbol{x}^2\rangle|\leq\varepsilon}\,\exp\left(c\langle \boldsymbol{x}^1,\boldsymbol{x}^2\rangle^2\right)>u\right\}du \\ &= \int_0^\infty \Pr\left\{|\langle \boldsymbol{x}^1,\boldsymbol{x}^2\rangle|\leq\varepsilon \quad\text{and}\quad \exp\left(c\langle \boldsymbol{x}^1,\boldsymbol{x}^2\rangle^2\right)>u\right\}du \\ &\leq 1+\int_1^\infty \Pr\left\{|\langle \boldsymbol{x}^1,\boldsymbol{x}^2\rangle|\leq\varepsilon \quad\text{and}\quad \exp\left(c\langle \boldsymbol{x}^1,\boldsymbol{x}^2\rangle^2\right)>u\right\}du. \end{aligned}$$

Applying the change of variables $u=\exp(ct)$,

$$\begin{aligned} &= 1+\int_0^\infty \Pr\left\{|\langle \boldsymbol{x}^1,\boldsymbol{x}^2\rangle|\leq\varepsilon \quad\text{and}\quad \langle \boldsymbol{x}^1,\boldsymbol{x}^2\rangle^2>t\right\}c\exp(ct)\,dt \\ &\leq 1+\int_0^{\varepsilon^2} \Pr\left\{\langle \boldsymbol{x}^1,\boldsymbol{x}^2\rangle^2>t\right\}c\exp(ct)\,dt. \end{aligned}$$

Provided ε is sufficiently small, we can apply the local Chernoff bound (18):

$$\leq 1 + Cc \int_0^{\varepsilon^2} \exp\left(-\frac{1}{2}(1-\eta)nt + ct\right) dt.$$

Let $\widehat{\gamma} := n/N$, so that $\widehat{\gamma} \to \gamma$ as $n \to \infty$. Letting $c =: \widehat{c}n$ where $\widehat{c} = -\log(1-\beta^2\varepsilon^2)/(2\varepsilon^2\widehat{\gamma})$,

$$\leq 1 + C \cdot \widehat{c}n \int_0^{\varepsilon^2} \exp\left[\left(-\frac{1}{2}(1-\eta) + \widehat{c}\right) nt\right] dt.$$

We have $\lim_{\varepsilon \to 0^+} \widehat{c} = \frac{\beta^2}{2\widehat{\gamma}}$. Since $\beta^2 < \gamma$, we have that for sufficiently large n, $\frac{\beta^2}{2\widehat{\gamma}} < \frac{1}{2}$. Thus we can choose ε and η small enough so that for sufficiently large n, $-\frac{1}{2}(1-\eta) + \widehat{c} \leq -\alpha$ for some $\alpha > 0$. Now

$$\begin{aligned}
&\leq 1 + C \cdot \widehat{c}n \int_0^{\infty} \exp\left(-\alpha nt\right) dt \\
&= 1 + \frac{C \cdot \widehat{c}}{\alpha} \\
&= O(1),
\end{aligned}$$

completing the proof. ◀

B.4 Above the BBP Threshold

Proof of Theorem 3.4 (Part 2). Let $\beta^2 > \gamma$. Recall

$$\begin{aligned}
\|L^{\leq D}_{n,\gamma,\beta,\mathcal{X}}\|^2_{L^2(\mathbb{Q}_n)} &= \mathop{\mathbb{E}}_{\boldsymbol{x}^1,\boldsymbol{x}^2 \sim \mathcal{X}_n} \left[\varphi_{N,\lfloor D/2 \rfloor}\left(\frac{\beta^2}{4}\langle \boldsymbol{x}^1, \boldsymbol{x}^2\rangle^2\right)\right] \\
&= \sum_{d=0}^{\lfloor D/2 \rfloor} \mathop{\mathbb{E}}_{\boldsymbol{x}^1,\boldsymbol{x}^2 \sim \mathcal{X}_n} \left(\frac{\beta^2}{4}\langle \boldsymbol{x}^1, \boldsymbol{x}^2\rangle^2\right)^d \sum_{\substack{d_1,\ldots,d_N \\ \sum d_i = d}} \prod_{i=1}^{N} \binom{2d_i}{d_i}. \qquad (35)
\end{aligned}$$

Since each term in the outer summation of (35) is nonnegative, it sufficies to fix a single $d \leq D/2$ and show that the corresponding term is $\omega(1)$. We can write $\langle \boldsymbol{x}^1, \boldsymbol{x}^2\rangle = \frac{1}{n}\sum_{i=1}^n \Pi_i$ where $\Pi_1, \ldots, \Pi_n$ are i.i.d. with distribution of the product $\Pi = \pi\pi'$ of two independent copies of π. This means

$$\mathop{\mathbb{E}}_{\boldsymbol{x}^1,\boldsymbol{x}^2 \sim \mathcal{X}_n} \langle \boldsymbol{x}^1, \boldsymbol{x}^2\rangle^{2d} = \mathbb{E}\left(\frac{1}{n}\sum_{i=1}^n \Pi_i\right)^{2d} = n^{-2d} \sum_{i_1,\ldots,i_{2d} \in [n]} \mathbb{E}[\Pi_{i_1}\Pi_{i_2}\cdots\Pi_{i_{d2}}].$$

Since π is symmetric about zero, Π is also symmetric about zero, so all moments $\mathbb{E}[\Pi^k]$ are nonnegative. This means each term in the remaining sum is nonnegative, so we can obtain a lower bound by only considering terms where each index occurring among the $i_1, \ldots, i_{2d}$ occurs exactly twice:

$$\mathop{\mathbb{E}}_{\boldsymbol{x}^1,\boldsymbol{x}^2 \sim \mathcal{X}_n} \langle \boldsymbol{x}^1, \boldsymbol{x}^2\rangle^{2d} \geq n^{-2d}\binom{n}{d}\frac{(2d)!}{2^d}\left(\mathbb{E}[\Pi^2]\right)^d = n^{-2d}\binom{n}{d}\frac{(2d)!}{2^d}.$$

Next, we bound the inner summation of (35) below by taking only the terms with $d_i \in \{0, 1\}$ for all $i \in [N]$:

$$\sum_{\substack{d_1,\ldots,d_N \\ \sum d_i = d}} \prod_{i=1}^{N} \binom{2d_i}{d_i} \geq \binom{N}{d} 2^d.$$

Combining these bounds, we find that for any fixed $0 \leq d \leq \lfloor D/2 \rfloor$,

$$\begin{aligned} \|L_{n,\gamma,\beta,\mathcal{X}}^{\leq D}\|_{L^2(\mathbb{Q}_n)}^2 &\geq \frac{\beta^{2d}}{4^d} n^{-2d} \binom{n}{d} \frac{(2d)!}{2^d} \binom{N}{d} 2^d \\ &= \left(\frac{\beta^2}{4n^2}\right)^d \frac{(2d)!\, n!\, N!}{(d!)^2\, (n-d)!\, (N-d)!} \\ &\geq \left(\frac{\beta^2 (n-d)(N-d)}{4n^2}\right)^d \binom{2d}{d}. \end{aligned}$$

Using the standard bound $\binom{2d}{d} \geq 4^d/(2\sqrt{d})$,

$$\|L_{n,\gamma,\beta,\mathcal{X}}^{\leq D}\|_{L^2(\mathbb{Q}_n)}^2 \geq \frac{1}{2\sqrt{d}} \left(\frac{\beta^2 (n-d)(N-d)}{n^2}\right)^d.$$

This final expression will be $\omega(1)$ provided that $1 \ll d \ll n$, since $n/N \to \gamma$ and $\beta^2 > \gamma$. ◀

Pseudo-Deterministic Streaming

Shafi Goldwasser
Simons Institute for the Theory of Computing, Berkeley, CA, USA
MIT, Cambridge, MA, USA
http://people.csail.mit.edu/shafi/
shafi.goldwasser@gmail.com

Ofer Grossman
MIT, Cambridge, MA, USA
ofer.grossman@gmail.com

Sidhanth Mohanty
University of California Berkeley, CA, USA
http://sidhanthm.com/
sidhanthm@cs.berkeley.edu

David P. Woodruff
Carnegie-Mellon University, Pittsburgh, PA, USA
http://www.cs.cmu.edu/~dwoodruf/
dwoodruf@cs.cmu.edu

Abstract

A pseudo-deterministic algorithm is a (randomized) algorithm which, when run multiple times on the same input, with high probability outputs the same result on all executions. Classic streaming algorithms, such as those for finding heavy hitters, approximate counting, ℓ_2 approximation, finding a nonzero entry in a vector (for turnstile algorithms) are not pseudo-deterministic. For example, in the instance of finding a nonzero entry in a vector, for any known low-space algorithm A, there exists a stream x so that running A twice on x (using different randomness) would with high probability result in two different entries as the output.

In this work, we study whether it is inherent that these algorithms output different values on different executions. That is, we ask whether these problems have low-memory pseudo-deterministic algorithms. For instance, we show that there is no low-memory pseudo-deterministic algorithm for finding a nonzero entry in a vector (given in a turnstile fashion), and also that there is no low-dimensional pseudo-deterministic sketching algorithm for ℓ_2 norm estimation. We also exhibit problems which do have low memory pseudo-deterministic algorithms but no low memory deterministic algorithm, such as outputting a nonzero row of a matrix, or outputting a basis for the row-span of a matrix.

We also investigate multi-pseudo-deterministic algorithms: algorithms which with high probability output one of a few options. We show the first lower bounds for such algorithms. This implies that there are streaming problems such that every low space algorithm for the problem must have inputs where there are many valid outputs, all with a significant probability of being outputted.

2012 ACM Subject Classification Theory of computation → Streaming, sublinear and near linear time algorithms

Keywords and phrases streaming, pseudo-deterministic

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.79

Funding *Shafi Goldwasser*: Supported by NSF CNS-1413920, DARPA/NJIT 491512803, Sloan Foundation 996698, and MIT/IBM W1771646. This work was done at the Simons Institute for the Theory of Computing.
Ofer Grossman: Supported by the Fannie and John Hertz Foundation fellowship, an NSF GRFP award, NSF CNS-1413920, DARPA/NJIT 491512803, Sloan Foundation 996698, and MIT/IBM W1771646. This work was done in part at the Simons Institute for the Theory of Computing.

Sidhanth Mohanty: Supported by NSF grant CCF-1718695. This work was done in part at the Simons Institute for the Theory of Computing.
David P. Woodruff: Supported by the National Science Foundation under Grant No. CCF-1815840. This work was done in part at the Simons Institute for the Theory of Computing.

1 Introduction

Consider some classic streaming problems: heavy hitters, approximate counting, ℓ_p approximation, finding a nonzero entry in a vector (for turnstile algorithms), counting the number of distinct elements in a stream. These problems were shown to have low-space randomized algorithms in [4, 27, 7, 2, 20, 26], respectively. All of these algorithms exhibit the property that when running the algorithm multiple times on the same stream, different outputs may result on the different executions.

For the sake of concreteness, let's consider the problem of ℓ_2 approximation: given a stream of poly(n) updates to a vector (the vector begins as the zero vector, and updates are of the form "increase the i^{th} entry by 1" or "decrease the j^{th} entry by 1"), output an approximation of the ℓ_2 norm of the vector. There exists a celebrated randomized algorithm for this problem [2]. This algorithm has the curious property that running the same algorithm multiple times on the same stream may result in different approximations. That is, if Alice runs the algorithm on the same stream as Bob (but using different randomness), Alice may get some approximation of the ℓ_2 norm (such as 27839.8), and Bob (running the same algorithm, but with your own randomness) may get a different approximation (such as 27840.2). The randomized algorithm has the guarantee that both of the approximations will be close to the true value. However, interestingly, Alice and Bob end up with slightly different approximations. Is this behavior inherent? That is, could there exist an algorithm which, while being randomized, for all streams with high probability both Alice and Bob will end up with the *same* approximation for the ℓ_2 norm?

Such an algorithm, which when run on the same stream multiple times outputs the same output with high probability is called *pseudo-deterministic*. The main question we tackle in this paper is:

What streaming problems have low-memory pseudo-deterministic algorithms?

1.1 Our Contributions

This paper is the first to investigate pseudo-determinism in the context of streaming algorithms. We show certain problems have pseudo-deterministic algorithms substantially faster than the optimal deterministic algorithm, while other problems do not.

1.1.1 Lower Bounds

1.1.1.1 Find-Support-Elem

We show pseudo-deterministic lower bounds for finding a nonzero entry in a vector in the turnstile model. Specifically, consider the problem Find-Support-Elem of finding a nonzero entry in a vector for a turnstile algorithm (the input is a stream of updates of the form "increase entry i by 1" or "decrease entry j by 1", and we wish to find a nonzero entry in the final vector). We show this problem does not have a low-memory pseudo-deterministic algorithm:

▶ **Theorem 1.** *There is no pseudo-deterministic algorithm for* Find-Support-Elem *which uses* $\tilde{o}(n)$ *memory.*

This is in contrast with the work of [26], which shows a randomized algorithm for the problem using polylogarithmic space.

Theorem 1 can be viewed as showing that any low-memory algorithm A for Find-Support-Elem must have an input x where the output $A(x)$ (viewed as a random variable depending on the randomness used by A) must have at least a little bit of entropy. The algorithms we know for Find-Support-Elem have a very high amount of entropy in their outputs (the standard algorithms, for an input which is the all 1s vector, will find a uniformly random entry). Is this inherent, or can the entropy of the output be reduced? We show that this is inherent: for every low memory algorithm there is an input x such that $A(x)$ has high entropy.

▶ **Theorem 2.** *Every randomized algorithm for* Find-Support-Elem *using* $o(s)$ *space must have an input* x *such that* $A(x)$ *has entropy at least* $\log\left(\frac{n}{s\log n}\right)$.

So, in particular, an algorithm using $n^{1-\varepsilon}$ space must have outputs with entropy $\Omega(\log n)$, which is maximal up to constant factors.

We also show analogous lower bounds for the problem Find-Duplicatein which the input is a stream of $3n/2$ integers between 1 and n, and the goal is to output a number k which appears at least twice in the stream:

▶ **Theorem 3.** *Every randomized algorithm for* Find-Duplicate *using* $o(s)$ *space must have an input* x *such that* $A(x)$ *has entropy at least* $\log\left(\frac{n}{s\log n}\right)$.

1.1.1.2 Techniques

To prove a pseudo-deterministic lower bound for Find-Support-Elem, the idea is to show that if a pseudo-deterministic algorithm existed for Find-Support-Elem, then there would also exist a pseudo-deterministic one-way communication protocol for the problem One-Way-Find-Duplicate, where Alice has a subset of $[n]$ of size $3n/4$, and so does Bob, and they wish to find an element which they share.

To prove a lower bound on the one-way communication problem One-Way-Find-Duplicate, we show that if such a pseudo-deterministic protocol existed, then Bob can use Alice's message to recover many $(n/10)$ elements of her input (which contains much more information than one short message). The idea is that using Alice's message, Bob can find an element they have in common. Then, he can remove the element he found that they have in common from his input, and repeat to find another element they have in common (using the original message Alice sent, so Alice does not have to send another message). After repeating $n/10$ times, he will have found many elements which Alice has.

It may not be immediately obvious where pseudo-determinism is being used in this proof. The idea is that because the algorithm is pseudo-deterministic, the element which Bob finds as the intersection with high probability does not depend on the randomness used by Alice. That is, let $b_1, b_2, \ldots$ be the sequence of elements which Bob finds. Because the algorithm is pseudo-deterministic, there exists a specific sequence $b_1, b_2, \ldots$ such that with high probability this will be the sequence of elements which Bob finds. Notice that a randomized (but not pseudo-deterministic) algorithm for One-Way-Find-Duplicatewould result in different sequences on different executions.

When the sequence $b_1, b_2, \ldots$ is determined in advance, we can use a union bound and argue that with high probability, one of Alice's messages will likely work on all of Bob's inputs. If $b_1, b_2, \ldots$ is not determined in advance, then it's not possible to use a union bound.

Proving a lower bound on the entropy of the output of an algorithm for Find-Support-Elem uses a similar idea, but is more technically involved. It is harder to ensure that Bob's later inputs will be able to succeed with Alice's original message. The idea, at a very high level, is to have Alice send many messages (but not too many), so that Bob's new inputs will not strongly depend on any part of Alice's randomness, and also to have Alice send additional messages to keep Bob from going down a path where Alice's messages will no longer work.

This lower bound technique may seem similar to the way one would show a deterministic lower bound. It's worth noting that for certain problems, deterministic lower bounds do not generalize to pseudo-deterministic lower bounds; see our results on pseudo-deterministic upper bounds for some examples and intuition for why certain problems remain hard in the pseudo-deterministic setting while others do not.

1.1.1.3 Sketching lower bounds for pseudo-deterministic ℓ_2 norm estimation

The known randomized algorithms (such as [2]) for approximating the ℓ_2 norm of a vector x in a stream rely on *sketching*, i.e., storing $\boldsymbol{S}x$ where $\boldsymbol{S}$ is a $d \times n$ random matrix where $d \ll n$ and outputting the ℓ_2 norm of $\boldsymbol{S}x$. More generally, an abstraction of this framework is the setting where one has a distribution over matrices $\mathcal{D}$ and a function f. One then stores a *sketch* of the input vector $\boldsymbol{S}x$ where $\boldsymbol{S} \sim \mathcal{D}$ and outputs $f(\boldsymbol{S}x)$. By far, most streaming algorithms fall into this framework and in fact some recent work [24, 1] proves under some caveats and assumptions that low-space turnstile streaming algorithms imply algorithms based on low-dimensional sketches. Since sketching-based streaming algorithms are provably optimal in many settings, it motivates studying whether there are low-dimensional sketches of x from which the ℓ_2 norm can be estimated pseudo-deterministically.

We prove a lower bound on the dimension of sketches from which the ℓ_2 norm can be estimated pseudo-deterministically:

▶ **Theorem 4.** *Suppose $\mathcal{D}$ is a distribution over $d \times n$ matrices and f is a function from $\mathbb{R}^d$ to $\mathbb{R}$ such that for all $x \in \mathbb{R}^n$, when $\boldsymbol{S} \sim \mathcal{D}$:*

- *$f(\boldsymbol{S}x)$ approximates the ℓ_2 norm of x to within a constant factor with high probability,*
- *$f(\boldsymbol{S}x)$ takes a unique value with high probability.*

Then d must be $\Omega(n)$.

As an extension, we also show that

▶ **Theorem 5.** *For every constant $\varepsilon, \delta > 0$, every randomized sketching algorithm A for ℓ_2 norm estimation using a $O(n^{1-\delta})$-dimensional sketch, there is a vector x such that the output entropy of $A(x)$ is at least $1 - \varepsilon$. Furthermore, there is a randomized algorithm using a $O(\operatorname{poly}\log n)$-dimensional sketch with output entropy at most $1 + \varepsilon$ on all input vectors.*

1.1.1.4 Techniques

The first insight in our lower bound is that if there is a pseudo-deterministic streaming algorithm A for ℓ_2 norm estimation in k space, then that means there is a fixed function g such that $g(x)$ approximates $\|x\|_2$ and A is a randomized algorithm to compute $g(x)$ with high probability. The next step uses a result in the work of [17] to illustrate a (randomized) sequence of vectors $\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(t)}$ only depending on g such that any linear sketching-based

algorithm that uses sublinear dimensional sketches outputs an incorrect approximation to the ℓ_2 norm of some vector in that sequence with constant probability, thereby implying a dimension lower bound.

1.1.2 Upper Bounds

On the one hand, all the problems considered so far were such that

1. There were "low-space" randomized algorithms.
2. The pseudo-deterministic and deterministic space complexity were "high" and equal up to logarithmic factors.

This raises the question if there are natural problems where pseudo-deterministic algorithms outperform deterministic algorithms (by more than logarithmic factors). We answer this question in the affirmative.

We illustrate several natural problems where the pseudo-deterministic space complexity is strictly smaller than the deterministic space complexity.

The first problem is that of finding a nonzero row in a matrix given as input in a turnstile stream. Our result for this problem has the bonus of giving a natural problem where the pseudo-deterministic streaming space complexity is strictly sandwiched between the deterministic and randomized streaming space complexity.

In the problem Find-Nonzero-Row, the input is an $n \times d$ matrix A streamed in the turnstile model, and the goal is to output an i such that the i^{th} row of the matrix A is nonzero.

▶ **Theorem 6.** *The randomized space complexity for* Find-Nonzero-Row *is* $\widetilde{\Theta}(1)$*, the pseudo-deterministic space complexity for* Find-Nonzero-Row *is* $\widetilde{\Theta}(n)$*, and the deterministic space complexity for* Find-Nonzero-Row *is* $\widetilde{\Theta}(nd)$*.*

The idea behind the proof of Theorem 6 is to sample a random vector x, and then deterministically find a nonzero entry of Ax. With high probability, if a row of A is nonzero, then the corresponding entry of Ax will be nonzero as well.

1.1.2.1 Discussion

Roughly speaking, in this problem there is a certain structure that allows us to use randomness to "hash" pieces of the input together, and then apply a deterministic algorithm on the hashed pieces. The other upper bounds we show for pseudo-deterministic algorithms also have a structure which allows us to hash, and then use a deterministic algorithm. It is interesting to ask if there are natural problems which have faster pseudo-deterministic algorithms than the best deterministic algorithms, but for which the pseudo-deterministic algorithms follow a different structure.

The next problems we show upper bounds for are estimating frequencies in a length-m stream of elements from a large universe $[n]$ up to error εm, and that of estimating the inner product of two vectors x and y in an insertion-only stream of length-m up to error $\varepsilon \cdot \|x\|_1 \cdot \|y\|_1$. We show a separation between the deterministic and (weak) pseudo-deterministic space complexity in the regime where $m \ll n$.

▶ **Theorem 7.** *There is a pseudo-deterministic algorithm for point query estimation and inner product estimation that uses* $O\left(\frac{\log m}{\varepsilon} + \log n\right)$ *bits of space. On the other hand, any deterministic algorithm needs* $\Omega\left(\frac{\log n}{\varepsilon}\right)$ *bits of space.*

1.2 Related work

Pseudo-deterministic algorithms were introduced by Gat and Goldwasser [9]. Such algorithms have since been studied in the context of standard (sequential algorithms) [15, 30], average case algorithms [18], parallel algorithms [12], decision tree algorithms [11, 10], interactive proofs [13], learning algorithms [31], approximation algorithms [31, 6], and low space algorithms [16]. In this work, we initiate the study of pseudo-determinism in the context of streaming algorithms (and in the context of one-way communication complexity).

The problem of finding duplicates in a stream of integers between 1 and n was first considered by [14], where an $O(\log^3 n)$ bits of space algorithm is given, later improved by [22] to $O(\log^2 n)$ bits. We show that in contrast to these low space randomized algorithms, a pseudo-deterministic algorithm needs significantly more space in the regime where the length of the stream is, say, $3n/2$. [23] shows optimal lower bounds for randomized algorithms solving the problem.

The method of ℓ_p-sampling to sample an index of a turnstile vector with probability proportional to its ℓ_p mass, whose study was initiated in [26], is one way of outputting an element from the support of a turnstile stream. A line of work [8, 26, 22, 3], ultimately leading to an optimal algorithm in [21] and tight lower bounds in [23], characterizes the space complexity of randomized algorithms to output an element from the support of a turnstile vector as $\Theta(\operatorname{poly}\log n)$, in contrast with the space lower bounds we show for algorithms constrained to a low entropy output.

1.3 Open Problems

Morris Counters

In [27], Morris showed that one can approximate (up to a multiplicative error) the number of elements in a stream with up to n elements using $O(\log\log n)$ bits of space. Does there exist an $O(\log\log n)$ bits of space pseudo-deterministic algorithm for the problem?

ℓ_2-norm estimation

In this work, we show that there are no low-dimensional pseudo-deterministic sketching algorithms for estimating the ℓ_2-norm of a vector. However, we do not show a turnstile streaming lower bound for pseudo-deterministic algorithms, which motivates the following question. Does there exist a $O(\operatorname{poly}\log n)$ space pseudo-deterministic algorithm for ℓ_2-norm estimation?

Multi-pass streaming lower bounds

All the streaming lower bounds we prove are in the single pass model, i.e., where the algorithm receives the stream exactly once. How do these lower bounds extend to the multi-pass model, where the algorithm receives the stream multiple times? All of the pseudo-deterministic streaming lower bounds in this paper do not even extend to 2-pass streaming algorithms.

1.4 Table of complexities

In the below table, we outline the known space complexity of various problems considered in our work.

Table 1 Table of space complexities.

Problem	Randomized	Deterministic	Pseudo-deterministic
Morris Counters	$\Theta(\log\log n)$	$\Theta(\log n)$	$O(\log n)$, $\Omega(\log\log n)$
Find-Duplicate	$\Theta(\log n)$	$\Theta(n)$	$\widetilde{\Theta}(n)$
ℓ_2-approximation (streaming)	$\Theta(\log n)$	$\widetilde{\Theta}(n)$	$\widetilde{\Theta}(n)$
ℓ_2-approximation (sketching)			$\widetilde{O}(n)$, $\widetilde{\Omega}(\log n)$
Find-Nonzero-Row	$\widetilde{\Theta}(1)$	$\widetilde{\Theta}(nd)$	$\widetilde{\Theta}(n)$

2 Preliminaries

A randomized algorithm is called *pseudo-deterministic* if for every valid input x, when running the algorithm twice on x, the same output is obtained with probability at least $2/3$. Equivalently (up to amplification of error probabilities), one can think of an algorithm as pseudo-deterministic if for every input x, there is a unique value $f(x)$ such that with probability at least $2/3$ the algorithm outputs $f(x)$ on input x

▶ **Definition 8** (Pseudo-deterministic). *A (randomized) algorithm A is called pseudo-deterministic if for all valid inputs x, the algorithm A satisfies*

$$\Pr_{r_1,r_2}[A(x,r_1) = A(x,r_2)] \geq 2/3.$$

An extension of pseudo-determinism is that of k-entropy randomized algorithms [16]. Such algorithms have the guarantee that for every input x, the distribution $A(x,r)$ (over a random choice of randomness r) has low entropy, in particular bounded by k.

Another extension of pseudo-determinism is that of m-pseudo-deterministic algorithms, from [10]. Intuitively speaking, any algorithm is k-pseudo-deterministic if for every valid input, with high probability the algorithm outputs one of k options (so, a 1-pseudo-deterministic algorithm is the same as a standard pseudo-deterministic algorithm, since it outputs the one unique option with high probability):

▶ **Definition 9** (k-pseudo-deterministic). *We say that an algorithm A is k-pseudo-deterministic if for all valid inputs x, there is a set $S(x)$ of size at most k, such that $\Pr_r[A(x,r) \in S(x)] \geq \frac{k+1}{k+2}$*

For the purposes of this work, we define a simple notion that we call a *k-concentrated algorithm.*

▶ **Definition 10.** *We say that an algorithm A is k-concentrated if for all valid inputs x, there is some output $F(x)$ such that $\Pr_r[A(x,r) = F(x)] \geq \frac{1}{k}$.*

The reason for making this definition is that any $\log k$-entropy randomized algorithm, and any $(k+2)$-pseudo-deterministic algorithm is k-concentrated. Thus, showing an impossibility result for k-concentrated algorithms also shows an impossibility result for $\log k$-entropy and $(k+2)$-pseudo-deterministic algorithms. Indeed, in this work, we use space lower bounds against k-concentrated algorithms to simultaneously conclude space lower bounds against low entropy and multi-pseudo-deterministic algorithms.

▶ **Definition 11.** *A turnstile streaming algorithm is one where there is a vector v, and the input is a stream of updates of the form "increase the i^{th} coordinate of v by r" or "decrease the i^{th} coordinate of v by r'". The goal is to compute something about the final vector, after all of the updates.*

We use a pseudorandom generator for space-bounded computation due to Nisan [29], which we recap below.

▶ **Theorem 12.** *There is a function* $G : \{0,1\}^{s \log r} \to \{0,1\}^r$ *such that*

1. *Any bit of* $G(x)$ *for any input* x *can be computed in* $O(s \log r)$ *space.*
2. *For all functions* f *from* $\{0,1\}^r$ *to some set* A *such that* f *is computable by a finite state machine on* 2^s *states, the total variation distance between the random variables* $f(\boldsymbol{x})$ *and* $f(G(\boldsymbol{y}))$ *where* $\boldsymbol{x}$ *is uniformly drawn from* $\{0,1\}^r$ *and* $\boldsymbol{y}$ *is uniformly drawn from* $\{0,1\}^{s \log r}$ *is at most* 2^{-s}.

3 Find-Duplicate: Pseudo-deterministic lower bounds

Consider the following problem: the input is a stream of $3n/2$ integers between 1 and n. The goal is to output a number k which appears at least twice in the stream. Call this problem Find-Duplicate. Recall that this problem has been considered in the past literature, specifically in [14, 22, 23], where upper and lower bounds for randomized algorithms have been shown.

Indeed, we know the following is true from [14, 22].

▶ **Theorem 13.** Find-Duplicate *has an algorithm which uses* $O(\text{poly} \log n)$ *memory and succeeds with all but probability* $\frac{1}{\text{poly}(n)}$.

We formally define a *pseudo-deterministic streaming algorithm* and show a pseudo-deterministic lower bound for Find-Duplicate to contrast with the randomized algorithm from Theorem 13.

▶ **Definition 14** (Pseudo-deterministic Streaming Algorithm)**.** *A pseudo-deterministic streaming algorithm is a (randomized) streaming algorithm* A *such that for all valid input streams* $s = \langle x_1, \ldots, x_m \rangle$*, the algorithm* A *satisfies* $\Pr_{r_1,r_2}[(A(x,r_1) = A(x,r_2)] \geq 2/3$.

One can also think of a pseudo-deterministic streaming algorithm as an algorithm A such that for every valid input stream s, there exists some valid output $f(s)$ such that the algorithm A outputs $f(s)$ with probability at least $2/3$ (one would have to amplify the success probability using repetition to see that this alternate notion is the same as the definition above).

▶ **Definition 15** (Find-Duplicate)**.** *Define* Find-Duplicate *to be the streaming problem where the input is a stream of length* $3n/2$ *consisting of up to* n*, and the output must be an integer which has occured at least twice in the string.*

▶ **Theorem 16.** Find-Duplicate *has no pseudo-deterministic algorithm with memory* $o(n)$.

Proof Overview

In order to prove Theorem 16, we introduce two communication complexity problems – One-Way-Find-Duplicate and One-Way-Partial-Recovery:

In the One-Way-Find-Duplicate problem, Alice has a list of $3n/4$ integers between 1 and n, and so does Bob. Alice sends a message to Bob, after which Bob must output an integer which is in both Alice's and Bob's list. Formally:

▶ **Definition 17** (One-Way-Find-Duplicate)**.** *Define* One-Way-Find-Duplicate *to be the one-way communication complexity problem where Alice has input* $S_A \subseteq [n]$ *and Bob has input* $S_B \subseteq [n]$*, where* $|S_A|, |S_B| \geq 3n/4$*. The goal is for Bob to output an element in* $S_A \cap S_B$.

The idea is that one can reduce One-Way-Find-Duplicate to Find-Duplicate. So, our new goal will be to show that One-Way-Find-Duplicate requires high communication. To do so, we will show that it is possible to reduce a different problem, denoted One-Way-Partial-Recovery(defined below), to One-Way-Find-Duplicate. Informally, in the One-Way-Partial-Recovery problem, Alice has a list of $3n/4$ integers between 1 and n. Bob does not have an input. Alice sends a message to Bob, after which Bob must output $n/10$ distinct elements which are all in Alice's list. Formally:

▶ **Definition 18** (One-Way-Partial-Recovery)**.** *Define* One-Way-Partial-Recovery *to be the one-way communication complexity problem where Alice has input $S_A \subseteq [n]$ and Bob has no input. The goal is for Bob to output a set S satisfying $S \subseteq S_A$ and $|S| \geq n/10$.*

We will show in Claim 19 that a low memory pseudo-deterministic algorithm for Find-Duplicate implies a low-communication pseudo-deterministic algorithm for One-Way-Find-Duplicate, and in Claim 20 that a low-communication pseudo-deterministic algorithm for One-Way-Find-Duplicate implies a low communication algorithm for One-Way-Partial-Recovery. Finally, in Claim 21, we show that One-Way-Partial-Recovery cannot be solved with low communication. Combining the claims yields Theorem 16.

Proof of Theorem 16.

▷ Claim 19. A pseudo-deterministic algorithm for Find-Duplicate with space S and success probability p implies a pseudo-deterministic communication protocol for One-Way-Find-Duplicate with communication S and success probability at least p.

Proof. To prove the above claim, we construct a protocol for One-Way-Find-Duplicate from a streaming algorithm for Find-Duplicate. Given an instance of One-Way-Find-Duplicate, Alice can stream her input set of integers in increasing order, and simulate the streaming algorithm for Find-Duplicate. Then, she sends the current state of the algorithm (which is at most S bits) to Bob, who continues the execution of the streaming algorithm. At the end, the streaming algorithm outputs a repetition with probability p, which means the element showed up in both Alice and Bob's lists. Note that for a given input to Alice and Bob, Bob outputs a unique element with high probability because the streaming algorithm is pseudo-deterministic. ◁

▷ Claim 20. A pseudo-deterministic one-way communication protocol for One-Way-Find-Duplicate with S communication and failure probability $O\left(\frac{1}{n^2}\right)$ implies a pseudo-deterministic communication protocol for One-Way-Partial-Recovery with S communication and $O\left(\frac{1}{n}\right)$ failure probability.

Proof. We will show how to use a protocol for One-Way-Find-Duplicate to solve the instance of One-Way-Partial-Recovery.

Suppose we have an instance of One-Way-Partial-Recovery. Alice sends the same message to Bob as if the input was an instance of One-Way-Find-Duplicate, which is valid since in both of these problems, Alice's input is a list of length $3n/4$ of integers between 1 and n.

Now, Bob's goal is to use the message sent by Alice to recover $n/10$ elements of Alice. Let X be the (initially empty) set of elements of Alice's input that Bob knows and let B be a set of $3n/4$ elements in $\{1, \ldots, n\}$ disjoint from X, where we initially set B to $\{1, 2, \ldots, n\}$. While the size of X is less than $n/10$, Bob simulates the protocol of One-Way-Find-Duplicate with Alice's message and input B. This will result in Bob finding a single element x in Alice's input that is (i) in B, and (ii) not in X. Bob adds x to X, and deletes x from B. Once the size of X is $n/10$, Bob outputs X.

If Alice has the set A as her input, define $f_A(B)$ to be the output which the pseudo-deterministic algorithm for One-Way-Find-Duplicate outputs with high probability when Alice's input is A and Bob's input is B. Now, set $B_0 = \{1, 2, \ldots, n\}$, and $B_i = B_{i-1} \setminus \{f_A(B)\}$. Note that these B_i (for $i = 0$ through $n/10$) are the sets which, assuming the pseudo-deterministic algorithm never errs during the reduction (where we say the algorithm errs if it does not output the unique element which is guaranteed to be output with high probability), Bob will use as his inputs for the simulated executions of One-Way-Find-Duplicate. The pseudo-deterministic algorithm does not err on any of the B_i except with probability at most $1/n$, by a union bound. If Bob succeeds on all of the B_i, that means that the sequence of inputs which will be his inputs for the simulated executions of One-Way-Find-Duplicate are indeed $B_0, B_1, \ldots, B_{n/10}$. So, since we have shown with high probability the algorithms succeeds on all of the B_i, and therefore with high probability the B_i are also Bob's inputs for the simulated executions of One-Way-Find-Duplicate, we see that with high probability Bob will succeed on all of the $n/10$ inputs he tries to simulate executions of One-Way-Find-Duplicate with.

Note that we used the union bound over all the B_i for $i = 1$ through $n/10$. All of these B_i are a function of A. In particular, notice that by definition, the B_i do not depend on the randomness chosen by Alice. ◁

▷ Claim 21. Every pseudo-deterministic One-Way-Partial-Recovery protocol which succeeds with probability at least $\frac{2}{3}$ requires $\Omega(n)$ bits of communication.

Proof. We prove this lower bound by showing that a protocol for One-Way-Partial-Recovery can be used to obtain a protocol with exactly the same communication for the problem where Alice is given a string x in $\{0,1\}^{Cn}$ as input, she sends a message to Bob, and Bob must exactly recover x from Alice's message with probability at least $2/3$. This problem has a lower bound of $\Omega(n)$ bits of communication.

Suppose there exists a pseudo-deterministic algorithm for One-Way-Partial-Recovery. Given such a pseudo-deterministic protocol that succeeds with probability at least $2/3$, there is a function F such that $F(S)$ (a set with $n/10$ elements) is Bob's output after the protocol with probability at least $2/3$ when Alice is given S as input.

We will construct sets $S_1, \ldots, S_t$ to be subsets of $[n]$ of size $3n/4$ such that for any $i \neq j$, $F(S_i)$ is not a subset of S_j. To do so, we use the probabilistic method: set $S_1, \ldots, S_t$ be random subsets of $[n]$ of size $3n/4$. The probability that $F(S_i)$ is contained S_j for fixed $i \neq j$ is at most $\left(\frac{3}{4}\right)^{n/10}$. Thus, by a union bound, the probability that for any $i \neq j$, $F(S_i)$ is contained S_j is at most $t^2 \left(\frac{3}{4}\right)^{n/10}$, a quantity which is strictly less than 1 when t is $\left(\frac{4}{3}\right)^{n/100}$, so $S_1, \ldots, S_t$ satisfying the desired guarantee exist.

Alice and Bob can (ahead of time) agree on an encoding of $\lfloor \log t \rfloor$-bit strings that is an injective function G from $\{0,1\}^{\lfloor \log t \rfloor}$ to $\{S_1, \ldots, S_t\}$. Now, if Alice is given a $\lfloor \log t \rfloor$-bit string x as input, she can send a message to Bob according to the pseudo-deterministic protocol for One-Way-Partial-Recovery by treating her input as $G(x)$. Bob then recovers $F(G(x))$ with probability at least $2/3$, and can use it to recover $G(x)$ since there is unique S_i in which $F(G(x))$ is contained. Since G is injective, Bob can also recover x with probability $2/3$.

This reduction establishes a lower bound of $\Omega(\lfloor \log t \rfloor)$ on the pseudo-deterministic communication complexity of One-Way-Partial-Recovery, which is an $\Omega(n)$ lower bound. ◁

Combining Claim 19, Claim 20 and Claim 21 completes the proof of Theorem 16. ◀

It is worth noting that the problem has pseudo-deterministic algorithms with sublinear space if one allows multiple passes through the input. Informally, a p-pass streaming algorithm is a streaming algorithm which, instead of seeing the stream only once, gets to see the stream p times.

▷ Claim 22. There is a p-pass deterministic streaming algorithm that uses $\widetilde{O}(n^{1/p})$ memory for the Find-Duplicate problem.

Proof. At the start of t-th pass, the algorithm maintains a candidate interval I of width $n^{1-(t-1)/p}$ from which it seeks to find a repeated element. At the very beginning, this candidate interval is $[1, n]$. In the t-th pass, first partition the interval into $n^{1/p}$ equal sized intervals $I'_1, \ldots, I'_{n^{1/p}}$, each of whose width (the width of an interval $[a, b]$ is $b - a$) is $n^{1-t/p}$ and count the number of elements of the stream that lie in each such subinterval – this count must exceed the width of at least one subinterval I'_t. Update I to I'_t and proceed to the next pass. After p passes, this interval will contain at most 1 integer. ◁

4 Entropy Lower Bound for Find-Duplicate

▶ **Theorem 23.** *Every zero-error randomized algorithm for* Find-Duplicate *that is* $\frac{n}{s}$*-concentrated must use* $\Omega\left(\frac{s}{\log n}\right)$ *space.*

By *zero error*, we mean that the algorithm never outputs a number k which is not repeated. With probability one it either outputs a valid output, or $\perp$.

Proof. We use a reduction similar to that of the pseudo-deterministic case (cf. Proof of Claim 20). Using the exact same reduction from the proof of Claim 19, we get that a $\frac{n}{s}$-concentrated streaming algorithm for Find-Duplicate using T space must give us a $\frac{n}{s}$-concentrated protocol for One-Way-Find-Duplicate with communication complexity T. If we can give a way to convert such a protocol for One-Way-Find-Duplicate into an $O\left(\frac{Tn \log n}{s}\right)$-communication protocol for One-Way-Partial-Recovery, the desired lower bound on T follows from the lower bound on communication complexity of One-Way-Partial-Recovery from Claim 21. We will now show how to make such a conversion by describing a protocol for One-Way-Partial-Recovery.

Alice sends Bob $\Theta(n \log n/s)$ messages according to the protocol for One-Way-Find-Duplicate (that is, she simulates the protocol for One-Way-Find-Duplicate a total of $\Theta(n \log n/s)$ times). Bob's goal is to use these $\Theta(n \log n/s)$ messages to recover at least $n/10$ input elements of Alice. Towards this goal, he maintains a set of elements recovered so far, X (initially empty), and a family of "active sets" $\mathcal{B}$ (initially containing the set $\{1, 2, \ldots, n\}$). While the size of X is smaller than $n/10$, Bob simulates the remainder of the One-Way-Find-Duplicate protocol on every possible pair (B, M) where B is a set in $\mathcal{B}$ and M is one of the messages of Alice. For each such pair (B, M) where the protocol is successful in finding a duplicate element x, Bob adds x to X, removes B from $\mathcal{B}$ and adds $B \setminus \{x\}$ to $\mathcal{B}$.

We now wish to prove that this protocol indeed lets Bob recover $n/10$ elements of Alice. Suppose Alice has input A. For each set S, define $f_A(S)$ be an element of $A \cap S$ that has probability at least s/n of being outputted by Bob on input S at the end of a One-Way-Find-Duplicate protocol. Let $S_0 := \{1, 2, \ldots, n\}$ and $S_i := S_{i-1} \setminus \{f_A(S_i)\}$ be defined for $0 \leq i \leq n/10$. Note that S_i are predetermined: it is a function of Alice's input (and, in particular, not a function of the randomness she uses when choosing her messages). For a fixed i, the probability of failure to recover $f_A(S_i)$ from any of Alice's messages is at most

$1/n^2$. A failure to fill in X with $n/10$ elements implies that for some i, Bob failed to recover $f_A(S_i)$ from all of Alice's messages. The probability that such a failure happens for a specific i is at most $(1-s/n)^{\Theta(n\log n/s)}$. By setting the constant in the Θ to be large enough, we can have this be at most $\frac{1}{n^2}$, and so by a union bound the probability that there is an i such that $f_A(S_i)$ is not recovered by Bob is at most $1/n$.

Thus, we obtain a protocol for One-Way-Partial-Recovery with communication complexity $O(Tn\log n/s)$, and so $T \le s/\log n$, completing the proof. ◀

We obtain the following as immediate corollaries:

▶ **Corollary 24.** *Any zero-error* $\log\left(\frac{n}{s}\right)$*-entropy randomized algorithm for* Find-Duplicate *must use* $\Omega\left(\frac{s}{\log n}\right)$ *space.*

▶ **Corollary 25.** *Any zero-error* $O\left(\frac{n}{s}\right)$*-pseudo-deterministic algorithm for* Find-Duplicate *must use* $\Omega\left(\frac{s}{\log n}\right)$ *space.*

Below we show that the above lower bound is tight up to log factors.

▶ **Theorem 26.** *For all* s*, there exists a zero-error randomized algorithm for* Find-Duplicate *using* $\widetilde{O}(s)$ *space (where* $\widetilde{O}$ *hides factors polylogarithmic in* n*) that is* $O\left(\frac{n}{s}\right)$*-concentrated.*

Proof. Define the following algorithm A for Find-Duplicate: pick a random number i in $[3n/2]$, then remember the i^{th} element a of the stream, and see if a appears again later in the stream. If it does, return x. Otherwise return $\perp$.

The $O\left(\frac{n}{s}\right)$-concentrated algorithm algorithm is as follows: Run $s\log n$ copies of Algorithm A independently (in parallel), and then output the minimum of the outputs.

We are left to show that this algorithm is indeed $O\left(\frac{n}{s}\right)$-concentrated.

Define f to be a function where $f(i)$ is the total number of times which i shows up in the stream, and define $g(i) = \max((f(i)-1), 0)$. Note that then, the probability that i is outputted by algorithm A is $g(i)/(3n/2)$, since i will be outputted if A chooses to remember one of the first $i-1$.

Consider the smallest a such that $\sum_{i=1}^{a} g(i) \ge n/(2s)$. We will show that the probability that the output is less than a with high probability. It will follow that the algorithm is s-concentrated, since of the $a-1$ smallest elements, at most $\sum_{i=1}^{a-1} g(i)$ outputs are possible (since if $g(i)=0$, then i is not a possible output). So, we will see that with high probability, one of at most $\sum_{i=1}^{a-1} g(i) + 1 \le n/(2s)$ outputs (namely, the valid outputs less than or equal to a) will be outputted with high probability. And hence, at least one of them will be outputted with probability at least $\frac{s}{n}$.

The probability that the output is at most a in a single run of algorithm A is $\frac{3n}{2}\sum_{i=1}^{a} g(i)$ $\ge 3/(4s)$. So, the probability that in $s\log n$ runs of algorithm, in at least one of them an element which is at most a is outputted is $1-(1-\frac{3}{4s})^{s\log n}$, which is polynomially small in n. Hence, with high probability an element which is at most a (and there are $n/(2s)$ valid outputs less than a) will be outputted. ◀

4.1 Getting Rid of the Zero Error Requirement

A downside of Theorem 23 is that it shows a lower bound only for zero-error algorithms. In this section, we strengthen the theorem by getting rid of that requirement:

▶ **Theorem 27.** *Every randomized algorithm for* Find-Duplicate *that is* $\frac{n}{s}$*-concentrated and errs with probability at most* $\frac{1}{n^2}$ *must use* $\tilde{\Omega}\left(s^{1-\epsilon}\right)$ *space (for all* $\epsilon > 0$*).*

Proof overview

We begin by outlining why the approach of Theorem 23 does not work without the zero-error requirement. Recall that the idea in the proof was to have Alice send many messages (for One-Way-Find-Duplicate) to Bob, and Bob simulates the One-Way-Find-Duplicate algorithm (using simulated inputs he creates for himself) using these messages to find elements in Alice's input.

The problem is that the elements we end up removing from Bob's simulated input[1] depend on Alice's messages, and therefore we can't use a union bound to bound the probability that the protocol failed for a certain simulated input. So, we want the elements we remove from Bob's fake input not to depend on the inputs Alice sent. One idea to achieve this is to have Alice send a bunch of messages (for finding a shared element), and then Bob will remove the element that gets output the largest number of times (by simulating the protocol with each of the many messages Alice sent). The issue with this is that if the two most common outputs have very similar probability, the outputted element depends not only on Alice's input, but also on the randomness she uses when choosing what messages to send to Bob. This makes it again not possible to use a union bound.

There are two new ideas to fix this issue. The first is to use the following "Threshold" technique: Bob will pick a random "threshold" T between $ks/(2n)$ and $ks/(4n)$ (where we wish to show a lower bound on n/s-concentrated algorithms, and k is the total number of messages Alice sends to Bob). He simulates the algorithm for One-Way-Find-Duplicate with all k messages Alice sent him, and gets a list L of k outputs. Then, he will consider the "real" output to be the lexicographically first output $y \in L$ where there are more than T copies of y in the list L (note that since the algorithm is n/s-concentrated, its very unlikely for no such element to exist).

Now, it follows that with high probability, the shared element does not really depend on the messages. This is because with all but probability approximately $1/\sqrt{ks/n}$, the threshold is far (more than $\sqrt{ks/n}\log^2 ks/n$ away) from the the frequency of every element in L. We note that we pick $\sqrt{ks/n}\log^2 ks/n$ since from noise we would expect to have the frequencies of elements in L change by up to $\sqrt{ks/n}\log^2 ks/n$, depending on the randomness of A. We want the threshold to be further than that from the expected frequencies, so that with high probability there will be no element which sometimes has frequency more than T and sometimes has frequency less than T, depending on Alice's messages (recall that the goal is to make the outputs depend as little as possible on Alice's messages, but to only depend on shared randomness and on Alice's input).

This is still not enough for us: we still cannot use a union bound, as $1/\sqrt{ks/n}$ fraction of the time Bob's output will depends on Alice's message (and not just her input). The next idea resolves this. What Alice will do is send $n/\sqrt{ks/n}$ additional pieces of information: telling Bob where the chosen thresholds are bad, and what threshold to use instead. We assume that we have shared randomness so Alice knows all of the thresholds that will be chosen by Bob (note heavy-recovery is hard, even in the presence of shared randomness, so the lower bound is sufficient with shared randomness). Now, Alice can tell for which executions there the threshold chosen will be too close to the likelihood of an element. So, Alice will send approximately $n/\sqrt{ks/n}$ additional pieces of information: telling Bob where the chosen thresholds are bad, and what threshold to use instead. By doing so, Alice has guaranteed that a path independent of her messages will be taken.

[1] recall that Bob simulates an input to the One-Way-Find-Duplicate problem, and then he repeatedly finds elements he shares with Alice, removes them from the "fake" input, and reconstructs a large fraction Alice's inputs

To recap, idea 1 is to use the threshold technique so that with probability $1 - 1/\sqrt{ks/n}$ what Bob does doesn't depend on Alice's messages (only on her input). Idea 2 is to have Alice tell Bob where these $1/\sqrt{ks/n}$ bad situations are, and how to fix them.

The total amount of information Alice sends (ignoring logs) is $\tilde{\Theta}(kb + n/\sqrt{ks/n})$, (where b is the message size we are assuming exists for pseudo-deterministically finding a shared element, and k is the number of messages Alice sends). The factor $n/\sqrt{ks/n}$ follows since $1/\sqrt{ks/n}$ of the times, short messages will be sent to Bob due to a different threshold. A threshold requires $\log n$ bits to describe, which can be dropped since we are ignoring log factors. Setting $n/s \ll k \ll n/b$, we conclude that Alice sends a total of $\tilde{o}(n)$ bits. This establishes a contradiction, since we need $\tilde{\Theta}(n)$ bits to solve One-Way-Partial-Recovery. So, whenever $s = \tilde{\omega}(b)$, we can pick a k such that we get a contradiction.

Proof. Below we write the full reduction written out as an algorithm for One-Way-Find-Duplicate.

- Alice Creates $k = n/\sqrt{sb}$ messages for One-Way-Find-Duplicate, and sends them to Bob (Call these messages of type A).
- Additionally, Alice looks at the thresholds in the shared randomness. every time there is a threshold that is close (within $\sqrt{ks/n}\log^2 ks/n$) of the expected number of times a certain y will be outputted on the corresponding input (that is, for each fake input Bob will try, Alice checks if the probability of outputting some y is close to T – to be precise, say she checks if its probability of being outputted, assuming a randomly chosen message by Alice, is close to T), she sends a message to Bob informing him about the bad threshold, and suggests a good threshold to be used instead (call these messages of type B). Notice that these messages do not depend on the messages of type A that Alice sends, and that each such message is of size $O(\log n)$.
- Bob sets B to be the simulated input $\{1, ..., n\}$
- Bob uses each of the messages of type A that Alice sent, along with B, to construct a list of outputs.
- Bob looks at the shared randomness to find a threshold T (if Alice has informed him it is a bad threshold, use the threshold Alice suggests instead), and consider the lexicographically minimal output y that is contained in the multiset more than T times.
- Bob removes y from the fake input and repeat the last three steps of the algorithm (this time using a new threshold).

▷ Claim 28. The above protocol solves One-Way-Partial-Recoverywith high probability using $\tilde{o}(n)$ bits.

Proof. First we show that the total number of bits communicated is $\tilde{o}(n)$. Notice that the total number of messages of type A that are sent is $n/\sqrt{sb}$. We assume that each of these is of size at most b, giving us a total of $n\sqrt{b}/\sqrt{s}$ bits sent in messages of type A. Under the assumption that $b = \tilde{o}(n)$, we see that this is $\tilde{o}(n)$ total bits for messages of type A.

We now count the total number of bits communicated in messages of type B. Each message of type B is of size $O(\log n)$ (it is describing a single element, and a number corresponding to which execution the message is relevant for, each requiring $O(\log n)$ bits). So, we wish to show that with high probability the total number of messages of type B is $\tilde{o}(n)$. The total number of messages of type B that will be sent is $O(\frac{n}{\sqrt{ks/n}})$, since for every input, the probability that the randomly chosen threshold (which is sampled using public randomness) is more than $\sqrt{ks/n}\log^2 ks/n$ away from the frequency of every output is $O(\frac{1}{\sqrt{ks/n}})$. Note that $\frac{n}{\sqrt{ks/n}} = \tilde{o}(n)$ since $ks = n\sqrt{\frac{s}{b}}$, and we assume $b = \tilde{o}(s)$.

We are now left to show the protocol correctly solves One-Way-Partial-Recovery with high probability. We will first show that, after fixing Alice's input and the public randomness, with high probability there will be a single sequence of inputs that Bob will try that will occur with high probability (that is, there is a sequence of y's that Bob goes through with high probability). To do this, consider a certain input that Bob tries. We will bound the probability that there are two values y and y' such that both y and y' have probability at least $\frac{1}{n}$ of being outputted. Suppose there exists two such y and y' that means that at least one of them (say y, without loss of generality) has to be the output of more than T of the k executions with probability more than $\frac{1}{n}$, but less than $\frac{n-1}{n}$. Additionally, we know that the expected number of times that y will be outputted of the k times is more than $\sqrt{ks/n}\log^2 ks/n$ away from T (otherwise Alice will pick a different value of T such that this will be true, and send that value to Bob in a message of type B). However, the probability of being more than $\sqrt{ks/n}\log^2 ks/n = \Theta((\frac{s}{b})^{1/4}\log^2 s/b) = \Theta(n^{\varepsilon/4}\log^2 n)$ away from the expectation, by a Chernoff bound, is (asymptotically) less than $\frac{1}{n}$.

Notice also, that by the assumption that the algorithm in n/s-concentrated, there will always be an output $y_{\max}$ which is expected to appear at least $\frac{s}{n}$ of the time. Also, since the threshold T is at most $ks/2n$, the probability that $y_{\max}$ appeared fewer than T times is exponentially low in $ks/n = \sqrt{s/b} = \tilde{\Theta}(n^{\epsilon/2})$, and so with high probability there will always exist a y which was outputted on more than T of the executions, so in the second to last step, the multiset will always have an element that appears at least T_1 times.

Hence, by a union bound over all inputs that Bob tries, with high probability there will be a single sequence of inputs which Bob goes through (which depends only only on the public thresholds and Alice's input).

We will show that each y generated by Bob is an element in Alice's input with high probability. Notice that the y that Bob picks has appeared more than T times out of k, where T is at least $ks/(4n)$. If y is not a valid output then its probability of being outputted is $\frac{1}{n^2}$. The probability it is outputted at least once is at most $\frac{k}{n^2} \leq \frac{1}{n}$. Taking a union bound over the inputs that Bob tries (of which there are $n/10$), we get that the probability that there is an invalid y at any point is at most $1/10$. So, with probability $9/10$, no invalid y is ever outputted. ◁

◀

5 Entropy lower bounds for finding a support element

Consider the turnstile model of streaming, where a vector $z \in \mathbb{R}^n$ starts out as 0 and receives updates of the form "increment z_i by 1" or "decrement z_i by 1", and the goal of outputting a nonzero coordinate of z. This is a well studied problem and a common randomized algorithm to solve this problem in a small amount of space is known as ℓ_0 sampling [8]. ℓ_0 sampling uses polylogarithmic space and outputs a uniformly random coordinate from the support of z. A natural question one could ask is whether the output of any low space randomized algorithm is necessarily close to uniform, i.e., has high entropy. We answer this affirmatively and show a nearly tight tradeoff between the space needed to solve this problem and the entropy of the output of a randomized algorithm under the assumption that the algorithm is not allowed to output anything outside the support[2]

[2] We note that using similar ideas to those in Subsection 4.1, the zero error requirement could be removed. We omit this adaptation since it is very similar to that of Subsection 4.1.

▶ **Theorem 29.** *Every zero-error randomized algorithm for* Find-Support-Elem *that is* $\frac{n}{s}$*-concentrated must use* $\Omega\left(\frac{s}{\log n}\right)$ *space.*

We only provide a sketch of the proof and omit details since they are nearly identical to the proof of Theorem 23.

Proof Sketch. Let $\mathcal{A}$ be such an algorithm that uses T space. Just like the proof of Theorem 23, the way we show this lower bound is by illustrating that $\mathcal{A}$ can be used to obtain an $O\left(\frac{Tn\log n}{s}\right)$-communication protocol for One-Way-Partial-Recovery, which combined with Claim 21 yields the desired result.

For every element a in Alice's input set A, she streams "increment z_a by 1" and runs $\Theta\left(\frac{n}{s}\log n\right)$ independent copies of $\mathcal{A}$ on the input. She then sends the states of each these independent runs of $\mathcal{A}$ to Bob, which is at most $\frac{Tn\log n}{s}$ bits, to Bob. Bob maintains a set of states $\mathcal{M}$, initially filled with all of Alice's messages. While he has not yet recovered $n/10$ elements, Bob picks a message $M \in \mathcal{M}$ and recovers x in A using algorithm $\mathcal{A}$. And for each $M \in \mathcal{M}$, Bob resumes $\mathcal{A}$ on state M and streams "decrement z_x by 1" and adds the new state to $\mathcal{M}$, and deletes M from $\mathcal{M}$.

The proof of correctness for why Bob indeed eventually recovers $n/10$ elements of A is identical to that in the proof of Theorem 23, thus giving a protocol for One-Way-Partial-Recovery and proving the statement. ◀

We can immediately conclude the following.

▶ **Corollary 30.** *Any zero-error* $\log\left(\frac{n}{s}\right)$*-entropy randomized algorithm for* Find-Support-Elem *must use* $\Omega\left(\frac{s}{\log n}\right)$ *space.*

▶ **Corollary 31.** *Any zero-error* $O\left(\frac{n}{s}\right)$*-pseudo-deterministic algorithm for* Find-Support-Elem *must use* $\Omega\left(\frac{s}{\log n}\right)$ *space.*

This lower bound is also tight up to polylogarithmic factors due to an algorithm nearly identical to the one from Theorem 26. In particular, we have:

▶ **Theorem 32.** *For all* s*, there exists a zero-error randomized algorithm for* Find-Duplicate *using* $O(s)$ *space that is* $O\left(\frac{n}{s}\right)$*-concentrated.*

6 Space complexity of pseudo-deterministic ℓ_2-norm estimation

In this section, we once again consider the pseudo-deterministic complexity of ℓ_2 norm estimation in the *sketching model*. The algorithmic question here is to design a distribution $\mathcal{D}$ over $s \times n$ matrices along with a function $f : \mathbb{R}^s \to \mathbb{R}$ so that for any $x \in \mathbb{R}^n$:

$$\Pr_{\boldsymbol{S}\sim\mathcal{D}}[f(\boldsymbol{S}x) \notin \left[\frac{1}{\alpha}\|x\|_2, \alpha\|x\|_2\right] \leq \frac{1}{\text{poly}(n)}.$$

Further, we want $f(\boldsymbol{S}x)$ to be a pseudo-deterministic function; i.e., we want $f(\boldsymbol{S}x)$ to be a unique number with high probability.

▶ **Theorem 33.** *The pseudo-deterministic sketching complexity of* ℓ_2 *norm estimation is* $\Omega(n)$*.*

The following query problem is key to our lower bound.

▶ **Definition 34** (ℓ_2 adaptive attack)**.** *Let $\alpha > 0$ be some constant. Let S be an $s \times n$ matrix with real-valued entries and $f : \mathbb{R}^s \to \mathbb{R}$ be some function. Now, consider the query model where an algorithm is allowed to specify a vector $x \in \mathbb{R}^n$ as a query and is given $f(Sx)$ as a response. The goal of the algorithm is to output y such that*

$$f(Sy) \notin \left[\frac{1}{\alpha}\|y\|_2, \alpha\|y\|_2\right]$$

*in as few queries as possible. We call this algorithmic problem the ℓ_2-*adaptive attack problem.

We use a theorem on adaptive attacks on ℓ_2 sketches proved in [17].

▶ **Theorem 35.** *There is a $\mathrm{poly}(n)$-query protocol to solve the ℓ_2 adaptive attack problem with probability at least $9/10$, i.e., the problem in Definition 34 when $s = o(n)$.*

Proof of Theorem 33. Suppose $\mathcal{D}$ is a distribution over $s \times n$ sketching matrices and f is a function mapping $\mathbb{R}^s$ to $\mathbb{R}$ with the property that the pair $(\mathcal{D}, f)$ gives a pseudo-deterministic sketching algorithm for ℓ_2 norm estimation. Henceforth, we use $\boldsymbol{S}$ to denote a random matrix sampled from $\mathcal{D}$. Then there is a function $g : \mathbb{R}^n \to \mathbb{R}$ such that:

1. g is an α-approximation of the ℓ_2 norm.
2. On every input x, $f(\boldsymbol{S}x) = g(x)$ with probability at least $1 - \frac{1}{n^c}$ for some constant c.

We will show that s must be $\Omega(n)$ by deducing a contradiction when $k = o(n)$. Let r be a parameter to be chosen later. Let $\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(r)}$ be the (random) sequence of vectors in $\mathbb{R}^n$ obtained by the adaptive query protocol from Theorem 35 based on responses $g(\boldsymbol{x}^{(0)}), \ldots, g(\boldsymbol{x}^{(r)})$ where $r = \mathrm{poly}(n)$, and let $\boldsymbol{y}$ be the (random) output of the protocol. Note that the guarantee that $r = \mathrm{poly}(n)$ hinges on assuming $s = o(n)$. From the guarantees of Theorem 35, for any fixed matrix B and function h such that $h(B\boldsymbol{x}^{(i)}) = g(\boldsymbol{x}^{(i)})$ for all i, it is true with probability at least $9/10$ that $h(B\boldsymbol{y}) \neq g(\boldsymbol{y})$. On the other hand, for any sequence of $r + 2$ fixed vectors $v_0, \ldots, v_{r+1}$, $f(\boldsymbol{S}v_i) = g(v_i)$ for all i with probability at least $1 - \frac{1}{\mathrm{poly}(n)}$. Call the event $\{f(\boldsymbol{S}\boldsymbol{x}^{(0)}) = g(\boldsymbol{x}^{(0)}), \ldots, f(\boldsymbol{S}\boldsymbol{x}^{(r)}) = g(\boldsymbol{x}^{(r)}), f(\boldsymbol{S}\boldsymbol{y}) = g(\boldsymbol{y})\}$ as $\mathcal{E}$. Let p_S be the probability density function of $\boldsymbol{S}$ and let p_T be the probability density function of $(\boldsymbol{x}^{(0)}, \ldots, \boldsymbol{x}^{(r)}, \boldsymbol{y})$. This results in the following two estimates of $\Pr[\mathcal{E}]$.

On the one hand,

$$\begin{aligned}\Pr[\mathcal{E}] &= \int_{\boldsymbol{S}} \Pr[\mathcal{E}|\boldsymbol{S}] p_S(\boldsymbol{S}) \\ &\leq \int_{\boldsymbol{S}} \frac{1}{10} p_S(\boldsymbol{S}) \\ &= \frac{1}{10},\end{aligned}$$

and on the other hand,

$$\begin{aligned}\Pr[\mathcal{E}] &= \int_{\boldsymbol{x}^{(0)},\ldots,\boldsymbol{x}^{(r)},\boldsymbol{y}} \Pr[\mathcal{E}|\boldsymbol{x}^{(0)}, \ldots, \boldsymbol{x}^{(r)}, \boldsymbol{y}] p_T(\boldsymbol{x}^{(0)}, \ldots, \boldsymbol{x}^{(r)}, \boldsymbol{y}) \\ &\geq \int_{\boldsymbol{x}^{(0)},\ldots,\boldsymbol{x}^{(r)},\boldsymbol{y}} \left(1 - \frac{1}{\mathrm{poly}(n)}\right) p_T(\boldsymbol{x}^{(0)}, \ldots, \boldsymbol{x}^{(r)}, \boldsymbol{y}) \\ &= 1 - \frac{1}{\mathrm{poly}(n)}.\end{aligned}$$

The contradiction arises since $\Pr[\mathcal{E}]$ cannot simultaneously be at least $1 - \frac{1}{\mathrm{poly}(n)}$ and at most $\frac{1}{10}$, and hence s cannot be $o(n)$. ◀

▶ **Corollary 36.** *For any constant $\delta > 0$, any $(2-\delta)$-concentrated sketching algorithm that where the sketching matrix is $s \times n$ can be turned into a pseudo-deterministic one by running $\log n$ independent copies of the sketch and outputting the majority answer. Thus, as an upshot of Theorem 33 we obtain a lower bound of $\Omega\left(\frac{n}{\log n}\right)$ on $(2-\delta)$-concentrated algorithms for pseudo-deterministic ℓ_2-norm estimation in the sketching model.*

In contrast to Corollary 36 which says that $(2-\delta)$-concentrated algorithms for ℓ_2 estimation in the sketching model need near linear dimension, we show that there is an $O(\text{poly}\log n)$-dimension $(2+\delta)$-concentrated sketching algorithm to solve the problem, thus exhibiting a "phase transition".

▶ **Theorem 37.** *There is a distribution $\mathcal{D}$ over $s \times n$ matrices and a function $f : \mathbb{R}^s \to \mathbb{R}$ when $s = O(\text{poly}\log n)$ For every constant $\delta > 0$, there is an $O(\text{poly}(\log n, \log m))$-space $(2+\delta)$-concentrated sketching algorithm for ℓ_2-norm estimation.*

Proof. Let the true ℓ_2 norm of the input vector be r. Run the classic sketching algorithm of [2] for randomized ℓ_2 norm estimation with error $\min\{1/2^{20}, \varepsilon^4\}$ and failure probability $\frac{1}{\text{poly}(n)}$ where $(1+\varepsilon)$ is the desired approximation ratio. This uses a sketch of dimension $O(\text{poly}\log n)$. Now, we describe the function f we use. Take the output of the sketching algorithm of [2] and return the number obtained by zeroing out all its bits beyond the first $\max\{2\log\left(\frac{1}{\varepsilon}\right), 5\}$ significant bits.[3] First, the outputted number is a $(1+\varepsilon)$ approximation. Further, for each input, the output is one of two candidates with probability $1 - \frac{1}{\text{poly}(n)} > 1-\delta$ for every constant δ. This is because [2] produces a $(1+\varepsilon^4)$-approximation to r, and there are only two candidates for the $2\log\left(\frac{1}{\varepsilon}\right)$ most significant bits of any real number that lies in an interval $[(1-\varepsilon^4)r, (1+\varepsilon^4)r]$. ◀

7 Pseudo-deterministic Upper Bounds

7.1 Finding a nonzero row

Given updates to an $n \times d$ matrix A (where we assume $d \leq n$) that is initially 0 in a turnstile stream such that all entries of A are always in range $[-n^3, n^3]$, the problem Find-Nonzero-Row is to either output an index i such that the ith row of A is nonzero, or output `none` if A is the zero matrix.

▶ **Theorem 38.** *The randomized space complexity for* Find-Nonzero-Row *is $\widetilde{\Theta}(1)$, the pseudo-deterministic space complexity for* Find-Nonzero-Row *is $\widetilde{\Theta}(n)$, and the deterministic space complexity for* Find-Nonzero-Row *is $\widetilde{\Theta}(nd)$.*

Proof. We first will show a randomized $\widetilde{\Theta}(1)$ space algorithm for the problem, then we will show pseudo-deterministic upper and lower bounds, and then show the deterministic lower bound.

Randomized algorithm for Find-Nonzero-Row. A randomized algorithm for this problem is given below. Note that the version of the algorithm as stated below does not have the desirable $\widetilde{O}(1)$ space guarantee, but we will show how to use a pseudorandom generator of Nisan [29] to convert the below algorithm to one that uses low space.

[3] The parameters $1/2^{20}, 5$ and ε^4 are chosen purely for *safety* reasons.

1. Sample a random d-dimensional vector $\boldsymbol{x}$ where each entry is an independently drawn integer in $[-n^3, n^3]$ and store it.
2. Simulate a turnstile stream which maintains $A\boldsymbol{x}$. In particular, consider the n-dimensional vector y, which is initially 0, and for each update to A of the form "add Δ to A_{ij}", add $\Delta \boldsymbol{x}_j$ to y_i. We run an ℓ_0-sampling algorithm [8] on this simulated stream updating y, and return the output of the ℓ_0-sampler, which is close in total variation distance to a uniformly random element in the support of y.

In the above algorithm, step 1 is not low-space as stated. Before we give a way to perform step 1 in $\widetilde{O}(1)$ space, we prove the correctness of the above algorithm. Suppose A_i is a nonzero row of A, then let j be an index where A_i is nonzero. Suppose all coordinates of $\boldsymbol{x}$ except for the j-th coordinate have been sampled, there is at most one value C for $\boldsymbol{x}_j$ for which $\langle A_i, \boldsymbol{x}\rangle$ is 0, and there is at most a $1/n^3$ probability that $\boldsymbol{x}_j$ equals C, which means if i is a nonzero row, then $(A\boldsymbol{x})_i$ is nonzero except with probability at most $1/n^3$. In fact, by taking a union bound over all nonzero rows we can conclude that the set of nonzero rows and the set of nonzero indices of $A\boldsymbol{x}$ are exactly the same, except with probability bounded by $1/n^2$.

Now we turn our attention to implementing step 1 in low space. Towards doing so we use Nisan's pseudorandom generator for space bounded computation in a very similar manner to [19].

Instead of sampling $3d \log n + 1$ bits to store $\boldsymbol{x}$, we sample and store a uniformly random seed $\boldsymbol{w}$ of length $O(\text{poly}\log(n, d))$ and add $\Delta G(\boldsymbol{w})_j$ to y_i when an update "add Δ to A_{ij}" is received, where G is the function from Theorem 12 that maps the random seed to a sequence $3d \log n + 1$ bits. To prove the algorithm is still correct if we use the pseudorandom vector $G(\boldsymbol{w})$ instead of the uniformly random vector $\boldsymbol{x}$, we must show that when A_i is nonzero, then $\langle A_i, G(\boldsymbol{w})\rangle$ is nonzero with probability at least $1 - O(1/n^3)$. Towards this, for a fixed d-dimensional vector q, consider the following finite state machine. The states are labeled by pairs (i, a) where i is in $\{0, 1, \ldots, d\}$ and a is in $[-n^6 d, n^6 d]$. The FSM takes a d-dimensional vector r as input, starts at state $(0, 0)$, and transitions from state (i, a) to $(i+1, a+q_{i+1} \cdot r_{i+1})$ until it reaches a state (d, ℓ). The FSM then outputs ℓ. This establishes that for a fixed q, the function $f(x) \coloneqq \langle q, x\rangle$ is computable by an FSM on $\text{poly}(d, n)$ states, and hence from Theorem 12, $f(\boldsymbol{x})$ and $f(G(\boldsymbol{w}))$ are $1/dn^6$ close in total variation distance, which means when A_i is nonzero, then $\langle A_i, G(\boldsymbol{w})\rangle$ is nonzero except with probability bounded by $O(1/n^3)$.

A pseudo-deterministic algorithm and lower bound for Find-Nonzero-Row. The pseudo-deterministic algorithm is very similar to the randomized algorithm from the previous section.

1. Sample a random d-dimensional vector $\boldsymbol{x}$ where each entry is an independently drawn integer in $[-n^3, n^3]$. Store $\boldsymbol{x}$ and maintain $A\boldsymbol{x}$.
2. Output the smallest index i such that $(A\boldsymbol{x})_i$ is nonzero.

Storing $\boldsymbol{x}$ takes $O(d \log n)$ space, and maintaining $A\boldsymbol{x}$ takes $O(n \log n)$ space. Recall from the discussion surrounding the randomized algorithm that the set of nonzero indices of $A\boldsymbol{x}$ and the set of nonzero rows were equal with probability $1 - 1/n^2$, which establishes correctness of the above pseudo-deterministic algorithm. The space complexity is thus $O((d + n) \log n)$, which is equal to $O(n \log n)$ from the assumption that $d \leq n$.

A pseudo-deterministic lower bound of $\widetilde{\Omega}(n)$ follows immediately from Corollary 31 since Find-Nonzero-Row specialized to the $d = 1$ case is the same as Find-Support-Elem.

Lower Bound for deterministic algorithms. An $\Omega(nd \log n)$ bit space lower bound for deterministic algorithms follows from a reduction to the communication complexity problem of EQUALITY. Alice and Bob are each given $nd \log n$ bit strings as input, which they interpret as $n \times d$ matrices, A and B respectively, where each entry is a chunk of length $\log n$. Suppose a deterministic algorithm $\mathcal{A}$ takes S bits of space to solve this problem. We will show that this can be converted to a S-bit communication protocol to solve EQUALITY. Alice runs $\mathcal{A}$ on a turnstile stream updating matrix X initialized at 0 by adding A_{ij} to X_{ij} for all (i, j) in $[n] \times [d]$. Alice then sends the S bits corresponding to the state of the algorithm to Bob and he continues running $\mathcal{A}$ on the updates "add $-B_{ij}$ to X_{ij}". $\mathcal{A}$ outputs `none` if and only if $A = B$ and thus Bob outputs the answer to EQUALITY depending on the output of $\mathcal{A}$. Due to a communication complexity lower bound of $\Omega(nd \log n)$ on EQUALITY, S must be $\Omega(nd \log n)$. ◀

7.2 Point Query Estimation and Inner Product Estimation

In this section, we give pseudo-deterministic algorithms that beat the deterministic lower bounds for two closely related streaming problems – point query estimation and inner product estimation.

Point Query Estimation

Given a parameter ε and a stream of m elements where each element comes from a universe $[n]$, followed by a query $i \in [n]$, output f_i' such that $|f_i - f_i'| \leq \varepsilon m$ where f_i is the frequency of element i in the stream.

Inner Product Estimation

Given a parameter ε and a stream of m updates to (initially 0-valued) n-dimensional vectors x and y in an insertion-only stream[4], output estimate e satisfying $|e - \langle x, y \rangle| < \varepsilon \cdot \|x\|_1 \cdot \|y\|_1$. In the above problems, we will be interested in the regime where $m \ll n$.

Our main result regarding a pseudo-deterministic algorithm for point query estimation is:

▶ **Theorem 39.** *There is an $O\left(\frac{\log m}{\varepsilon} + \log n\right)$-space pseudo-deterministic algorithm $\mathcal{A}$ for point query estimation with the following precise guarantees. For every sequence $s_1, \ldots, s_m$ in $[n]^m$, there is a sequence $f_1', \ldots, f_n'$ such that*

1. *For all i, $|f_i' - f_i| \leq \varepsilon m$ where f_i is the frequency of i in the stream.*
2. *Except with probability $1/m$, for all $i \in [n]$ $\mathcal{A}$ outputs f_i' on query i.*

We remark that the deterministic complexity of the problem is $\Omega(\frac{\log n}{\varepsilon}$ (see Theorem 44). Towards establishing Theorem 39, we recall two facts.

▶ **Theorem 40** (Misra–Gries algorithm [25])**.** *Given a parameter ε and a length-m stream of elements in $\{1, \ldots, d\}$, there is a deterministic $O\left(\frac{\log d + \log m}{\varepsilon}\right)$-space algorithm that given any query $s \in [d]$, outputs f_s' such that $|f_s' - f_s| \leq \varepsilon m$ where f_s is the number of occurrences of s in the stream. An additional guarantee that the algorithm satisfies is the following, which we call* permutation invariance*. Consider the stream*

$$s_1, s_2, \ldots, s_m$$

[4] A stream where only increments by positive numbers are promised.

and for any permutation $\pi : [d] \to [d]$, consider the stream

$$\pi(s_1), \pi(s_2), \ldots, \pi(s_m).$$

When the algorithm is given the first stream as input, let f'_s denote its output on query s, and when the algorithm is given the second stream as input, let $g'_{\pi(s)}$ denote its output on query $\pi(s)$. The algorithm has the guarantee that $f'_s = g'_{\pi(s)}$.

▶ **Theorem 41** (Pairwise independent hashing, [32, Corollary 3.34]). *Assume $d \ll n$. There is a pairwise independent hash function $h : [n] \to [d]$, which can be sampled using $O(\log n)$ random bits and also can be stored in $O(\log n)$ bits.*

Proof of Theorem 39. The algorithm is as follows.

- Sample a random pairwise independent hash function $h : [n] \to [m^3]$, which can be sampled and stored in $O(\log n)$ bits.
- Run the Misra–Gries algorithm with the following simulated stream as input: for each s streamed as input, stream $h(s)$ to the simulation.
- Given any query s, perform query $h(s)$ to the Misra–Gries algorithm running on the simulated stream, and return its output.

Let S be the collection of elements of $[n]$ that occur in the input stream $s_1, \ldots, s_m$. Assuming h maps S into $[m^3]$ without any collisions[5], it follows from the permutation invariance property of the Misra–Gries algorithm from Theorem 40 the output of the above algorithm on any query q is equal to $F(s_1, \ldots, s_m, q)$ for a *fixed* function F. Thus if we show that h indeed maps S into $[m^3]$ injectively pseudo-determinism of the given algorithm would follow.

Given $i, j \in S$, due to pairwise independence of h, the probability that $h(i) = h(j)$ is equal to $1/m^3$. A union bound over all pairs of elements in S tells us that h is collision-free except with probability at most $1/m$, which implies that the above algorithm is indeed pseudo-deterministic. ◀

▶ **Theorem 42.** *There is a (weakly) pseudo-deterministic algorithm for inner product estimation that uses $O\left(\frac{\log m}{\varepsilon} + \log n\right)$ space.*

The algorithm for inner product estimation is based on point query estimation, and towards stating the algorithm we first state a known result that helps relate the two problems.

▶ **Lemma 43** (Easily extracted from the proof of [28, Theorem 1]). *Let x, y, x', y' be vectors such that $\|x - x'\|_\infty \le \varepsilon \|x\|_1$ and $\|y - y'\|_\infty \le \varepsilon \|y\|_1$. Now, let x'' (and respectively y'') denote x' with everything except the maximum $1/\varepsilon$ entries zeroed out. Then the following holds:*

$$|\langle x'', y'' \rangle - \langle x, y \rangle| \le \varepsilon \cdot \|x\|_1 \cdot \|y\|_1.$$

Proof of Theorem 42. Given a stream of updates to x and y, run two instances of the point query estimation algorithm from Theorem 39 – one for updates to x and one for updates to y. There are x' and y' that only depend on the stream such that

$$\|x - x'\|_\infty \le \varepsilon \cdot \|x\|_1 \quad \text{and} \quad \|y - y'\|_\infty \le \varepsilon \cdot \|y\|_1$$

and except with probability $O(1/m)$ both point query algorithms respond to any query i with x'_i (and y'_i respectively). Maintaining these two instances takes $O\left(\frac{\log m}{\varepsilon} + \log n\right)$ space.

[5] I.e. the restriction of h to domain S is an injective function.

Next, enumerate over elements of $[n]$ and for each $i \in [n]$ query both instances with i, and store the running max-$1/\varepsilon$ answers to queries to each instance along with the hashed identities of the indices of entries that are part of the running max. Storing the running max takes $O\left(\frac{\log m}{\varepsilon}\right)$ space, and storing a counter to enumerate over $[n]$ takes $\log n$ space. Thus, at the end of this routine, except with probability $O(1/m)$ our two lists are equal to $(x'_{i_1}, h(i_1)), \ldots, (x'_{i_{1/\varepsilon}}, h(i_{1/\varepsilon}))$ and $(y'_{j_1}, h(j_1)), \ldots, (y'_{j_{1/\varepsilon}}, h(j_{1/\varepsilon}))$ respectively where $x'_{i_1}, \ldots, x_{i_{1/\varepsilon}}$ are the max-$1/\varepsilon$ entries of x' and $y'_{j_1}, \ldots, y_{j_{1/\varepsilon}}$ are the max-$1/\varepsilon$ entries of y'.

Finally, if there is t, u such that $h(i_t) = h(i_u)$ or $h(j_t) = h(j_u)$, return "fail"; otherwise output

$$\sum_{\ell \in \{h(i_t)\}_{t=1,\ldots,1/\varepsilon} \cap \{h(j_t)\}_{t=1,\ldots,1/\varepsilon}} x'_\ell y'_\ell.$$

With probability at least $1 - 2/m$, the above quantity is equal to $\langle x'', y'' \rangle$ from Lemma 43, which lets us conclude via Lemma 43 that the output is within $\varepsilon \cdot \|x\|_1 \cdot \|y\|_1$ of the true inner product. ◀

Finally, we remark that the following lower bounds can be proved for deterministic algorithms.

▶ **Theorem 44.** *Any deterministic algorithm for point query estimation and inner product estimation needs* $\Omega\left(\frac{\log n}{\varepsilon}\right)$ *space.*

Proof. We prove a lower bound for point query estimation via a reduction from EQUALITYin communication complexity. Alice encodes a $\log \binom{n}{1/(3\varepsilon)}$ bit string as a subset S of $[n]$ of size $1/(3\varepsilon)$ and runs the point query streaming algorithm on the input where she streams each element of this subset $3\varepsilon m$ times. She then sends the state of the algorithm to Bob, who can query every index in the universe and learn S (the element corresponding to the query is in S if and only if the response to the query is at least $2\varepsilon \cdot m$), decode S back to a $\log \binom{n}{1/(3\varepsilon)}$ and check if it is equal to his own input. The space lower bound from the theorem statement then follows since $\log \binom{n}{1/(3\varepsilon)} = \Omega\left(\frac{\log n}{\varepsilon}\right)$.

A space lower bound for inner product estimation follows from the lower bound for point query estimation since the latter is a special case of the former when x is the vector of frequencies and y is a standard unit vector e_i corresponding to query i. ◀

7.3 Retrieving a Basis of a Row-space

We now work in a "mixed" model, where an input $n \times d$ matrix A of rank-$\leq k$ is given to us via a sequence of updates in a turnstile stream, and each entry at all times in the stream can be represented by an $O(\log n)$-bit word. During this phase, there is an upper bound T on the number of bits of space an algorithm is allowed to use. In the "second phase", we are allowed to perform arbitrary computation and the goal is to output a basis for the row-span of A

We show a lower bound on T of $\widetilde{\Omega}(nd)$ for deterministic algorithms, and a pseudo-deterministic algorithm that uses $\widetilde{O}(\text{poly}(k) \cdot d)$ space in the streaming phase.

▶ **Theorem 45.** *Any deterministic streaming algorithm for* RECOVERBASIS *needs* $\widetilde{\Omega}(nd)$ *space.*

Proof. Suppose the matrix A is 0, then the algorithm would have to output the empty set. A T space streaming algorithm for this problem could be used to solve the communication complexity problem of equality EQUALITY using T bits of communication. In particular,

Alice and Bob could encode their respective inputs x and y as matrices M_x and M_y. Alice can then run the T-space algorithm on adding M_x in a turnstile stream, and send Bob the state of the algorithm. Bob can then resume running the algorithm from Alice's state on updates that subtract M_y. If Bob outputs the empty set, then $x = y$ and Bob outputs "yes". Otherwise, Bob outputs "no". ◀

While the deterministic complexity is $\widetilde{\Omega}(nd)$, there is a pseudo-deterministic streaming algorithm which uses only $\widetilde{O}(\text{poly}(k) + k \cdot d)$ in its streaming phase:

▶ **Theorem 46.** *There is a pseudo-deterministic algorithm for* RecoverBasis *that uses* $\widetilde{O}(\text{poly}(k) + k \cdot d)$ *space in its streaming phase, where the* $\widetilde{O}(\cdot)$ *hides factors of* $\text{poly}\log n$.

Towards giving a pseudo-deterministic algorithm, we first state a result about pseudorandom matrices that is a special case of [5, Lemma 3.4].

▶ **Theorem 47.** *There is a distribution* $\mathcal{D}$ *over* $m \times n$ *matrices where* $m = O(k \log n)$ *with* ± 1 *entries such that for any* $n \times m$ *matrix* U *with orthonormal columns and* $\boldsymbol{S} \sim \mathcal{D}$*, the following holds with probability* $1 - 1/\text{poly}(n)$*:*

$$\|U^T \boldsymbol{S}\boldsymbol{S}^T U - I\|_2 \leq 1/2.$$

Further, the rows of $\boldsymbol{S}$ *are independent and each row can be generated by a* $(k + \log n)$*-wise independent hash family.*

▶ **Theorem 48** (t-wise independent hash families [32, Corollary 3.34])**.** *There is a* t*-wise independent hash family* $\mathcal{H}$ *of functions from* $[n] \to \{\pm 1\}$ *such that sampling a uniformly random* h *from* $\mathcal{H}$ *can be done using a* $\text{poly}(\log n, t)$*-length random seed, and* $h(x)$ *for any* $x \in [n]$ *can be computed in* $\text{poly}(\log n, t)$ *time and space from the random seed used to sample it.*

As a consequence we have:

▶ **Corollary 49.** *Let* A *be a* $n \times d$ *matrix of rank* k *and let* $\mathcal{D}$ *be the distribution over* $O(k \log n) \times n$ *matrices from the statement of Theorem 47. Then, for* $\boldsymbol{S} \sim \mathcal{D}$*,* $\boldsymbol{S}A$ *has rank* k *with probability* $1 - 1/\text{poly}(n)$*.*

Proof. We start by writing A in its singular value decomposition $U\Sigma V^T$. Since A has rank k, U is a $n \times k$ matrix with orthonormal columns and ΣV^T surjectively maps $\mathbb{R}^d$ to $\mathbb{R}^k$. From Theorem 47, $\boldsymbol{S}A$ is also full rank, which means the collection of vectors

$$\{\boldsymbol{S}Ax : x \in \mathbb{R}^d\} = \{\boldsymbol{S}U\Sigma V^T x : x \in \mathbb{R}^d\} = \{\boldsymbol{S}Ux : x \in \mathbb{R}^k\}$$

is a k-dimensional space, and hence $\boldsymbol{S}A$ has rank k. ◀

Proof of Theorem 46. Begin by sampling $\boldsymbol{S} \sim \mathcal{D}$ via a seed $\boldsymbol{s}$ of length $O(\text{poly}(k) \cdot \text{poly}\log(n))$ from which entries of $\boldsymbol{S}$ can be efficiently computed where $\mathcal{D}$ is the distribution over matrices given by Corollary 49, and maintain the sketch $\boldsymbol{S}A$ in the stream.

The row-span of $\boldsymbol{S}A$ is exactly the same as that of A assuming the two matrices have equal rank, which happens with probability $1 - 1/\text{poly}(n)$.

$\boldsymbol{S}A$ is an $O(k) \times d$ matrix and each entry is a signed combination of at most n entries of A and hence there is a bit complexity bound of $\widetilde{O}(kd)$ on the space used to store $\boldsymbol{S}A$.

In the second phase (i.e., after the stream is over) of the algorithm, we first find an orthonormal basis Q for the row-span of $\boldsymbol{S}A$ and compute $\widetilde{\Pi}_A = QQ^T$. And finally, use a deterministic algorithm to compute the singular value decomposition $\widetilde{U}\widetilde{\Sigma}\widetilde{V}^T$ of $\widetilde{\Pi}_A$ and output the rows of $\widetilde{V}^T$.

The row-span of $\boldsymbol{S}A$ and A are equal except with probability $1/\text{poly}(n)$; assuming this happens, $\widetilde{\Pi}_A$ is exactly equal to Π_A, the unique projection matrix onto the row-span of A. Write Π_A in its singular value decomposition $U\Sigma V^T$. If $\widetilde{\Pi}_A = \Pi_A$, $\widetilde{V}^T$ is exactly equal to V^T. Since V^T is given by a deterministic function of A, and the output of the algorithm $\widetilde{V}$ is equal to V^T with high probability, our algorithm is pseudo-deterministic. ◀

References

1 Yuqing Ai, Wei Hu, Yi Li, and David P. Woodruff. New Characterizations in Turnstile Streams with Applications. In *31st Conference on Computational Complexity, CCC 2016, May 29 to June 1, 2016, Tokyo, Japan*, pages 20:1–20:22, 2016. doi:10.4230/LIPIcs.CCC.2016.20.

2 Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. *Journal of Computer and system sciences*, 58(1):137–147, 1999.

3 Alexandr Andoni, Robert Krauthgamer, and Krzysztof Onak. Streaming algorithms from precision sampling. *arXiv preprint*, 2010. arXiv:1011.1263.

4 Moses Charikar, Kevin Chen, and Martin Farach-Colton. Finding frequent items in data streams. *Theoretical Computer Science*, 312(1):3–15, 2004.

5 Kenneth L Clarkson and David P Woodruff. Numerical linear algebra in the streaming model. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 205–214. ACM, 2009.

6 Peter Dixon, A Pavan, and NV Vinodchandran. On Pseudodeterministic Approximation Algorithms. In *43rd International Symposium on Mathematical Foundations of Computer Science (MFCS 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

7 Philippe Flajolet. Approximate counting: a detailed analysis. *BIT Numerical Mathematics*, 25(1):113–134, 1985.

8 Gereon Frahling, Piotr Indyk, and Christian Sohler. Sampling in dynamic data streams and applications. *International Journal of Computational Geometry & Applications*, 18(01n02):3–28, 2008.

9 Eran Gat and Shafi Goldwasser. Probabilistic Search Algorithms with Unique Answers and Their Cryptographic Applications. In *Electronic Colloquium on Computational Complexity (ECCC)*, volume 18, page 136, 2011.

10 Oded Goldreich. Multi-pseudodeterministic algorithms. In *Electronic Colloquium on Computational Complexity (ECCC)*, 2019.

11 Oded Goldreich, Shafi Goldwasser, and Dana Ron. On the possibilities and limitations of pseudodeterministic algorithms. In *Proceedings of the 4th conference on Innovations in Theoretical Computer Science*, pages 127–138. ACM, 2013.

12 Shafi Goldwasser and Ofer Grossman. Perfect Bipartite Matching in Pseudo-Deterministic RNC. In *Electronic Colloquium on Computational Complexity (ECCC)*, volume 22, page 208, 2015.

13 Shafi Goldwasser, Ofer Grossman, and Dhiraj Holden. Pseudo-deterministic Proofs. *arXiv preprint*, 2017. arXiv:1706.04641.

14 Parikshit Gopalan and Jaikumar Radhakrishnan. Finding duplicates in a data stream. In *Proceedings of the twentieth annual ACM-SIAM symposium on Discrete algorithms*, pages 402–411. Society for Industrial and Applied Mathematics, 2009.

15 Ofer Grossman. Finding Primitive Roots Pseudo-Deterministically. In *Electronic Colloquium on Computational Complexity (ECCC)*, volume 22, page 207, 2015.

16 Ofer Grossman and Yang P Liu. Reproducibility and Pseudo-Determinism in Log-Space. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 606–620. SIAM, 2019.

17 Moritz Hardt and David P Woodruff. How robust are linear sketches to adaptive inputs? In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 121–130. ACM, 2013.

18 Dhiraj Holden. A Note on Unconditional Subexponential-time Pseudo-deterministic Algorithms for BPP Search Problems. *arXiv preprint*, 2017. `arXiv:1707.05808`.

19 Piotr Indyk. Stable distributions, pseudorandom generators, embeddings, and data stream computation. *Journal of the ACM (JACM)*, 53(3):307–323, 2006.

20 Piotr Indyk and David Woodruff. Optimal approximations of the frequency moments of data streams. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 202–208. ACM, 2005.

21 Rajesh Jayaram and David P Woodruff. Perfect lp sampling in a data stream. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 544–555. IEEE, 2018.

22 Hossein Jowhari, Mert Sağlam, and Gábor Tardos. Tight bounds for lp samplers, finding duplicates in streams, and related problems. In *Proceedings of the thirtieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 49–58. ACM, 2011.

23 Michael Kapralov, Jelani Nelson, Jakub Pachocki, Zhengyu Wang, David P Woodruff, and Mobin Yahyazadeh. Optimal lower bounds for universal relation, and for samplers and finding duplicates in streams. In *Foundations of Computer Science (FOCS), 2017 IEEE 58th Annual Symposium on*, pages 475–486. Ieee, 2017.

24 Yi Li, Huy L Nguyen, and David P Woodruff. Turnstile streaming algorithms might as well be linear sketches. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pages 174–183. ACM, 2014.

25 Jayadev Misra and David Gries. Finding repeated elements. *Science of computer programming*, 2(2):143–152, 1982.

26 Morteza Monemizadeh and David P Woodruff. 1-pass relative-error L p-sampling with applications. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*, pages 1143–1160. Society for Industrial and Applied Mathematics, 2010.

27 Robert Morris. Counting large numbers of events in small registers. *Communications of the ACM*, 21(10):840–842, 1978.

28 Jelani Nelson, Huy L Nguyen, and David P Woodruff. On deterministic sketching and streaming for sparse recovery and norm estimation. *Linear Algebra and its Applications*, 441:152–167, 2014.

29 Noam Nisan. Pseudorandom generators for space-bounded computation. *Combinatorica*, 12(4):449–461, 1992.

30 Igor C Oliveira and Rahul Santhanam. Pseudodeterministic Constructions in Subexponential Time. *arXiv preprint*, 2016. `arXiv:1612.01817`.

31 Igor C Oliveira and Rahul Santhanam. Pseudo-Derandomizing Learning and Approximation. In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 116. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

32 Salil P Vadhan et al. Pseudorandomness. *Foundations and Trends® in Theoretical Computer Science*, 7(1–3):1–336, 2012.

Limits to Non-Malleability

Marshall Ball
Columbia University, New York City, NY, USA
marshall@cs.columbia.edu

Dana Dachman-Soled
University of Maryland, College Park, MD, USA
danadach@umd.edu

Mukul Kulkarni[1]
University of Massachusetts Amherst, MA, USA
mukul@cs.umass.edu

Tal Malkin
Columbia University, New York City, NY, USA
tal@cs.columbia.edu

Abstract

There have been many successes in constructing explicit non-malleable codes for various classes of tampering functions in recent years, and strong existential results are also known. In this work we ask the following question:

When can we rule out the existence of a non-malleable code for a tampering class $\mathcal{F}$?

First, we start with some classes where positive results are well-known, and show that when these classes are extended in a natural way, non-malleable codes are no longer possible. Specifically, we show that no non-malleable codes exist for any of the following tampering classes:

- Functions that change $d/2$ symbols, where d is the distance of the code;
- Functions where each input symbol affects only a single output symbol;
- Functions where each of the n output bits is a function of $n - \log n$ input bits.

Furthermore, we rule out constructions of non-malleable codes for certain classes $\mathcal{F}$ via reductions to the assumption that a distributional problem is hard for $\mathcal{F}$, that make black-box use of the tampering functions in the proof. In particular, this yields concrete obstacles for the construction of efficient codes for NC, even assuming average-case variants of $P \not\subseteq \mathsf{NC}$.

2012 ACM Subject Classification Security and privacy → Information-theoretic techniques; Security and privacy → Mathematical foundations of cryptography; Security and privacy → Cryptography

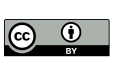
Keywords and phrases non-malleable codes, black-box impossibility, tamper-resilient cryptogtaphy, average-case hardness

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.80

Related Version Full version of this paper is available at `https://eprint.iacr.org/2019/449`.

Funding The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either express or implied, of ODNI, IARPA, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein.
Marshall Ball: M. Ball is supported by an IBM Research PhD Fellowship. This research is based upon work supported in part by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA) via Contract No. 2019-1902070006.
Dana Dachman-Soled: This work is supported in part by NSF grants #CNS-1933033, #CNS-1840893,

[1] Part of this work was done when the author was studying at University of Maryland, USA



#CNS-1453045 (CAREER), by a research partnership award from Cisco and by financial assistance award 70NANB15H328 from the U.S. Department of Commerce, National Institute of Standards and Technology.
Tal Malkin: This research is based upon work supported in part by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA) via Contract No. 2019-1902070006.

1 Introduction

Since the introduction of non-malleable codes (NMC) by Dziembowski, Pietrzak, and Wichs in 2010, there has been a long line of work constructing non-malleable codes for various classes [43]. A plethora of upper bounds, explicit and implicit (to varying degrees), have been shown for a wealth of classes of tampering functions. However, to our knowledge, relatively little is known about when non-malleability is impossible. In this work, we initiate the study of the limits to non-malleability.

Non-malleability for a class $\mathcal{F}$ is defined via the following "tampering" experiment:
Let $f \in \mathcal{F}$ denote a tampering function.

1. Encode message m using a (public) randomized encoding algorithm: $c \leftarrow \mathsf{E}(m)$,
2. Tamper the codeword: $\tilde{c} = f(c)$,
3. Decode the tampered codeword (with public decoder): $\tilde{m} = \mathsf{D}(\tilde{c})$.

Roughly, the encoding scheme, (E, D), is non-malleable for a class $\mathcal{F}$, if for any $f \in \mathcal{F}$ the result of the above experiment, $\tilde{m}$, is either identical to the original message, or completely unrelated. More precisely, the outcome of a $\mathcal{F}$-tampering experiment should be simulatable without knowledge of the message m (using a special flag "same" to capture the case of unchanged message).

[43] showed that, remarkably, this definition is achievable for any $\mathcal{F}$ such that $\text{loglog}|\mathcal{F}| < n - 2 \cdot \log(1/\epsilon)$, where n is the length of the codeword (the input/output of functions in $\mathcal{F}$), and ϵ parameterizes the quality of simulation possible (see Definition 3). However the definition is not achievable in general. It is easy to observe that if $\mathcal{F}$ is the class of all functions, there is a trivial tampering attack: decode, maul, and re-encode. This same observation rules out the possibility of *efficient* codes against efficient tampering, as this attack only requires that decoding and outputting constants conditioned on the result is in the tampering class. By a similar argument, the decoding function of a non-malleable code with respect to the distribution formed by encoding a random one-bit message can be seen as existence of hard decision problem for the tampering class. (This, in turn, informs us of where to hope for unconditional constructions.)

In this work, we give a variety of impossibility results for non-malleable codes, in disparate tampering regimes. We present 3 unconditional impossibility results for various classes, which hold even for *inefficient* NMC. These impossibility results apply to classes that are simple and natural extensions of classes with well-known and efficient NMC constructions. Additionally, we rule out constructions of NMC for a wide range of complexity classes with security reductions that are only given black-box access to the tampering function. This result is more technically complex than the previous ones, and requires the introduction of a new notion of fine-grained black-box reductions appropriate for the non-malleability setting, as we explain below. This result allows us to study the minimal assumptions necessary for achieving NMC for complexity classes contained in P (e.g., NC^1), and to rule out such NMC constructions (with black-box reductions) from minimal average-case hardness assumptions.

To our knowledge, the only previously-known impossibility results beyond the simple observations above, are related to other variants of NMC. These include bounds on locality of locally decodable and updatable NMC, bounds on continuous NMC, and impossibility of "look-ahead" or "block-wise" NMC (which also follows from a simple observation). There are also several bounds related to the *rate* of NMC. We discuss these and other related works in Section 1.4. In contrast, our results hold regardless of rate. In fact, our lower bounds rule out even message spaces of size two or three.

1.1 Strictly Impossible

We identify 3 tampering regimes where non-malleability is strictly impossible.

On tampering functions that change $d/2$ symbols, where d is the distance of the code

It is common to present non-malleable codes as a strict relaxation of error-correcting codes (ECC). It is easy to see that ECC are NMC against tampering that changes up to the allowed fraction of symbols, but since NMC only require correctness of decoding in the absence of errors, they can provide "security" for tampering functions that ECC cannot, in particular functions that can modify most, or even all, symbols of the codeword. This suggests that we could potentially construct NMC for tampering classes that are strictly larger than any class for which ECC could exist. However, no such construction is known: all NMC results that allow to modify more symbols, also require that the computation of the tampering function is restricted in some way. In contrast, ECC do not place any restrictions on the tampering adversary beyond the limit to the number of modified symbols.

In the current work we ask whether this trade-off is in fact necessary. Specifically, can we construct NMC that allow for modifying more symbols of the codeword than ECC *without* placing any other restrictions on the tampering? Note that for ECC it is known that if the distance of the code is d, it is not possible to correct when $d/2$ symbols are modified, but there are constructions that allow for error correction after arbitrary modification of at most $d/2 - 1$ symbols (e.g., Reed-Solomon ECC achieve this bound). Indeed, for the case of potentially inefficient coding schemes, the above is tight: A coding scheme with distance d implies error correction against $d/2 - 1$ errors. Thus, fixing a message space $\mathcal{M}$ and a codeword space $\mathcal{C}$, we consider the optimal ECC for this message and codeword space, which has some distance d (and therefore can correct $d/2 - 1$ errors). We then ask whether one can construct a non-malleable code with message space $\mathcal{M}$ and codeword space $\mathcal{C}$ against the class of functions that may arbitrarily tamper with $d/2$ codeword symbols.

We fully resolve our question. We show that for message space of size 2, non-malleable codes that tolerate arbitrary modification of even up to $d - 1$ symbols are *possible* (via a repetition code, see Section 3). On the other hand, for message space of size greater than 2, it is *impossible* to construct non-malleable codes with distance d for tampering functions that arbitrarily modify $d/2$ codeword symbols. This indicates that for message space larger than 2, in order to obtain improved parameters beyond what is possible with error correcting codes, imposing some additional restrictions on the tampering function is *necessary*.

On tampering functions where each input symbol affects at most one output symbol

In their recent work, Ball et al. [17] presented unconditional NMC for the class of output-local functions, where each output symbol depends on a bounded number of input symbols. As an intermediate step, they also considered the class of functions that are both input and output

local. The class of input-local functions is the class of functions where each input symbol affects a bounded number of output symbol. A natural question is whether non-malleable codes can be constructed for the class of input-local functions, where for codeword length n, each input bit affects $\ll n$ output bits.

In this work, we answer this question negatively in a very strong sense. We show that even achieving NMC for 1-input local functions (where each input bit affects at most one output bit) is impossible. In fact, our proof shows an even stronger result: the impossibility holds even if each input symbol can only affect the same single output symbol. That is, it is impossible to construct NMC for the tampering class that allows to change only one codeword symbol in a way that depends on the input (while the other symbols may be changed into constant values). Stated like this, this result can also be viewed as an extension of our first result above on the maximum number of symbols that can be modified in a non-malleable code.

On tampering functions where each output symbol depends on $n - \log(1/(4\epsilon))$ input symbols

Here we move on to consider achieving NMC for output-local tampering functions. The prior work of [17] constructed efficient NMC for tampering functions with locality n^c, for constant $c < 1$. The size of the class of all output-local tampering functions (with locality sufficiently smaller than n) is also bounded in size and therefore NMC for this class follow from the existential results of [43]. A natural question is how large can the output-locality be?

We prove the impossibility of ϵ-non-malleable codes (see Definition 3) for the class of $(n - \log(1/(4\epsilon)))$-output-local tampering functions. In addition to the above motivation, parity over n bits is average-case hard for this class.[1] Therefore, our impossibility result can be viewed as a "separation" between average-case hardness and non-malleability, as it exhibits a class for which we have average-case hardness bounds, but cannot construct non-malleable codes for. Furthermore, the class $\mathcal{F}'$ constructed in our lower bound proof has size only $4^n \cdot 2^{2^{n-\log(1/(4\epsilon))}}$, which in turn means that $\log\log|\mathcal{F}'| = n - \log(1/(4\epsilon)) = n - \log(1/\epsilon) + 2$. On the other hand, the aforementioned result of Dziembowski et al. [43] shows existence of an ϵ-non-malleable code for any class $\mathcal{F}$ such that $\log\log|\mathcal{F}| \leq n - 2\log(1/\epsilon)$. Thus, our lower bound result is close to matching the existential upper bound.

Remark: deterministic vs. randomized decoding

The standard (and original) definition of NMC requires deterministic decoding and perfect correctness, although some later work took advantage of randomized decoding.[2] We note that our lower bound for the class of input-local functions holds for standard schemes (with deterministic decoding and perfect correctness). Our lower bound (with $\epsilon = \frac{1}{4n}$) for the class of $n - \log(n)$ output-local functions holds even for coding schemes that have *randomized* decoding and perfect correctness. The lower bound for the class of functions that change $d/2$ symbols holds even for coding schemes with *randomized* decoding and *imperfect* correctness.

---

[1] Note that, even arbitrary decision trees of depth $n - 1$ have no advantage in computing the parity of n bits with respect to the uniform distribution. [22]

[2] For the class of output-local functions (where each output depends on at most ℓ inputs) we have explicit constructions with randomized decoding (and $\epsilon = \mathsf{negl}(n)$) for $\ell < n/\log n$ [17], whereasconstructions with deterministic decoding are known for locality up to $n^{1/2-\epsilon}$ for small ϵ. [28, 15].

1.2 Impossibility of Black-Box Security Reductions

In recent work, unconditional constructions of non-malleable codes for progressively larger tampering classes, such as NC^0 [17, 29, 15] and AC^0 [29, 15], have been presented. In fact, the construction of [15] remains secure for circuit depths as large as $\Theta(\log(n)/\log\log(n))$. Moreover, due to the impossibility of efficient NMC for all of P, extending their result to obtain unconditional NMC for circuits with asymptotically larger depth would require separating P from NC^1, a problem that is well out of reach with current complexity-theoretic techniques. However, rather than ruling out such constructions entirely, in this regime we ask what are the minimal assumptions necessary for achieving non-malleable codes for NC^1, as well as other classes $\mathcal{F}$ that are believed to be strictly contained in P.

The above question was partially addressed by Ball et al. [18, 16] in their recent work, where they presented a general framework for construction of NMC for various classes $\mathcal{F}$ in the CRS model and under cryptographic assumptions. Instantiating their framework for NC^1 yields a computational, CRS-model construction of 1-bit NMC for NC^1, assuming there is a distributional problem that is hard for NC^1, but easy for P. Moreover, such distributional problems for NC^1 can be based on worst-case assumptions.[3]

In this work, we ask whether 1-bit non-malleable codes for NC^1 in the standard (no-CRS) model can be constructed from the assumption that there are distributional problems that are hard for NC^1 but easy for P. Recall that this assumption is minimal, since the decoding function of a 1-bit non-malleable code for NC^1 w.r.t. the distribution of random encodings of 1 bit messages yields such a distributional problem.

We provide a negative answer, showing that, under black-box reductions (restricting use of the tampering function in the security proof to be black-box), this is impossible.

Specifically, we define a notion of black-box reductions for the setting of 1-bit non-malleable codes (E, D) against a complexity class $\mathcal{F}$ to a distributional problem (Ψ, L) that is hard for $\mathcal{F}$. This type of reduction is required to use the "adversary" – i.e. the tampering function in our setting – in a black-box manner, but is not restricted in its use of the underlying assumptions. To motivate our new notion, we begin by recalling the notions of reductions in complexity theory and cryptography, and how they are used.

Reductions in Complexity Theory

A reduction is a technique in complexity theory that is used to show that *Problem 1* is as hard as *Problem 2*. For example, the famous Cook-Levin theorem showed that SAT (Boolean satisfiability) is as hard as any problem in NP, by showing a reduction from any problem in NP to SAT. In more detail, a reduction R, is an algorithm that receives as input an algorithm A that solves Problem 1 and uses it to solve Problem 2. Typically, R will only access A in an input/output manner as a subroutine (also known as oracle access and denoted as R^A). When a reduction R uses a solver A in this way, R is known as a black-box reduction. Intuitively, in this case, R does not care *how* A solves Problem 1, it just cares that A exhibits the correct input-output behavior. Therefore, the reduction R should still be able to solve Problem 2, even when A is computationally unbounded. In fact, in the Cook-Levin theorem, only a single oracle query is made by the reduction to the algorithm solving SAT.

When is a reduction R between two problems useful? Note that if R is in a complexity class that can solve Problem 2, then existence of such a reduction R is tautological, since R can simply ignore its oracle and solve Problem 2 on its own (so no relationship is demonstrated

[3] Assuming $\oplus\mathsf{L}/\,\mathrm{poly} \not\subseteq \mathsf{NC}^1$ yields a distributional problem since randomized encodings for $\oplus\mathsf{L}/\,\mathrm{poly}$ are known to exist [13, 21, 41, 14].

about the relative hardness of the problems). This is why in the Cook-Levin theorem the reduction is required to be *polynomial time*. Reductions are also useful since they allow us to draw conclusions about the relationships between different complexity classes. For example, using the Cook-Levin theorem, we conclude that if there exists a polynomial time algorithm for solving SAT then there exists a polynomial time algorithm for all of NP (i.e. P = NP). Note that there is actually a subtlety here: In order for the above to hold, we need that whenever A is polynomial time, R^A is also polynomial time. This holds trivially for the case of polynomial time, since the class P is closed under composition. However, as we will see later, this closure does not necessarily hold in some of the settings we consider. Many variations on reductions beyond the setting of the Cook-Levin Theorem and NP-completeness have been considered in complexity theory. For example, polynomial time reductions have no utility when P is the object of study. Instead, the theory of P-completeness uses NC-computable reductions to argue about whether or not problems in P can be parallelized. Another such example is the theory of fine-grained complexity, which seeks to understand the *exact* (or, more exact) complexity of problems in P. Fine-grained reductions allow one to argue that a problem cannot be solved in, for example, $O(n^{2-\epsilon})$ time (for any $\epsilon > 0$), by reduction from another problem believed to require $\Omega(n^{2-o(1)})$ time. For this reasoning to hold, such a fine-grained reduction must run in sub-quadratic time, and moreover it can make at most $n^{o(1)}$ queries to an oracle defined on instances of length n if it is to remain useful. As we will see, this tension between the length of the inputs queried to the oracle and the number of such queries will also be relevant in our setting.

Reductions in Cryptography

Reductions in cryptography are exactly like reductions in complexity theory. For example, the seminal result of [54] proves by reduction that *breaking a pseudorandom function* (Problem 1) is as hard as *breaking a pseudorandom generator* (Problem 2). In order to prove this, they present a reduction R such that that given an algorithm A that breaks the constructed pseudorandom function, R^A breaks the underlying pseudorandom generator. Note that since R only has oracle access to A, again R does not care how A works, as long as it exhibits input-output behavior that qualifies it as a valid distinguisher between a pseudorandom and random function. Thus, R is black-box. As before, R is only useful if it is polynomial time, since otherwise R can break the pseudorandom generator on its own. Furthermore, we again want to use the existence of the reduction to draw conclusions about the security relationship between the pseudorandom function and the pseudorandom generator. Here we want to show that if there exists a polynomial-time algorithm that breaks the pseudorandom function, then there exists a polynomial-time algorithm that breaks the pseudorandom generator. Therefore, we want it to be the case that whenever A is polynomial time, R^A is also polynomial time. This trivially holds, as before, since P is closed under composition. However, in the following we will consider cases where this type of closure does not necessarily hold. For example, when A and R are in NC^1, R^A may no longer be in NC^1 (in fact R^A could have depth up to $\log^2(n)$). We therefore need to include a notion of closure under composition as one of the requirements of a black-box reduction in our setting.

A fine-grained setting: Security reductions for non-malleable codes

What would a security reduction in the setting of non-malleable codes look like? In this case, we want to show that *breaking the non-malleable code* (Problem 1) is as hard as *breaking distributional problem* (Ψ, L) (Problem 2). Here, an algorithm that breaks the non-malleable

code simply consists of a *tampering function* f. A reduction R is provided black-box access to the tampering function f and must use it to break the distributional problem (Ψ, L). First, note that since we assume R is black-box, R is only allowed to use f as a subroutine (gives it inputs and obtains its output), regardless of *how* f performs its computation. Thus, as in all the cases discussed above, we require that R^f break the distributional problem (Ψ, L), even in the case that f is not contained in $\mathcal{F}$. Note that the distributional problem (Ψ, L) is *easy* for polynomial-time. Therefore, for the reduction to be non-trivial, R must be in a complexity class that does not contain P. Indeed, since we only assume that (Ψ, L) is hard for $\mathcal{F}$, R must be contained in $\mathcal{F}$ in order for us to draw any conclusions (otherwise, we cannot rule out the possibility that R simply ignores its oracle and solves (Ψ, L) on its own). Furthermore, as discussed above, the point of the reduction is to be able to conclude that if there is a tampering function f in $\mathcal{F}$ that breaks the non-malleable code, then there exists an algorithm in $\mathcal{F}$ that breaks the distributional problem (Ψ, L). Therefore, it is not enough that $R \in \mathcal{F}$, and we actually need that whenever $f \in \mathcal{F}$, $R^f \in \mathcal{F}$. We will then use the fact that R^f breaks (Ψ, L) and is used to obtain a contradiction to the hardness of (Ψ, L) for $\mathcal{F}$.

Overall, at a high level (skipping some technical details), we require two properties of a black-box reduction R from (E, D) to (Ψ, L):

- If the tampering function f succeeds in breaking the non-malleable code, the reduction, R^f, should succeed, regardless of whether $f \in \mathcal{F}$. This represents the fact that R uses f in a black-box manner.
- For any $f \in \mathcal{F}$, R^f must also be in $\mathcal{F}$, and in particular, R itself must be in $\mathcal{F}$. This represents the fact that the black-box reduction R should allow one to obtain a contradiction to the assumption that (Ψ, L) is hard for $\mathcal{F}$, in the case that (E, D) is malleable by $\mathcal{F}$.

Note that for arbitrary classes $\mathcal{F}$ (unlike the usual polynomial-time adversaries typically used in cryptography), the fact that $R \in \mathcal{F}$ and $f \in \mathcal{F}$ does not necessarily imply that $R^f \in \mathcal{F}$. This introduces some additional complexity in our definitions and also requires us to restrict our end results to classes $\mathcal{F}$ that behave appropriately under composition.

We present general impossibility results for constructing 1-bit non-malleable codes for a class $\mathcal{F}$ from a distributional problem that is hard for $\mathcal{F}$ but easy for P. We present three types of results: results ruling out *security parameter preserving* reductions for tampering class $\mathcal{F}$ that behave nicely under composition; results ruling out *"approximate" security parameter preserving* reductions for tampering class $\mathcal{F}$ with slightly stronger compositional properties; and results ruling out *non-security parameter preserving* reductions for tampering class $\mathcal{F}$ that are fully closed under composition. See Definitions 21, 22 and Lemmas 31, 35, 37 for the formal statements.

Briefly, security parameter preserving reductions have the property that the reduction only queries the adversary (in our case the tampering function) on the same security parameter that it receives as input. The security parameter preserving reductions have been used in constructions of leakage resilient circuit compilers [11]. The notion of "approximate" security parameter preserving reductions is new to this work. Such reductions are parameterized by polynomial functions $\ell(\cdot), u(\cdot)$ and on input security parameter n, the reduction may query the adversary on any security parameter in the range $\ell(n)$ to $u(n)$. This notion is somewhat less restrictive than a security parameter preserving reduction. Finally, in a non-security parameter preserving reduction, the reduction receives security parameter n as input and may query the adversary on arbitrary security parameter n'. Note that $n'(n)$ must be in $O(n^c)$ for some constant c, since the reduction must be polynomial time. This notion allows us to rule out the most general type of black-box reduction discussed above.

We can instantiate the tampering class $\mathcal{F}$ from our generic lemma statements with various classes of interest. Our results on security parameter preserving and approximate security parameter preserving reductions apply to the class NC^1 as a special case. Our result ruling out non-security parameter preserving reductions applies to the class (non-uniform) NC as a special case. See Corollaries 32, 36, 38 for the formal statements. As the proofs are already quite involved, we make the simplifying assumption of deterministic decoding and perfect correctness. However, this is not inherent to the proof and we expect the results to extend to coding schemes with imperfect correctness and randomized decoding.

Do reductions for NMC take the above form?

So far, in the non-malleable codes setting, results have either been *unconditional* (e.g. [43, 4]) or have been based on polynomial-hardness assumptions (e.g. [64, 2]). The results that are based on polynomial-hardness assumptions have all used black-box security reductions, in the standard polynomial-time sense [69]. Our notion is new since it captures a fine-grained setting where the underlying distributional problem is, in fact, *easy* for polynomial-time algorithms. As discussed above, this is the minimal computational hardness assumption necessary to construct non-malleable codes for classes $\mathcal{F}$ for which we cannot prove unconditionally that $\mathsf{P} \not\subseteq \mathcal{F}$. While this type of reduction implicitly arises in the work of [18], our work is the first to formally define and explore this notion of fine-grained black-box reductions in a cryptographic setting.

1.3 Technical Overview

In order to prove impossibility of constructing non-malleable codes in different scenarios, we need to show that any such code is malleable. Recall that for single-bit messages, non-malleability is equivalent to showing that when applying the tampering function to a randomly generated encoding of a random bit, the decoded value flips with probability at most $1/2 + \mathsf{negl}(n)$. Thus, proving that something is malleable, corresponds to showing that the decoded value flips with probability at least $1/2 + 1/\operatorname{poly}(n)$. We will use this fact in the following exposition.

1.3.1 On tampering functions that change $d/2$ symbols, where d is the distance of the code

We observe that for any coding scheme, there must be some message, x^*, such that *every* encoding of that message is at most distance $d/2$ from something that is likely to decode to something other than x^*. Our tampering function will only modify encodings of x^*, and it will do so by moving each encoding to one of these nearby points that decodes differently. We claim that if there are at least 3 messages in the message space, then the output of decoding with the tampering function described above depends on the input message (and thus cannot be simulated). Indeed, when starting with message x^* (which is encoded, tampered, and decoded) there must be some other message $y^* \neq x^*$ that is not output a majority of the times by this process. On the other hand, when starting with y^*, since the tampering does not change anything in this case, correctness of decoding means that y^* should be output a majority of times.

This argument falls apart if there are only two possible messages, and in this case a repetition code with a decoding that outputs a fixed value on invalid codewords is, in fact, non-malleable with respect to tampering functions that can change up to $n - 1$ symbols.

1.3.2 On tampering functions where each input symbol affects at most one output symbol

Consider any two codewords c_x and c_y corresponding to distinct messages, x and y. Now consider any sequence of n codewords between c_x and c_y, made by altering one symbol at a time. There must be two adjacent codewords, c_i, c_{i+1}, (differing on a single bit) in this sequence that decode differently. Therefore, to tamper, simply output c_i if the input is an encoding of 0, and c_{i+1} otherwise. Because c_i and c_{i+1} just differ on a single bit, the tampering function has input locality 1.

1.3.3 On tampering functions where each output symbol depends on $n - \log(1/(4\epsilon))$ input symbols

We begin by considering a simpler argument, that only rules out tampering functions of output locality $n-1$ (each output bit can depend on at most $n-1$ input bits), where n is the bit length of the codeword. To illustrate the idea in the locality $n-1$ case, we also assume that the decoding algorithm is deterministic. We consider two cases and show that each case leads to a different tampering attack:

- **Case 1:** Given the first $n-1$ bits of the codeword, the codeword decodes to the same bit b, regardless of whether the final bit is 0 or 1. In this case, the tampering function contains a hardwired valid codeword encoding 0 and a valid codeword encoding 1. The tampering function derives the bit b, given only the first $n-1$ bits (since the decoded bit b is independent of the final bit) and replaces the codeword with the hardwired encoding of $1-b$.
- **Case 2:** Given the first $n-1$ bits of the codeword, the codeword decodes to 0 if the final bit is set to b and decodes to 1 if the final bit is set to $1-b$. In this case, the tampering function just flips the final bit, causing the decoding to flip.

The key observation is that we can extend the attacks for Cases 1 and 2 above to tampering functions of output locality $n - \log(1/(4\epsilon))$. We will sketch the special case corresponding to extending to $\epsilon = \frac{1}{4n}$ (and locality $(n - \log(n))$), to rule out non-malleable codes with negligible error. Case 1 now corresponds to the case that, for a randomly generated codeword, when the first $n - \log(n)$ bits of the codeword are fixed and the remaining $\log(n)$ bits are set at random, the decoded value *remains the same* with probability at least $1/2 + 1/(4n)$ In this case, the tampering function gets the first $n - \log(n)$ bits, randomly sets the final $\log(n)$ bits and decodes to obtain a bit b. Then, the tampering function succeeds in flipping the encoding with probability $1/2 + 1/(4n)$ by replacing the codeword with the hardwired encoding of $1 - b$.

Case 2 now corresponds to the case that for a randomly generated codeword, when the first $n - \log(n)$ bits of the codeword are fixed and the remaining $\log(n)$ bits are set at random, the decoded value *flips* with probability at least $1/2 - 1/(4n)$. Note that the decoded value *never flips* when the randomly chosen $\log(n)$ bits happen to be the same as the original value, which occurs with probability $1/n$. Thus, if the final $\log(n)$ bits are chosen at random, conditioned on being different from the original value, then the decoded value must flip with probability at least $\frac{1/2 - 1/(4n)}{1 - 1/n} \geq 1/2 + 1/(4n)$. In this case, the tampering function ignores the first part of the codeword and simply sets the final $\log(n)$ bits at random, *conditioned on the value being different from the original value*. Then, the tampering function succeeds in flipping the value of the encoding with probability at least $1/2 + 1/(4n)$.

Our final argument for $n - \log(1/(4\epsilon))$-locality holds even for *randomized* decoding.

1.3.4 Impossibility of Black-Box Security Reductions

We begin by describing our proof showing the impossibility of a black-box, security-parameter preserving reduction, from NMC against the tampering class NC^1, to a distributional problem that is hard for NC^1. The proof for approximately security parameter preserving reductions is essentially the same, and so we subsequently describe the extension to impossibility of a black-box, *non*-security-parameter preserving reduction, from non-malleable codes against the tampering class NC, to a distributional problem that is hard for NC.

Our proof proceeds via the meta-reduction technique. Specifically, consider a black-box reduction R, reducing the security of a single-bit non-malleable code against NC^1 to a distributional problem that is hard for NC^1. The form that this reduction takes, is that it submits codewords c to the tampering function f and gets back (tampered) codewords y as responses. The main idea is to begin with a tampering function f, which is not in NC^1. This tampering function receives a codeword c, decodes it to obtain the bit b and then submits a randomly generated encoding of the bit $1-b$. In the proof, we assume the existence of a reduction R such that R^f breaks the underlying distributional problem (this follows from the definition of a black-box reduction). We then switch from f to a tampering function f' that *is* in NC^1, which behaves as follows: Upon receiving a codeword c, f simply responds with a (hardcoded) random codeword c' that encodes a random bit, independent of the bit that is obtained when the decoding algorithm is applied to c. This switch is desirable, since then $R^{f'}$ will be in NC^1 (note that we are guaranteed that $R^{f'}$ is in NC^1, since one of the properties of R is that whenever the tampering function f' is in NC^1, then $R^{f'}$ must also be in NC^1). It remains, however, to show that $R^{f'}$ succeeds in breaking the underlying distributional problem, which then implies that the underlying distributional problem is not hard for NC^1. In order to ensure this, we use a hybrid argument, where responses to queries from R are switched one by one, from responses according to f to responses according to f'. In each step, we must show that the reduction remains successful in breaking the underlying distributional problem. Importantly, in the i-th hybrid, the first $i-1$ responses are answered according to f, the i-th response and on are answered according to f'. Since R is in NC^1, we argue that if R can distinguish the $(i-1)$-st and i-th hybrids, then we obtain a *tampering attack in* NC^1 *on the non-malleable code*. To do this, we construct a tampering function that hardwires the input to R, the transcript (queries and responses) and entire state of R for the first $i-1$ queries made from R to f, the i-th query along with the value b that it decodes to, and the responses to the queries $i+1$ and on. Then, given an input codeword c', the tampering function inserts this value as the response to the i-th query, runs the reduction R from this point on (given the state of R at the point of the response to the i-th query) and responds with the random hardwired queries upon any future queries from R. If R distinguishes between Hybrids $i-1$ and i, then the above yields a distinguisher between randomly generated encodings of the bit b, versus randomly generated encodings of a random bit. It is not hard to see that such a distinguisher immediately yields a tampering attack, since it can be used to predict the underlying encoded value and the tampering attack can then replace the codeword with an encoding of a bit which is the opposite of the predicted bit.

In the above, note that it is crucial that the reduction is security parameter preserving. Indeed, if R queries codewords c' that are very short (say length $\log^2(n)$) then we can no longer use R to obtain a valid tampering function against the non-malleable code. This is because R has size $\mathrm{poly}(n)$ and depth $\log(n)$, which is not in NC^1 relative to input length $\log^2(n)$. To deal with this problem, we take advantage of the fact that R must be successful even for tampering functions f that work only for very sparse input lengths $\{1, 2, 2^2, 2^{2^2}, \ldots\}$.

In this way, we can essentially guarantee that the reduction queries at most a single input length ℓ which is greater than $\log(n)$ and at most $\text{poly}(n)$. We now consider two cases: Either for this input length ℓ it is the case that NC circuits can distinguish encodings of 0 and 1 with probability at least $3/4$, or for this input length ℓ it is the case that NC circuits can distinguish encodings of 0 and 1 with probability at most $1 - 1/\text{poly}(n)$. If we are in the first case, then we can actually honestly run the attack using a NC circuit (in this case we just use the distinguisher to guess the value of the encoding and succeed with probability $3/4$). If we are in the second case, then we can use Impagliazzo's Hard Core Set [55] to find a set of encodings such that a NC circuit can distinguish random encodings of 0 and 1 from this set with probability at most $1/2 + 1/\text{poly}(n)$. In this case, we modify the tampering function to hardcode random encodings *from the hard core set* and return these in response to the queries from R. Note that to obtain contradiction to the security of the constructed non-malleable code, we now require that when the reduction R is composed with any tampering circuit in NC, then the composed circuit is still in NC. This property holds for NC, but not NC^1, which is why our result on ruling out non-security preserving reductions holds only for NC.

1.4 Related Work

Non-Malleable Codes

Non-malleable codes (NMC) were introduced in the seminal work of Dziembowski, Pietrzak and Wichs [43]. In the same paper they pointed out the simple impossibility result for NMC for all polynomial tampering functions. Since then NMC have been studied in the information-theoretic as well as computational settings. Liu and Lysyanskaya [64] studied the split-state classes of tampering functions and constructed computationally secure NMC for split-state tampering. A long line of work followed in both the computational [2] as well as information theoretic setting with a series of advances – reduced number of states, improved rate, or adding desirable features to the scheme [42, 4, 31, 3, 8, 2, 26, 58, 62, 63, 7]. Recently efficient NMC have been constructed for tampering function classes other than split-state tampering [17, 9, 29, 45, 18, 15, 16, 19, 32] in both the computational and information-theoretic setting. Additionally, [43, 33, 48] present various inefficient, existential or randomized constructions for more general classes of tampering functions. These results are sometimes presented as efficient constructions in a random-oracle or CRS model. Other works on non-malleable codes include [46, 34, 24, 6, 57, 40, 47, 3, 25, 23, 60, 38, 5, 39, 59, 65, 61, 44, 27, 30, 68, 35].

Bounds on Non-Malleable Codes

Surprisingly, understanding the limitations and bounds on NMC has received relatively less attention. While there have been a few previous works exploring the lower and upper bounds on NMC and its variants [43, 33, 23, 38, 37], most of the effort has been focused on understanding and/or improving the bounds on the rates of NMC [2, 9, 8, 58, 63, 35]

Perhaps the closest to this work are the results of [33, 38, 37]. Cheragachi and Guruswami [33] studied the "capacity" of non-malleable codes in order to understand the optimal bounds on the efficiency of non-malleable codes. They showed that information theoretically secure efficient NMC exist for tampering families $\mathcal{F}$ of size $|\mathcal{F}|$ if $\text{loglog}|\mathcal{F}| \leq \alpha n$ for $0 \leq \alpha < 1$, moreover these NMC have optimal rate of $1 - \alpha$ with error $\varepsilon \in O(1/\mathsf{poly}(n))$. Dachman-Soled, Kulkarni, and Shahverdi [38] studied the bounds on the locality of locally decodable and updatable NMC. They showed that for any locally decodable and updatable NMC which allows rewind attacks, the locality parameter of the scheme must be $\omega(1)$, and

gave an improved version of [40] construction to match the lower bound in computational setting. Recently, Dachman-Soled and Kulkarni [37] studied the bounds on continuous non-malleable codes (CNMC), and showed that 2-split-state CNMC cannot be constructed from any falsifiable assumption without CRS. They also gave a construction of 2-split-state CNMC from injective one-way functions in CRS model. Faust et al. [46] showed the impossibility of constructing information-theoretically secure 2-split-state CNMC.

Black-Box Separations

Impagliazzo and Rudich [56] showed the impossibility of black-box reductions from key agreement to one-way function. Their oracle separation technique subsequently led to sequence of works, ruling out black-box reductions between different primitives. Notable examples are [71] separating collision resistant hash functions from one way functions, and [53] ruling out oblivious transfer from public key encryption. The meta-reduction technique (cf. [36, 66, 51, 49, 67, 52, 1, 70, 20, 50]) has been used for ruling out larger classes of reductions – where the construction is arbitrary (non-black-box), but the reduction uses the adversary in a black-box manner. The meta-reduction technique is often used to provide evidence that construction of a cryptographic primitive is impossible under "standard assumptions" (e.g. falsifiable or non-interactive assumptions).

2 Preliminaries

2.1 Notation

For any positive integer n, $[n] := \{1, \ldots, n\}$. For a vector $x \in \chi$ of length n, we denote its *hamming weight* by $\| x \|_0 := |\{x_i : x_i \neq 0 \text{ for } i \in [n]\}|$, where $|S|$ is the cardinality of set S, and x_i denotes the i-th element of x. For $x, y \in \{0,1\}^n$ define their distance to be $d(x,y) := \| x - y \|_0$. (I.e. x and y are ε-far if $d(x,y) \geq \varepsilon$.) We denote the *statistical distance* between two random variables, X and Y, over a domain S to be $\Delta(X,Y) := 1/2 \sum_{s \in S} |\Pr[X = s] - \Pr[Y = s]|$, where $|\cdot|$ denotes the absolute value. We say X and Y are ε-close, denoted by $X \approx_\varepsilon Y$, if $\Delta(X,Y) \leq \varepsilon$.

2.2 Non-Malleable Codes

▶ **Definition 1** (Coding Scheme [43]). *A* Coding scheme, (E, D), *consists of a (possibly randomized) encoding function* $\mathsf{E} : \{0,1\}^k \to \{0,1\}^n$ *and a deterministic decoding function* $\mathsf{D} : \{0,1\}^n \to \{0,1\}^k \cup \{\perp\}$ *such that* $\forall m \in \{0,1\}^k, \Pr[\mathsf{D}(\mathsf{E}(m)) = m] = 1$ *(over randomness of* E*).*

We define the distance of a *randomized* coding scheme by considering the minimum distance of all codes formed as follows: for each message $x \in \{0,1\}^k$ choosing an arbitrary codeword corresponding to that message, $c_x \in \{\mathsf{E}(x;r)\}_{r \in \{0,1\}^*}$. We take the distance of the randomized encoding scheme to be the maximum of all such minimum distances (i.e. the distance of the best sub-code).

▶ **Definition 2** (Distance of a Coding Scheme). *The* distance *of a (randomized) coding scheme,* (E, D)*, is*

$$\max_{\substack{S \subset \{c = \mathsf{E}(x;r) : x \in \{0,1\}^k, r \in \{0,1\}^*\}: \\ \forall x \in \{0,1\}^k, \exists c_x \in S : \Pr[\mathsf{D}(c_x) = x] > 1/2}} \min_{\substack{c_x, c_y \in S: \\ x \neq y \in \{0,1\}^k}} \|c_x - c_y\|_0$$

Note that this definition can be extended to arbitrary alphabets. Moreover, it is clear that the minimum distance any coding scheme with K messages and codeword space Σ^n is upper bounded by the maximum of the traditional notion of minimum distance in (non-randomized) codes with the same parameters: the minimum distance between codewords from a code (over Σ^n with K distinct code words).

▶ **Definition 3** (ε-Non-malleability [43]). *Let $\mathcal{F}$ be some family of functions. For each function $f \in \mathcal{F}$, and $m \in \{0,1\}^k$, define the tampering experiment:*

$$\mathbf{Tamper}_m^f \stackrel{\text{def}}{=} \left\{ \begin{array}{c} c \leftarrow \mathsf{E}(m), \tilde{c} := f(c), \tilde{m} := \mathsf{D}(\tilde{c}). \\ \textit{Output}: \ \tilde{m}. \end{array} \right\},$$

where the randomness of the experiment comes from E. *We say a coding scheme (E, D) is ε-non-malleable with respect to $\mathcal{F}$ if for each $f \in \mathcal{F}$, there exists a distribution D^f over $\{0,1\}^k \cup \{\mathsf{same}^*, \perp\}$ such that for every message $m \in \{0,1\}^k$, we have*

$$\mathbf{Tamper}_m^f \approx_\varepsilon \left\{ \begin{array}{c} \tilde{m} \leftarrow D^f. \\ \textit{Output}: m \textit{ if } \tilde{m} = \mathsf{same}^*; \\ \textit{otherwise } \tilde{m}. \end{array} \right\}$$

Here the indistinguishability can be either statistical or computational.

▶ **Lemma 4** (Lemma 2 [42]). *Let (E, D) be a coding scheme with $\mathsf{E} : \{0,1\} \to \mathcal{X}$ and $\mathsf{D} : \mathcal{X} \to \{0,1\}$. Let $\mathcal{F}$ be a set of functions $f : \mathcal{X} \to \mathcal{X}$. Then (E, D) is ε-non-malleable with respect to $\mathcal{F}$ if and only if for every $f \in \mathcal{F}$,*

$$\Pr_{b \leftarrow \{0,1\}}[\mathsf{D}(f(\mathsf{E}(b))) = 1 - b] \leq \frac{1}{2} + \varepsilon,$$

where the probability is over the uniform choice of b and the randomness of E.

▶ **Definition 5** (ε-Malleable Code). *Let (E, D) be a coding scheme with $\mathsf{E} : \{0,1\} \to \mathcal{X}$ and $\mathsf{D} : \mathcal{X} \to \{0,1\}$. Let $\mathcal{F}$ be a set of functions $f : \mathcal{X} \to \mathcal{X}$. Then (E, D) is ε-malleable with respect to $\mathcal{F}$, if $\exists f \in \mathcal{F}$ such that,*

$$\Pr_{b \leftarrow \{0,1\}}[\mathsf{D}(f(\mathsf{E}(b))) = 1 - b] \geq \frac{1}{2} + \varepsilon,$$

where the probability is over the uniform choice of b and the randomness of E.

2.3 Input/Output Local Functions

We next define input and output local functions. In input local functions, each input bit can affect a bounded number of output bits. In output local functions, each output bit is affected by a bounded number of input bits. Loosely speaking, an input bit x_i affects the output bit y_j if for any boolean circuit computing f, there is a path in the underlying DAG from x_i to y_j. The formal definitions are below, and our notation follows that of [12].

▶ **Definition 6** ([17]). *We say that a bit x_i* affects *the boolean function f, if $\exists \{x_1, x_2, \cdots x_{i-1}, x_{i+1}, \cdots x_n\} \in \{0,1\}^{n-1}$ such that, $f(x_1, x_2, \cdots x_{i-1}, 0, x_{i+1}, \cdots x_n) \neq f(x_1, x_2, \cdots x_{i-1}, 1, x_{i+1}, \cdots x_n)$.*

Given a function $f = (f_1, \ldots, f_n)$ (where each f_j is a boolean function), we say that input bit x_i affects *output bit y_j, or that output bit y_j* depends *on input bit x_i, if x_i affects f_j.*

▶ **Definition 7** (Input Locality [17]). *A function $f : \{0,1\}^n \to \{0,1\}^n$ is said to have* input locality ℓ *if every input bit f_i is affects at most ℓ output bits.*

▶ **Definition 8** (Output Locality [17]). *A function $f : \{0,1\}^n \to \{0,1\}^n$ is said to have* output locality m *if every output bit f_i is dependent on at most m input bits.*

▶ **Definition 9** (Input Local Functions [12]). *A function $f : \{0,1\}^n \to \{0,1\}^n$ is said to be* ℓ-input local, $f \in \text{Local}_\ell$, *if it has input locality ℓ.*

▶ **Definition 10** (Output Local Functions [12]). *A function $f : \{0,1\}^n \to \{0,1\}^n$ is said to be* m-output local, $f \in \text{Local}^m$, *if it has output locality m.*

Recall that NC^1 is the class of functions where each output bit can be computed by a efficiently and uniformly generated $\text{poly}(n)$ size boolean circuit with $O(\log n)$ depth and constant fan-in, where n is the input size. NC is the class of functions where each output is computed by a uniformly and efficiently generated $\text{poly}\log(n)$ depth $\text{poly}(n)$ size circuit. $\mathsf{nu}-\mathsf{NC}$ is the class of functions computed by a $\text{poly}\log(n)$ depth $\text{poly}(n)$ size circuit.

▶ **Definition 11** (Pseudorandom Generator [41]). *Let $n, n' \in \mathbb{N}$ such that $n' > n$, and let $\mathsf{PRG} = \{\mathsf{prg}_n : \{0,1\}^n \to \{0,1\}^{n'}\}$ be a family of deterministic functions which can be computed in computational class $\mathcal{C}_1$. We say PRG is a $\mathcal{C}_1$-*pseudorandom generator for $\mathcal{C}_2$ *if for any $D := \{D_n : \{0,1\}^{n'} \to \{0,1\}\} \in \mathcal{C}_2$:*

$$|\Pr[D_n(\mathsf{prg}_n(x)) = 1] - \Pr[D_n(r) = 1]| \leq \mathsf{negl}(n),$$

where $x \leftarrow \{0,1\}^n$ and $r \leftarrow \{0,1\}^{n'}$ are sampled uniform randomly.

For class $\mathcal{C}$, if $\mathcal{C}_1 = \mathcal{C}_2 = \mathcal{C}$ then we simply call it $\mathcal{C}$-pseudorandom generator.

2.4 Distributional Problems

▶ **Definition 12** (Distributional Problem). *A* distributional problem *is a decision problem along with ensembles $(\Psi = \{\Psi_n\}_{n=1}^\infty, L = \{L_n\}_{n=1}^\infty)$ for $n \in \mathbb{N}$, where Ψ_n is probability distribution over $\{0,1\}^n$. The decision problem is to decide whether $s \in L_n$ where, $s \leftarrow \Psi_n$. For a string $s \in \{0,1\}^n$, we define $L(s)$ to output 1, if and only if $s \in L_n$.*

Note that length of s need not be n.

We say distributional problem $(\Psi = \{\Psi_n\}_{n=1}^\infty, L = \{L_n\}_{n=1}^\infty)$ is in P if $L \in \mathsf{P}$. We say it is efficiently samplable if there is a randomized poly-time algorithm that on input 1^n samples Ψ_n.

▶ **Definition 13** ($\varepsilon(n)$-Hard Distributional Problem). *Let $(\Psi = \{\Psi_n\}_{n=1}^\infty, L = \{L_n\}_{n=1}^\infty)$ be a distributional problem, we say that (Ψ, L) is $\varepsilon(n)$-hard for family of boolean circuits $\mathcal{C} = \{C_n\}_{n=1}^\infty$, if and only if for every circuit $C_n \in \mathcal{C}$,*

$$\Pr_{x \leftarrow \Psi_n}[C_n(x) = L_n(x)] \leq \frac{1}{2} + \varepsilon(n)$$

▶ **Definition 14** ($\varepsilon(n)$-Easy Distributional Problem). *Let $(\Psi = \{\Psi_n\}_{n=1}^\infty, L = \{L_n\}_{n=1}^\infty)$ be a distributional problem, we say that (Ψ, L) is $\varepsilon(n)$-easy for family of boolean circuits $\mathcal{C} = \{C_n\}_{n=1}^\infty$, if there exists some circuit $C_n \in \mathcal{C}$,*

$$\Pr_{x \leftarrow \Psi_n}[C_n(x) = L_n(x)] \geq \frac{1}{2} + \varepsilon(n)$$

2.5 Hardness of Boolean Functions and Composition

▶ **Definition 15** (δ-hardness of boolean function). *Let $f : \{0,1\}^n \to \{0,1\}$ be a boolean function, and U_n be uniform distribution over $\{0,1\}^n$. Also let $0 < \delta < \frac{1}{2}$, and $n \leq s \leq \frac{2^n}{n}$. We say f is* δ-hard for size s *if for any boolean circuits C of size at most s*

$$\Pr_{x \leftarrow U_n}[C(x) = f(x)] \leq 1 - \delta.$$

▶ **Definition 16** (ε-hard-core function). *Let $f : \{0,1\}^n \to \{0,1\}$ be a boolean function, and D_S be a distribution over $\{0,1\}^n$ induced by the characteristic function of set $S \subset \{0,1\}^n$*[4]*. We call f* ε-hard-core *on S for size s (where $n \leq s \leq \frac{2^n}{n}$), if for any boolean circuits C of size at most s*

$$\Pr_{x \leftarrow D_S}[C(x) = f(x)] < \frac{1}{2} \cdot (1 + \varepsilon).$$

We also present the following theorem from [55].

▶ **Theorem 17** (Theorem 1 [55]). *Let $f : \{0,1\}^n \to \{0,1\}$ be boolean function that δ-hard for size s. Also, let $\varepsilon > 0$. Then $\exists$ set $S \subseteq \{0,1\}^n$ and constant c, such that $|S| \geq \delta \cdot 2^n$ and f is ε-hard-core on S for circuits of size $s' \leq c \cdot \varepsilon^2 \cdot \delta^2 \cdot s$.*

▶ **Definition 18** (Hard Core Set (HCS) Amenable). *We say that $\mathcal{F} = \{\mathcal{F}_n\}_{n=1}^{\infty}$ is HCS-Amenable if for any polynomial $p(\cdot)$, it holds that if $C_1, \ldots, C_{p(n)} \in \mathcal{F}_n$ then* $\mathsf{MAJ}(C_1, \ldots, C_{p(n)}) \in \mathcal{F}_n$.

We now present definitions of functionalities Unroll and Replace which will then allow us to define the appropriate notions of composition and closure for function classes.

▶ **Definition 19** (Unroll functionality.). *Let $F := \{f_n\}_{n=1}^{\infty} \in \mathcal{F}$, where $f_n : \{0,1\}^n \to \{0,1\}$ and $G = \{g_m\}_{m=1}^{\infty} \in \mathcal{G}$, where $g_m : \{0,1\}^m \to \{0,1\}^m$, be function families. Also let t, p be polynomials. Let $m \in \mathrm{poly}(n)$. Let F^G denote families functions $f_n : \{0,1\}^n \to \{0,1\} \in F$ which contains at most $t(n)$ oracle gates computing $g_m : \{0,1\}^m \to \{0,1\}^m \in G$ and get string of length $p(n)$ as non-uniform advice. On an n-bit input, consider the DAG whose left side consists of the circuit of f_n and whose right side consists of circuits $g_{n_1}, \ldots, g_{n_{t(n)}}$. The values of wires going from the left to the right correspond to (a topological ordering of) the oracle queries $x_1, \ldots, x_{t(n)}$ of lengths $n_1, \ldots, n_{t(n)}$, made in each of the $t(n)$ queries. For $i \in [t(n)]$, circuit g_{n_i} takes as input x_i and returns y_i. The values of wires going from the right to the left correspond to the responses $y_1, \ldots, y_{t(n)}$. We say that this DAG, denoted* $\mathsf{Unroll}(F^G)$, *is an unrolling of $F^G(x)$.*

▶ **Definition 20** (Replace Functionality.). *Consider replacing each g_{n_i}, $i \in [t(n)]$, in* $\mathsf{Unroll}(F^G)$ *with a circuit g'_{n_i} that takes input $(x_1, \ldots, x_i)$ and produces output y_i. This is denoted by* $\mathsf{Replace}(\mathsf{Unroll}(F^G), g'_{n_1}, \ldots, g'_{n_{t(n)}})$.

▶ **Definition 21** ($(\mathcal{G}, t, \ell, u)$-closure of $\mathcal{F}$). *Let $F := \{f_n\}_{n=1}^{\infty} \in \mathcal{F}$, where $f_n : \{0,1\}^n \to \{0,1\}$ and $G = \{g_m\}_{m=1}^{\infty} \in \mathcal{G}$, where $g_m : \{0,1\}^m \to \{0,1\}^m$, be function families. Also let t, ℓ, u be polynomials, and $\ell(n) \leq m \leq u(n)$. Let ${f_n}^{g_m}$ denote function $f_n : \{0,1\}^n \to \{0,1\}$ which has access to the output of $g_m : \{0,1\}^m \to \{0,1\}^m$ on at most $t(n)$ inputs of its choice.*

We say that $\mathcal{F}$ is $(\mathcal{G}, t, \ell, u)$-closed under compositions if for every $F \in \mathcal{F}$ such that for all $G \in \mathcal{G}$, $\mathsf{Unroll}(F^G) \in \mathcal{F}$, *we have that for all $G' \in \mathcal{G}$ and all $g'_{n_1}, \ldots, g'_{n_{t(n)}} \in G'$,* $\mathsf{Replace}(\mathsf{Unroll}(F^G), g'_{n_1}, \ldots, g'_{n_{t(n)}}) \in \mathcal{F}$.

[4] Characteristic function of set S outputs 1 if the input to the function is in set S.

▶ **Definition 22** ($(\mathcal{G},t)$-closure of $\mathcal{F}$ under Strong Composition)**.** *Let $F := \{f_n\}_{n=1}^{\infty} \in \mathcal{F}$, where $f_n : \{0,1\}^n \to \{0,1\}$ and $G = \{g_m\}_{m=1}^{\infty} \in \mathcal{G}$, where $g_m : \{0,1\}^m \to \{0,1\}^m$, be function families. Also let t, p be polynomials. Let $m \in \text{poly}(n)$. Let F^G denote families functions $f_n : \{0,1\}^n \to \{0,1\} \in F$ which contains at most $t(n)$ oracle gates computing $g_m : \{0,1\}^m \to \{0,1\}^m \in G$*

We say that $\mathcal{F}$ is $(\mathcal{G},t)$-closed under compositions if for every $F \in \mathcal{F}$ we have that for all $G, G' \in \mathcal{G}$ and all $g'_1, \ldots, g'_{t(n)} \in G'$, $\mathsf{Replace}(\mathsf{Unroll}(F^G), g'_1, \ldots, g'_{t(n)}) \in \mathcal{F}$.

2.6 Black Box Reductions

▶ **Definition 23** (Black-Box-Reduction)**.** *We say R is an (F, ϵ, δ)-*black-box reduction from a (single bit) non-malleable code, $(\mathsf{E}, \mathsf{D}) = \{(\mathsf{E}_n, \mathsf{D}_n)\}_{n=1}^{\infty}$, to a distributional problem, $(\Psi, L) = \{(\Psi_n, L_n)\}_{n=1}^{\infty}$, *if the following hold:*

1. *For every set of circuits $\{f_n\}_{n=1}^{\infty}$ parameterized by input length n such that f_n achieves $\epsilon(n)$-malleability, for non-negligible ϵ, i.e.*

$$\Pr_{b \xleftarrow{u} \{0,1\}}[\mathsf{D}_n(f_n(\mathsf{E}_n(b))) = 1 - b] > \frac{1}{2} + \epsilon(n),$$

then R^f solves $\{(\Psi_n, L_n)\}_{n=1}^{\infty}$ with advantage $\delta(n)$, where δ is non-negligible. I.e.

$$\Pr_{x \leftarrow \Psi_n}[L_n(x) = R^{\{f_k\}_{k=1}^{\infty}}(x)] > \frac{1}{2} + \delta(n).$$

2. *If $\{f_n\}_{n=1}^{\infty} \in F$, then $R^{\{f_k\}_{k=1}^{\infty}}(x) \in F$.*

We say a reduction R is length-preserving *if R, on input of length n is only allowed to make queries to oracles with security parameter n. Namely,*

$$\Pr_{x \leftarrow \Psi_n}[L_n(x) = R^{f_n}(x)] > \frac{1}{2} + \delta(n).$$

We say a reduction R is approximately length-preserving *if there are polynomials $p(\cdot), q(\cdot)$ such that R, on input of length n is only allowed to make queries to oracles with security parameter $k \in [p(n), q(n)]$. Namely,*

$$\Pr_{x \leftarrow \Psi_n}[L_n(x) = R^{\{f_k\}_{k=p(n)}^{q(n)}}(x)] > \frac{1}{2} + \delta(n).$$

We say a reduction is in NC^1 if it can be written as a family of circuits of $O(\log n)$-depth, $\text{poly}(n)$-size.

3 2-Message NMC against $d-1$ arbitrary errors

In this section, we show that when the message space has size 2 (i.e. single bit messages), non-malleable codes are possible against $d-1$ arbitrary errors, whereas error correcting codes can tolerate at most $(d-1)/2$ arbitrary errors. In the next section, we will show that if the message space is increased to 3 or more, then non-malleable codes are impossible even against $d/2$ errors.

The construction is simply a repetition code (E, D). On input a bit b, E outputs the string b^d (the bit b repeated d times). On input a string $b_1, \ldots, b_d$, D outputs 1 if there is some $i \in [d]$ such that $b_i = 1$. Otherwise, D outputs 0. Note that this code has distance d.

We next prove that (E, D) is a 0-non-malleable code (i.e. the two distributions in the security definition for non-malleable codes–see Definition 3–are identical). Applying Lemma 4, it is sufficient to show that for every tampering function f that modifies at most $d-1$ symbols,

$$\Pr_{b \leftarrow \{0,1\}}[\mathsf{D}(f(\mathsf{E}(b))) = 1 - b] \leq \frac{1}{2},$$

We will use the fact that for the decode algorithm defined above,

$$\Pr[\mathsf{D}(f(\mathsf{E}(1))) = 0] = 0,$$

since a tampering function that modifies at most $d-1$ bits cannot flip a 1 codeword to a tampered codeword that decodes to 0 under D.

Therefore,

$$\begin{aligned}\Pr_{b \leftarrow \{0,1\}}[\mathsf{D}(f(\mathsf{E}(b))) = 1 - b] &= \frac{1}{2}\Pr[\mathsf{D}(f(\mathsf{E}(0))) = 1] + \frac{1}{2}\Pr[\mathsf{D}(f(\mathsf{E}(1))) = 0] \\ &= \frac{1}{2}\Pr[\mathsf{D}(f(\mathsf{E}(0))) = 1] \\ &\leq \frac{1}{2}.\end{aligned}$$

This completes the proof.

4 Unconditional Negative Results

In this section we demonstrate that non-malleable codes are impossible to construct for 3 different classes. The first impossibility result holds for message spaces of size greater than 2 (which is tight, given the result in Section 3), the second and third impossibility results hold even for a single bit.

4.1 Functions that Modify Half the Max Minimum Distance

Let (E, D) be a coding scheme with distance d. Define the class of functions $\mathcal{F}_{d/2-1} = \{f : f$ changes $< d/2$ codeword symbols $\}$. We know that ECC exist, and thus NMC also exist, for $\mathcal{F}_{d/2-1}$ (e.g. Reed Solomon Codes achieve this bound).

We now define the slightly larger class $\mathcal{F}_{d/2} = \{f : f$ changes $\leq d/2$ symbols$\}$. In Theorem 24 we show that even inefficient NMC do not exist for $\mathcal{F}_{d/2}$. Recall that distance for randomized coding schemes is upper bounded by the notion of distance for (non-randomized) codes with the same message/codeword-space parameters. Specifically, for a set of codewords S, we define the distance of S ($\mathsf{dist}(S)$) as the minimum pairwise distance over all pairs of codewords in S. Let $\mathcal{S}$, be the set that consists of all sets S that contain exactly one codeword for each message in the message space. Then the distance of the code is defined as $\max_{S \in \mathcal{S}} \mathsf{dist}(S)$. Refer to definition 2 for formal definition of distance of coding scheme, presented earlier. Intuitively, we want that a code with distance d, ensures that any 2 codewords which are at least d distance apart decode to distinct messages. Therefore, first consider a set of codewords which contains at least one codeword corresponding to each message in the message space such that decoding that specific codeword returns the corresponding message with probability greater than 1/2. Now consider the minimum of pairwise distances for the codewords in this set (say d'). Note that, if the distance of the code is set to d', then for this particular set, we will ensures that any 2 codewords which are at least d' apart decode to distinct messages with high probability. Further, if we take the

maximum over all such distances d' corresponding to each set of codewords and call that value d as the distance of the code, then for any 2 codewords which are at least d apart decode to distinct messages with high probability.

▶ **Theorem 24.** *Let* (E, D) *be a coding scheme with message space of size greater than* 2*, alphabet* Σ *and distance* d*. Then, for any* $\epsilon > 0$*,* (E, D) *is not a* $\frac{1}{8} - \epsilon$*-NMC for* $\mathcal{F}_{d/2}$*.*

Proof. We begin with some notation Given $\alpha, \beta \in \Sigma^n$, we denote by $\|\alpha - \beta\|_0$ the number of positions $i \in [n]$ such that $\alpha_i \neq \beta_i$.

Let $(\mathsf{E} : U \to V, \mathsf{D} : V \to U)$ be a randomized encoding scheme, where $U \subseteq \Sigma^k, V \subseteq \Sigma^n$ and $|U| > 2$.

▷ Claim 25. $\exists x \in U$ such that $\forall c_x \in \mathsf{E}(x)$ there is a $z = z(c_x) \in V$:
1. $\|c_x - z\|_0 \leq \frac{d}{2}$
2. $\Pr[\mathsf{D}(z) \neq x] =\geq \frac{1}{2}$.

Assuming the claim, consider the following tampering function $f \in \mathcal{F}_{d/2}$. Let z_c be the z for each $c \in \mathsf{E}(x^*)$ guaranteed to exist for some $x^* \in U$ by the above claim.

$$f(c) := \begin{cases} z_c & \text{if } c \in \mathsf{E}(x^*) \\ c & \text{otherwise} \end{cases}$$

Let $\Pr_{c_{x^*} \leftarrow \mathsf{E}(x^*)}[\mathsf{D}(z(c_{x^*})) \neq x^*] = p \geq \frac{1}{2}$. Then, $\exists y^* \neq x^* \in U$ such that $\Pr_{c_{x^*} \leftarrow \mathsf{E}(x^*)}[\mathsf{D}(z(c_{x^*})) = y^*] \leq \frac{p}{|U|-1}$, but $\Pr[\mathsf{D}(f(\mathsf{E}(y^*))) = y^*] = 1$. This means that a distribution $D^f_{x^*}$ that exactly agrees with $\mathsf{D}(f(\mathsf{E}(\cdot)))$ on x^* must output same^* or x^* with probability $1-p$ and y^* with probability at most $\frac{p}{|U|-1}$. A distribution $D^f_{y^*}$ that exactly agrees with $\mathsf{D}(f(\mathsf{E}(\cdot)))$ on y^* must output same^* or y^* with probability 1. Thus, any distribution D^f can only agree with $\mathsf{D}(f(\mathsf{E}(\cdot)))$ for both x^* and y^* at most $(1-p) + \frac{p}{|U|-1} \leq 3/4$ fraction of the time (and must have statistical distance at least $1/8$ from one of them), since $p \geq 1/2$ and $|U| > 2$.

Next we prove the claim.

Proof. Suppose for the sake of contradiction that $\forall x \in U, \exists c_x \in \mathsf{E}(x)$ such that $\forall z \in V$ with $\|c_x - z\|_0 \leq \frac{d}{2}$ it is the case that $\Pr[\mathsf{D}(z) \neq x] < \frac{1}{2}$. Fix any such set of codewords corresponding to all messages $S = \{c_x : \forall z \in V \, \|c_x - z\|_0 \leq \frac{d}{2} \implies \Pr[\mathsf{D}(z) \neq x] < \frac{1}{2}\}_{x \in U}$. Note that the distance of S ($\min_{c_x \neq c_y \in S} \|c_x - c_y\|_0$) is at most d (by definition of the distance of a randomized code). Let $c_x \neq c_y \in U$ be two such codewords such that $\|c_x - c_y\| \leq d$. Then, $\exists z \in V$ such that $\|z - c_x\|_0 \leq d/2$ and $\|z - c_y\|_0 \leq d/2$. But then by assumption it follows that $\Pr[\mathsf{D}(z) = x] > \frac{1}{2}$ and $\Pr[\mathsf{D}(z) = y] > \frac{1}{2}$, which is a contradiction because $x \neq y$. ◁

◀

We observe that a randomized coding scheme with message space $\mathcal{M}$, codeword space $\mathcal{C}$ and distance d (as defined above for randomized coding schemes), implies a (possibly inefficient) error correcting code with the same message/codeword space that can correct up to $d/2 - 1$ errors. To see this, note that one can take the set $S \in \mathcal{S}$ (as in the definition of distance for randomized coding schemes) achieving the maximum $\mathsf{dist}(S)$. Since $S \subseteq \mathcal{C}$ contains exactly one codeword for each message in the message space, the set S itself comprises a code with message space $\mathcal{M}$, codeword space $\mathcal{C}$ and distance d. This, in turn, implies a (possibly inefficient) error correcting code with message space $\mathcal{M}$ and codeword space $\mathcal{C}$ that can correct up to $d/2 - 1$ errors. We thus obtain the following corollary:

▶ **Corollary 26.** *Fix a message space $\mathcal{M}$ and a codeword space $\mathcal{C}$. If the optimal (inefficient) error-correcting code for $(\mathcal{M}, \mathcal{C})$ can correct at most t errors, then there is no non-malleable code with message space $\mathcal{M}$ and codeword space $\mathcal{C}$ against tampering class $\mathcal{F}_{t+1}$.*

4.2 Input-Local Functions

We rule out non-malleable codes for *input*-local functions (see Section 2.3 for formal definition), where each input symbol affects ℓ output symbols and ℓ is the locality parameter. We show that even for $\ell = 1$, non-malleability is impossible to achieve. The specific tampering functions used in our proof fix all but one of the codeword symbols to constant values. So we can alternately view this result as building on the previous impossibility result: If one allows fixing codeword symbols to constants, then one cannot achieve non-malleability against functions where even a single output symbol's value depends on the input.

▶ **Theorem 27.** *Let (E, D) be a coding scheme with message space of at least 2 and alphabet Σ. Then, for any $\epsilon > 0$, (E, D) is not a $1/2 - \epsilon$-NMC for $Local_1$.*

Proof. Let $U \subseteq \{0,1\}^k$, $V \subseteq \{0,1\}^n$ where $|U| > 1$. Let $(\mathsf{E} : U \to V, \mathsf{D} : V \to U)$ be non-malleable code. Take $x \neq y \in U$. Consider $c_x = \mathsf{E}(x), c_y = \mathsf{E}(y)$ for some fixed randomness. By correctness $c_x \neq c_y$ and moreover, $\mathsf{D}(c_x) \neq \mathsf{D}(c_y)$. Also let $d := d(c_x, c_y)$ be the distance between c_x and c_y, note that $0 < d \leq n$. Consider $d+1$ codewords starting with, $c_0 = c_x, c_1, \ldots, c_d = c_y$ where $\forall i \in \{0, \ldots, d-1\}$, $d(c_i, c_{i+1}) = 1$. Notice that

$$\mathsf{D}(c_0) \neq \mathsf{D}(c_d) \implies \exists j \in \{0, \ldots, d-1\} : \mathsf{D}(c_j) \neq \mathsf{D}(c_{j+1}).$$

Let $x = \mathsf{D}(c_j)$ and let $y = \mathsf{D}(c_{j+1})$, where $x \neq y$. Now, consider the following $f \in \text{Local}_1$,

$$f(c) = \begin{cases} c_j & \text{if } c \in \mathsf{E}(y) \\ c_{j+1} & \text{otherwise} \end{cases}$$

(Note that all symbols except a single one are constant.) Because they have disjoint support, either $\mathsf{D}(f(\mathsf{E}(x)))$ or $\mathsf{D}(f(\mathsf{E}(y)))$ will be at least 1/2-far from any distribution D^f. ◀

4.3 Functions with Large Output Locality

A function $f : \{0,1\}^n \to \{0,1\}^n$ is $(n - \log(n))$-output-local if each output bit depends on at most $n - \log(n)$ input bits (see Section 2.3 for formal definition). The particular class $\mathcal{F}'$ that we use in our lower bound proof is a subclass of all $(n - \log(n))$-local tampering functions $\mathcal{F}$. Each $f \in \mathcal{F}'$ has the following structure: First, $f_1, \ldots, f_{n-\log(n)}$ (the functions that output the first $n - \log(n)$ bits) are all the same, except that two different bits from $\{0,1\}$ are hardcoded in each. Second, $f_{n-\log(n)+1}, \ldots, f_n$ are also the same, except that a different value from $\{0,1\}$ is hardcoded in each. Finally, the set of input bits upon which $f_1, \ldots, f_{n-\log(n)}$ depend and the set of input bits upon which $f_{n-\log(n)+1}, \ldots, f_n$ depend are fixed. Taken together, this means that the total number of functions f in $\mathcal{F}$ is at most $4^n \cdot 2^{2^{n-\log(n)}}$, so $\log\log|\mathcal{F}'| = n - \log(n)$. On the other hand, Dziembowski et al. [43] showed existence of a $1/n$-non-malleable code for any class $\mathcal{F}$ such that $\log\log|\mathcal{F}| \leq n - 2\log(n)$. Thus, our lower bound result is nearly tight matching the existential upper bound. In our theorem, we prove a more general statement:

▶ **Theorem 28.** *Let (E, D) be a coding scheme with $\mathsf{E} : \{0,1\} \to \{0,1\}^n$ and $\mathsf{D} : \{0,1\}^n \to \{0,1\}$. Let $\mathcal{F}$ be the class of $(n - \log(1/\epsilon) + 2)$-output- local functions, where $1/8 \geq \epsilon \geq 1/2^n$. Then (E, D) is ϵ-malleable with respect to $\mathcal{F}$.*

Note that the parameters discussed above can be obtained by setting $\epsilon = \frac{1}{4n}$.

Additionally, note that non-malleable codes whose decode function D may output values in $\{0, 1, \perp\}$ *imply* non-malleable codes whose decode function D may only output values in $\{0, 1\}$. Thus, ruling out the latter *implies* ruling out the former and only makes our result stronger.

Proof. Fix an arbitrary (E, D) with $\mathsf{E} : \{0,1\} \to \{0,1\}^n$ and $\mathsf{D} : \{0,1\}^n \to \{0,1\}$. Our analysis considers two cases and shows that for each case, there exists $f \in \mathcal{F}$ such that

$$\Pr_{b \leftarrow \{0,1\}}[\mathsf{D}(f(\mathsf{E}(b))) = 1 - b] \geq \frac{1}{2} + \epsilon.$$

By Definition 5, this is sufficient to prove Theorem 28.

We begin with some notation and the proceed to the case analysis. For codeword $c = c_1, \ldots, c_n$, let c^{top} (resp. c^{bot}) denote the first $n - \log(1/\epsilon) + 2$ bits (resp. last $\log(1/\epsilon) - 2$ bits) of c. I.e. $c^{\mathsf{top}} := c_1, \ldots, c_{n-\log(1/\epsilon)+2}$ ($c^{\mathsf{bot}} := c_{n-\log(1/\epsilon)+3}, \ldots, c_n$). For $t \in \mathbb{N}$, let S_t denote the set of all t-bit strings and let U_t denote the uniform distribution over t bits. Assume $n \geq 2$.

Case 1.

$$\Pr_{b \leftarrow \{0,1\}}[\mathsf{D}(c^{\mathsf{top}}||r) = b \mid c \leftarrow \mathsf{E}(b), r \leftarrow U_{\log(1/\epsilon)-2}] \geq 1/2 + \epsilon.$$

Let $c^{*,0} = c_1^{*,0}, \ldots, c_n^{*,0}$ (resp. $c^{*,1} = c_1^{*,1}, \ldots, c_n^{*,1}$) be the lexicographically first string that decodes to 0 (resp. 1) under D (i.e. $\mathsf{D}(c^{*,0}) = 0$ and $\mathsf{D}(c^{*,1}) = 1$.

In this case we consider the following distribution over tampering circuits $f = f_1, \ldots, f_n$, where f_i outputs the i-th bit of f:

Sample $r \leftarrow U_{\log(1/\epsilon)-2}$, construct circuits f_i for each $i \in [n]$, which take input c^{top} and output c'_i. Each f_i does the following:

- Compute $d := \mathsf{D}(c^{\mathsf{top}}||r)$.
- Output $c'_i = c_i^{*,1-d}$.

We now analyze $\Pr_{b \leftarrow \{0,1\}}[\mathsf{D}(f(\mathsf{E}(b))) = 1 - b]$.

$$\begin{aligned}
\Pr_{b \leftarrow \{0,1\}}[\mathsf{D}(f(\mathsf{E}(b))) = 1 - b] &= \Pr_{b \leftarrow \{0,1\}}[f(\mathsf{E}(b)) \text{ outputs } c^{*,1-b}] \\
&= \Pr_{b \leftarrow \{0,1\}}[\mathsf{D}(c^{\mathsf{top}}||r) = b \mid c \leftarrow \mathsf{E}(b), r \leftarrow U_{\log(1/\epsilon)-2}] \\
&\geq 1/2 + \epsilon,
\end{aligned}$$

where the two equalities follow from the definition of the tampering function f, and the inequality follows since we are in Case 1. This implies the ϵ-malleability of (E, D).

Case 2.

$$\Pr_{b \leftarrow \{0,1\}}[\mathsf{D}(c^{\mathsf{top}}||r) = 1 - b \mid c \leftarrow \mathsf{E}(b), r \leftarrow U_{\log(1/\epsilon)-2}] \geq 1/2 - \epsilon.$$

In this case we consider the following distribution over tampering circuits $f = f_1, \ldots, f_n$, where f_i outputs the i-th bit of f:

The first $n - \log(1/\epsilon) + 2$ circuits $(f_1, \ldots, f_{n-\log(1/\epsilon)+2})$ simply compute the identity function: I.e. f_i for $i \in [n - \log(1/\epsilon) + 2]$ takes c_i as input and produces c_i as output.

We next describe the distribution over circuits f_i for $i \in \{n - \log(1/\epsilon) + 3, \ldots, n\}$. Sample $r' \leftarrow [1/(4\epsilon) - 1]$. Construct circuits f_i for each $i \in \{n - \log(1/\epsilon) + 3, \ldots, n\}$ that take input c^{bot} and produce output c'_i. Each f_i does the following:

- Let $r := r_{n-\log(1/\epsilon)+3}, \ldots, r_n$ be the r'-th lexicographic string in the set $S_{\log(1/\epsilon)-2} \setminus \{c^{\mathsf{bot}}\}$[5].
- Output $c'_i = r_i$.

We now analyze $\Pr_{b\leftarrow\{0,1\}}[\mathsf{D}(f(\mathsf{E}(b))) = 1-b]$.
Since we are in Case 2 we have that:

$$\begin{aligned}
1/2-\epsilon &\leq \Pr_{b\leftarrow\{0,1\}}[\mathsf{D}(c^{\mathsf{top}}||r) = 1-b \mid c \leftarrow \mathsf{E}(b), r \leftarrow U_{\log(1/\epsilon)-2}] \\
&= \Pr_{b\leftarrow\{0,1\}}\left[\begin{array}{c} c^{\mathsf{bot}} = r \mid \\ c \leftarrow \mathsf{E}(b), r \leftarrow U_{\log(1/\epsilon)-2}\end{array}\right] \cdot \Pr_{b\leftarrow\{0,1\}}\left[\begin{array}{c} \mathsf{D}(c^{\mathsf{top}}||r) = 1-b \mid \\ c^{\mathsf{bot}} = r \wedge c \leftarrow \mathsf{E}(b), r \leftarrow U_{\log(1/\epsilon)-2}\end{array}\right] \\
&\quad + \Pr_{b\leftarrow\{0,1\}}\left[\begin{array}{c} c^{\mathsf{bot}} \neq r \mid \\ c \leftarrow \mathsf{E}(b), r \leftarrow U_{\log n}\end{array}\right] \cdot \Pr_{b\leftarrow\{0,1\}}\left[\begin{array}{c} \mathsf{D}(c^{\mathsf{top}}||r) = 1-b \mid \\ c^{\mathsf{bot}} \neq r \wedge c \leftarrow \mathsf{E}(b), r \leftarrow U_{\log(1/\epsilon)-2}\end{array}\right] \\
&= \Pr_{b\leftarrow\{0,1\}}\left[\begin{array}{c} c^{\mathsf{bot}} = r \mid \\ c \leftarrow \mathsf{E}(b), r \leftarrow U_{\log(1/\epsilon)-2}\end{array}\right] \cdot 0 \\
&\quad + \Pr_{b\leftarrow\{0,1\}}\left[\begin{array}{c} c^{\mathsf{bot}} \neq r \mid \\ c \leftarrow \mathsf{E}(b), r \leftarrow U_{\log(1/\epsilon)-2}\end{array}\right] \cdot \Pr_{b\leftarrow\{0,1\}}\left[\begin{array}{c} \mathsf{D}(c^{\mathsf{top}}||r) = 1-b \mid \\ c^{\mathsf{bot}} \neq r \wedge c \leftarrow \mathsf{E}(b), r \leftarrow U_{\log(1/\epsilon)-2}\end{array}\right] \\
&= \Pr_{b\leftarrow\{0,1\}}\left[\begin{array}{c} c^{\mathsf{bot}} \neq r \mid \\ c \leftarrow \mathsf{E}(b), r \leftarrow U_{\log(1/\epsilon)-2}\end{array}\right] \cdot \Pr_{b\leftarrow\{0,1\}}\left[\begin{array}{c} \mathsf{D}(c^{\mathsf{top}}||r) = 1-b \mid \\ c^{\mathsf{bot}} \neq r \wedge c \leftarrow \mathsf{E}(b), r \leftarrow U_{\log(1/\epsilon)-2}\end{array}\right] \\
&= (1-4\epsilon) \cdot \Pr_{b\leftarrow\{0,1\}}\left[\begin{array}{c} \mathsf{D}(c^{\mathsf{top}}||r) = 1-b \mid \\ c^{\mathsf{bot}} \neq r \wedge c \leftarrow \mathsf{E}(b), r \leftarrow U_{\log(1/\epsilon)-2}\end{array}\right].
\end{aligned}$$

Note that

$$\Pr_{b\leftarrow\{0,1\}}\left[\begin{array}{c} \mathsf{D}(c^{\mathsf{top}}||r) = 1-b \mid \\ c^{\mathsf{bot}} \neq r \wedge c \leftarrow \mathsf{E}(b), r \leftarrow U_{\log(1/\epsilon)-2}\end{array}\right] = \Pr_{b\leftarrow\{0,1\}}[\mathsf{D}(f(\mathsf{E}(b))) = 1-b].$$

Thus, we have that

$$1/2-\epsilon \leq (1-4\epsilon) \Pr_{b\leftarrow\{0,1\}}[\mathsf{D}(f(\mathsf{E}(b))) = 1-b].$$

Since

$$\begin{aligned}
(1/2+\epsilon)\cdot(1-4\epsilon) &= 1/2+\epsilon-2\epsilon-4\epsilon^2 \\
&\leq 1/2-\epsilon,
\end{aligned}$$

we have that

$$\begin{aligned}
\Pr_{b\leftarrow\{0,1\}}[\mathsf{D}(f(\mathsf{E}(b))) = 1-b] &\geq \frac{1/2-\epsilon}{1-4\epsilon} \\
&\geq 1/2+\epsilon.
\end{aligned}$$

This implies the ϵ-malleability of (E,D). ◀

5 On NMC via BB Reductions

For the formal definition of a (F,ϵ,δ)-*black-box reduction from a (single bit) non-malleable code,* $(\mathsf{E},\mathsf{D}) = \{(\mathsf{E}_n,\mathsf{D}_n)\}_{n=1}^{\infty}$*, to a distributional problem,* $(\Psi, L) = \{(\Psi_n, L_n)\}_{n=1}^{\infty}$, see Definition 23 in Section 2.6.

[5] Recall that, $t \in \mathbb{N}$, let S_t denote the set of all t-bit strings and let U_t denote the uniform distribution over t bits.

A crucial component of our impossibility result will be a lookup circuit that responds to queries submitted by the reduction with hardwired responses. However, we need the lookup circuit to maintain consistency: If the reduction queries the same query multiple times, the same response should be given each time. Such a lookup circuit is trivial to implement with polynomial-size circuits. However, in our case, we require that this lookup circuit is implementable in NC^1. In the following, we first formally define such a lookup circuit and then prove that it is implementable in NC^1.

▶ **Definition 29** (Look-Up Circuit.)**.** *A $(\ell(n), p(n))$ lookup circuit consists of $\ell(n)$ hardwired values of length $p(n)$ bits, denoted $y_1, \ldots, y_{\ell(n)}$. The lookup circuit receives as input $x_1, \ldots, x_{\ell(n)}$, where each x_i has length $p(n)$ bits. The circuit outputs $\ell(n)$ number of $p(n)$-bit strings: $y_{i_1}, \ldots, y_{i_{\ell(n)}}$, where for $j \in [\ell(n)]$, i_j is set to the first index $k \in [\ell(n)]$ such that $x_j = x_k$. For example, on input $x_1, x_2, x_3, x_4, \ldots$, where $x_1 = x_3$ and $x_2 = x_4$, the circuit outputs y_1, y_2, y_1, y_2.*

▶ **Proposition 30.** *For $p(n)$, $\ell(n) = O(n^c)$ for some fixed constant c, there exist polynomial size look-up circuits of depth $O(\log n)$.*

Sketch. The inputs, $x_1, \ldots, x_{\ell-1}$, can be put in sorted order via a circuit of size $O(n^c \log n)$ and depth $O(\log n)$ [10]. Then each sorted x_i can determine if it is the first of that value (if $x_1, \ldots, x_{\ell-1}$ are in sorted order then x_j is determining that there does not exist $x_i = x_j$ such that $i < j$), by comparing only to one neighboring value. This can be done in parallel. Finally, compare x_ℓ to all x_i that pass this test in parallel. If there is such an x_i such that $x_i = x_\ell$, the circuit will output y_i. Otherwise, the circuit will output y_ℓ. ◀

We now present the central technical lemma of the section.

▶ **Lemma 31.** *Assume that $\mathcal{F}$ is $(\mathcal{F}, t, p(n), p(n))$-closed under composition (see Definition 21), and contains $(t(n), p(n))$ look-up circuits for polynomials $t(\cdot)$, $p(\cdot)$.[6] If there is an $(\mathcal{F}, 1/2, \delta(n))$-black-box-reduction making $t(n)$ security parameter-preserving queries from a (single bit) non-malleable code for $\mathcal{F}$, $(\mathsf{E}, \mathsf{D}) = \{(\mathsf{E}_n, \mathsf{D}_n)\}_{n=1}^{\infty}$, to a distributional problem, $(\Psi, L) = \{(= \Psi_n, L_n)\}_{n=1}^{\infty}$, then one of the following must hold:*

1. *(E, D) is $\frac{\delta(n)}{2t(n)}$-malleable by $\mathcal{F}$.*
2. *(Ψ, L) is $(\delta(n)/2)$-easy for $\mathcal{F}$.*

Moreover, if (E, D) is efficient, then it suffices that $\mathcal{F}$ contains such look-up circuits generated in uniform polynomial time.

Proof. Let R be such a security parameter-preserving $(\mathcal{F}, 1/2, \delta(n)$-reduction, for a non-malleable code (E, D) and distributional problem (Ψ, L). Moreover, for security parameter n, let $p(n)$ be the length of the codeword generated by E, where $p(\cdot)$ is a polynomial.

Consider the following tampering functions $\{f_{p(n)}\}_{p(n)}$ whose behavior on a given codeword c is defined as follows (where H is a random function $H : \{0,1\}^{p(n)} \to \{0,1\}^*$ and $H(c)$ is the randomness used by encoding algorithm):

$$f_{p(n)}(c) := \begin{cases} \mathsf{E}_n(1; H(c)) & \text{if } \mathsf{D}_n(c) = 0 \\ \mathsf{E}_n(0; H(c)) & \text{if } \mathsf{D}_n(c) = 1 \end{cases}$$

Since, NMC are perfectly correct, we have (for any choice of H)

$$\Pr_{b \xleftarrow{u} \{0,1\}} [\mathsf{D}_n(f_{p(n)}(\mathsf{E}(b))) = 1 - b] = 1.$$

[6] $p(n)$ corresponds to the length of the codeword outputted by E_n.

Therefore, by our assumption on R we have that for all n,

$$\Pr_{x \leftarrow \Psi_n}[L_n(x) = R^{f_{p(n)}}(x)] \geq \frac{1}{2} + \delta(n).$$

Now, for the j-th oracle query, we define $f'^{,j}_{p(n)}$, a stateful simulation of the output of the tampering function $f_{p(n)}$ on the j-th query. Each $f'^{,j}_{p(n)}$ is a $(j, p(n))$ lookup circuit (with j number of inputs/outputs of length $p(n)$) that hardcodes a random codeword (sampled from $\mathsf{E}(b)$ where b is uniform) as the y_j value.

By our assumption on $\mathcal{F}$ (and R), we have that $\mathsf{Replace}(\mathsf{Unroll}(R^{f_{p(n)}}), f'^{,1}_{p(n)}, \ldots, f'^{,t(n)}_{p(n)}) \in \mathcal{F}$. We will abuse notation and denote the resulting circuit by $R^{f'_{p(n)}}$. So, it suffices to show that the behavior of $R^{f'_{p(n)}}(x)$ is close that of $R^{f^H_{p(n)}}(x)$, for any x, which will imply that $R^{f'_{p(n)}}(x) \in \mathcal{F}$ breaks the distributional problem w.h.p., since $R^{f^H_{p(n)}}(x)$ does. More accurately, if (E, D) is $\frac{\delta(n)}{2t(n)}$-non-malleable by $\mathcal{F}$, then we will show that

$$\forall n \in \mathbb{N}, \forall x \in \{0,1\}^n, \Delta(R^{f'_n}(x); R^{f_n}(x)) \leq \delta(n)/2.$$

By the above, this then implies that (Ψ, L) is $(\delta(n)/2)$-easy for $\mathcal{F}$.

To show that the outputs of $R^{f'_{p(n)}}(x)$ and $R^{f^H_{p(n)}}(x)$ are close, we will use a hybrid argument, reducing to the $\frac{\delta(n)}{2t(n)}$-non-malleablity of (E, D) at every step.

In the i-th hybrid, the function $f^{(i),j}_{p(n)}$ responding to the j-th query is a $(j, p(n))$ look-up circuit that hardcodes values $y^i_1, \ldots, y^i_j$. For $k \in [t = t(n)]$, the y^i_k values are sampled as follows: For $k \in [t - i]$, y^i_k is sampled as by $f^H_{p(n)}$. For $k > t - i$, y^i_k is a random encoding of a random bit. The concatenation of the t circuits for each query is denoted by $f^{(i)}_{p(n)}$. Clearly, $f^{(0)}_{p(n)} \equiv f_{p(n)}$ and $f^{(t)}_{p(n)} \equiv f'_{p(n)}$.

We will show that for all $x \in \{0,1\}^n$ (and any fixing of random coins r for R) $\Delta(R^{f^{(i)}_{p(n)}}(x); R^{f^{(i-1)}_{p(n)}}(x)) \leq \frac{\delta(n)}{2t(n)}$ (for $i \in [t(n)]$), which proves the claim above.($R^{f^{(0)}_{p(n)}}(x)$ has advantage $\delta(n)$ and in each of the subsequent $t(n)$ hybrids we lose at most an $\epsilon(n)$ factor.)

Suppose not, then there exists an x (and random coins r, if R is randomized) such that R's behavior differs with respect to $f^{(i)}_{p(n)}$ and $f^{(i-1)}_{p(n)}$: $|\Pr[R^{f^{(i)}_{p(n)}}(x) = 1] - \Pr[R^{f^{(i-1)}_{p(n)}}(x) = 1]| \geq \frac{\delta(n)}{2t(n)}$.

Note that for fixed random function H (that generates the random coins used to sample the y_j values) $f^{(i)}_{p(n)}$ and $f^{(i-1)}_{p(n)}$ differ solely on the response to $(t-i)$-th query. So, fix x, H and all but the $(t-i)$-th value y^i_{t-i} and "hardcode" all other y_k values in both cases. The reason that we can hardcode the y_j values except for the $(t-i)$-th response is the following: Clearly, up to the $(t-i)$-th query, the responses can be fully hardcoded since x is fixed and so all the queries and responses can also be fixed. The y_j values hardcoded in the $(t-i+1)$-st lookup circuit and on can also be fixed, since in both $f^{(i)}_{p(n)}$ and $f^{(i-1)}_{p(n)}$, the $(t-i+1)$-st value of y_j and on is a random codeword, that does not depend on the value encoded in the query submitted by the reduction. Let $s_{H,x}$ denote the value encoded in the $(t-i)$-th query in this hardcoded variant of the hybrid. Note that the value of $s_{H,x}$ is also fixed.

1. In $R^{f^{(i-1)}_{p(n)}}(x)$ all values up to the $(t-i)$-th response are hardcoded. The $(t-i)$-th response, which will be a random encoding of bit $1 - s_{H,x}$, is not hardcoded. All the other responses are computed by lookup circuits with hardwired y_j values.

2. In $R^{f^{(i)}_{p(n)}}(x)$, all values up to the $(t-i)$-th response are hardcoded. The $(t-i)$-th response, which will be a random encoding of a random bit, is not hardcoded. All the other responses are computed by lookup circuits with hardwired y_j values.

Thus, we will treat the above as a new function $R'_{H,x}(\cdot)$ that takes as input just the response to the $(t-i)$-th query and returns some value. Note that $R'_{H,x}(\cdot)$ is in $\mathcal{F}$, since it can be viewed as the circuit $R^{f^{(i)}_{p(n)}}$, with queries/responses to $f^{(i),j}$, $j \in [t-i-1]$ hardcoded, the $(t-i)$-th query hardcoded, the $(t-i)$-th value y^i_{t-i} as the input to the circuit, and for $j > t-i$, the $f^{(i),j}$ functions as lookup circuits contained in $\mathcal{F}$. Moreover, by the above, $R'_{H,x}(\cdot)$ distinguishes random codewords that encode the bit $1-s_{H,x}$ from random codewords that encode a random bit with advantage $\epsilon(n)$. Specifically,

$$\Pr[R'_{H,x}(c)=1 \mid c \leftarrow \mathsf{E}_n(1-s_{H,x})] - \Pr[R'_{H,x}(c)=1 \mid c \leftarrow \mathsf{E}_n(b), b \leftarrow \{0,1\}] \geq \frac{\delta(n)}{2t(n)}.$$

By standard manipulation, the above is equivalent to:

$$\frac{1}{2}\cdot\Pr[R'_{H,x}(c)=1 \mid c \leftarrow \mathsf{E}_n(1-s_{H,x})] + \frac{1}{2}\cdot\Pr[R'_{H,x}(c)=0 \mid c \leftarrow \mathsf{E}_n(s_{H,x})] \geq \frac{1}{2} + \frac{\delta(n)}{2t(n)}.$$

This implies that we can use $R'_{H,x}$ to construct a distribution over tampering functions in $\mathcal{F}$ that successfully break (E,D). Details follows.

Let $c_{s_{H,x}}$ be a codeword encoding bit $s_{H,x}$ and let $c_{1-s_{H,x}}$ be a codeword encoding bit $1-s_{H,x}$. Define $\hat{f}_{H,x}$ as follows: $\hat{f}_{H,x}$ hardcodes $c_{s_{H,x}}$ and $c_{1-s_{H,x}}$. On input (codeword) c,

- If $R'_{H,x}(c)=1$, output $c_{s_{H,x}}$;
- Otherwise, output $c_{1-s_{H,x}}$.

We now analyze

$$\Pr_{b\leftarrow\{0,1\}}[\mathsf{D}_n(\hat{f}_{H,x}(\mathsf{E}_n(b))) = 1-b].$$

$$\begin{aligned}
\Pr_{b\leftarrow\{0,1\}}[\mathsf{D}(\hat{f}_{H,x}(\mathsf{E}(b))) = 1-b] &= \Pr[b = 1-s_{H,x}]\cdot\Pr[R'_{H,x}(c)=1 \mid c\leftarrow\mathsf{E}_n(1-s_{H,x})] \\
&\quad + \Pr[b=s_{H,x}]\cdot\Pr[R'_{H,x}(c)=0 \mid c\leftarrow\mathsf{E}_n(s_{H,x})] \\
&= \frac{1}{2}\cdot\Pr[R'_{H,x}(c)=1 \mid c\leftarrow\mathsf{E}_n(1-s_{H,x})] \\
&\quad + \frac{1}{2}\cdot\Pr[R'_{H,x}(c)=0 \mid c\leftarrow\mathsf{E}_n(s_{H,x})] \\
&\geq \frac{1}{2} + \frac{\delta(n)}{2t(n)}.
\end{aligned}$$

But, the above implies that (E,D) is $\frac{\delta(n)}{2t(n)}$-malleable for $\mathcal{F}$.

Therefore, we conclude that either (E,D) is $\frac{\delta(n)}{2t(n)}$-malleable for $\mathcal{F}$ or the distributional problem, $(\Psi, L) = \{(\Psi_n, L_n)\}_{n=1}^{\infty}$ is $(\delta(n)/2)$-easy for $\mathcal{F}$. ◀

The following corollary holds since NC^1 is $(\mathsf{NC}^1, t, p(n), p(n))$-closed under composition (for all polynomials $p(\cdot)$), and NC^1 contains $(t(n), p(n))$ lookup circuits for any polynomials $t(\cdot)$, $p(\cdot)$.

▶ **Corollary 32.** *If there is an* $(\mathsf{NC}^1, 1/2, \delta(n))$*-black-box-reduction making* $t(n)$ *security parameter preserving queries from a (single bit) non-malleable code for* NC^1*,* $(\mathsf{E}, \mathsf{D}) = \{(\mathsf{E}_n, \mathsf{D}_n)\}_{n=1}^{\infty}$*, to a distributional problem,* $(\Psi, L) = \{(\Psi_n, L_n)\}_{n=1}^{\infty}$*, then one of the following must hold:*

1. (E, D) *is* $\frac{\delta(n)}{2t(n)}$*-malleable by* NC^1*.*
2. (Ψ, L) *is* $(\delta(n)/2)$*-easy for* NC^1*.*

▶ Note 33. The proof of Lemma 31 (as well as the other proofs in this section), does not extend to cases in which the reduction R is outside in the class of tampering functions $\mathcal{F}$. Specifically, in the hybrid arguments, we require that $R'_{H,x}(\cdot)$ is in $\mathcal{F}$. In particular, our proof approach does not extend to proving impossibility of constructing a (single bit) non-malleable code for $\mathcal{F}$, from a distributional problem, (Ψ, L) that is hard for some larger class $\mathcal{F}$. E.g. our techniques do not allow us to rule out constructions of non-malleable codes for NC^1 from a distributional problem that is hard for NC^2. Our techniques also do not rule out constructions of non-malleable codes for $\mathcal{F}$ from an "incompressibility"-type assumption, such as those used in the recent work of [16]. Briefly, if a function ψ is incompressible by circuit class $\mathcal{F}$, it means that for $t \ll n$, for any *computationally unbounded* Boolean function $D : \{0,1\}^t \to \{0,1\}$ and any $F : \{0,1\}^n \to \{0,1\}^t \in \mathcal{F}$, the output of $D \circ F(x_1, \ldots, x_n)$ is uncorrelated with $\psi(x_1, \ldots, x_n)$ (over uniform choice of $x_1, \ldots, x_n$). In our case, this would mean that the reduction R is allowed oracle access to a computationally unbounded Boolean function D, since the hardness assumption would still be broken by the reduction as long as $R \in \mathcal{F}$ and the query made to D has length $t \ll n$. Since R composed with D is clearly outside the tampering class $\mathcal{F}$, our proof approach does not apply in the incompressibility setting.

▶ Note 34. We can extend Lemma 31 to rule out $(u(n), \ell(n))$*-approximately* security parameter preserving reductions by allowing our reduction access to a greater range of inefficient/simulated tampering functions (defined in the same manner as above): $\{f_k\}_{k=\ell(n)}^{u(n)}$ and $\{f'_k\}_{k=\ell(n)}^{u(n)}$. In this case, we can, WLOG, conflate the security parameter queried to the oracle with the length of the query made to the oracle. However, we now require for our proof that $\mathcal{F}$ is $(\mathcal{F}, t, \ell, u)$-closed under composition and contains look-up circuits with $t(n)$ inputs, consisting of $\ell(n)$ to $u(n)$ number of bits, for polynomials $t(\cdot)$, $\ell(\cdot), u(\cdot)$.

▶ **Lemma 35.** *Assume* $\mathcal{F}$ *is* $(\mathcal{F}, t, \ell, u)$*-closed under composition (see Definition 21) and contains* $(t(n), p(n))$ *look-up circuits for polynomials* $t(\cdot)$*,* $p(\cdot)$*. If there is an* $(\mathcal{F}, 1/2, \delta(n))$*-black-box-reduction making* $t(n)$ *number of* $(\ell(n), u(n))$*-approximately length preserving queries, from a (single bit) non-malleable code for* $\mathcal{F}$*,* $(\mathsf{E}, \mathsf{D}) = \{(\mathsf{E}_n, \mathsf{D}_n)\}_{n=1}^{\infty}$*, to a distributional problem,* $(\Psi, L) = \{(\Psi_n, L_n)\}_{n=1}^{\infty}$*, then one of the following must hold:*

1. (E, D) *is* $\frac{\delta(n)}{2t(n)}$*-malleable by* $\mathcal{F}$*.*
2. (Ψ, L) *is* $(\delta(n)/2)$*-easy for* $\mathcal{F}$*.*

Moreover, if (E, D) *is efficient, then for the conclusion to hold it suffices that* $\mathcal{F}$ *contains such look-up circuits generated that are generated uniform polynomial time.*

The following corollary holds since NC^1 is $(\mathsf{NC}^1, t, \ell, u)$-closed under composition, where $\ell(n) = n^\gamma$, for any constant $\gamma \leq 1$, $u(n) = n^c$, for any constant $c \geq 1$ and NC^1 contains look-up circuits with $t(n)$ number of inputs of length $\ell(n)$ to $u(n)$-bits for polynomials $t(\cdot)$, $\ell(\cdot), u(\cdot)$.

▶ **Corollary 36.** *Fix constants $\gamma \leq 1$, $c \geq 1$. If there is an $(\mathsf{NC}^1, 1/2, \delta(n))$-black-box-reduction making $t(n)$ (n^γ, n^c)-approximately length preserving queries from a (single bit) non-malleable code for NC^1, $(\mathsf{E}, \mathsf{D}) = \{(\mathsf{E}_n, \mathsf{D}_n)\}_{n=1}^\infty$, to a distributional problem, $(\Psi, L) = \{(\Psi_n, L_n)\}_{n=1}^\infty$, then one of the following must hold:*

1. *(E, D) is $\frac{\delta(n)}{2t(n)}$-malleable by NC^1.*
2. *(Ψ, L) is $(\delta(n)/2)$-easy for NC^1.*

We extend to non-security parameter preserving reductions, but require a stronger compositional property for the tampering class $\mathcal{F}$. As for approximate security parameter preserving reductions, WLOG we may conflate the security parameter queried to the oracle with the length of the query made to the oracle.

▶ **Lemma 37.** *Let $\mathcal{F}$ be closed under strong composition (see Definition 22) and contain $(t(n), u(n))$ lookup circuits. If for every non-negligible ϵ, there is an $(\mathcal{F}, \epsilon, \delta(n))$-black-box-reduction (for some non-negligible δ) making $t(n)$ queries from an (single bit) $\epsilon(n)$-non-malleable code for $\mathcal{F}$, $(\mathsf{E}, \mathsf{D}) = \{(\mathsf{E}_n, \mathsf{D}_n)\}_{n=1}^\infty$, to a distributional problem, $(\Psi, L) = \{(\Psi_n, L_n)\}_{n=1}^\infty$, then (Ψ, L) is not $(\delta(n) - t(n) \cdot \epsilon(n))$-hard for $\mathcal{F}$.*

Proof. Let $\mathcal{S} := \{1, 2^1, 2^{2^1}, 2^{2^{2^1}}, \ldots\}$. Let $\epsilon(n)$ be the following non-negligible function:

$$\epsilon(n) := \begin{cases} \frac{1}{4} & \text{if } n \in \mathcal{S} \\ 0 & \text{if } n \notin \mathcal{S} \end{cases}$$

Assume there is some reduction R that succeeds with non-negligible probability $\delta := \delta(n)$ for this ϵ. Since δ is non-negligible, there must be an infinite set $\mathcal{S}'$ such that $\delta(n) \geq 1/n^c$ for some constant c and for all $n \in \mathcal{S}'$.

WLOG, we may assume that the reduction R, on input of length n, queries at most a single input length $\ell(n) \in \omega(\log(n))$, whereas all other queries are of input length $O(\log(n))$ (since we may assume the oracle simply returns strings of all 0's on any input of length $k \notin \mathcal{S}$). Additionally, we may assume that (1) $\ell(n)$ is polynomial in n (since otherwise the reduction does not have time to even write down the query) and (2) for any $k \in \mathbb{N}$, the size of the set $\ell^{-1}(k) \cap \mathcal{S}'$ is finite (otherwise we can hardcode all possible query/responses for a particular input length k into the reduction–which is constant size since k is constant–and obtain a circuit that breaks the underlying hard problem on an infinite number of input lengths). Moreover, we assume WLOG that $\ell(n) < n$, since otherwise our previous proof holds.

Since by assumption $\mathcal{F}$ is HCS-amenable, it means that Impagliazzo's hardcore set holds for adversaries in $\mathcal{F}$. Specifically, for random codewords $c \leftarrow \mathsf{E}_{\ell(n)}(b), b \leftarrow \{0,1\}$ of length $\ell = \ell(n)$ s.t. $\ell(n) < n$, there are two possible cases:

1. For infinitely many $n \in \mathcal{S}'$ (this set of values is denoted by $\mathcal{S}'' \subseteq \mathcal{S}'$), there is some adversary in $\mathcal{F} := \{\mathcal{F}_n\}_{n \in \mathbb{N}}$ that outputs $\mathsf{D}_{\ell(n)}(c)$ with probability at least $3/4$[7].
2. For infinitely many $n \in \mathcal{S}'$ (this set of values is denoted by $\mathcal{S}'' \subseteq \mathcal{S}'$), there is some hardcore set $\mathcal{H}$ of size at least $\epsilon'(n) \cdot 2^\ell$, where $\epsilon'(n) = \frac{1}{2 \cdot n^c \cdot t(n)}$ such that every adversary in $\mathcal{F} := \{\mathcal{F}_n\}_{n \in \mathbb{N}}$ outputs $\mathsf{D}_{\ell(n)}(c)$ with probability at most $1/2 + \epsilon'(n)$, when c is chosen at random from $\mathcal{H}$[8].

[7] Note that $\mathsf{D}_{\ell(n)}(c)$ takes inputs of length $\ell(n)$, whereas $\mathcal{F}_n$ takes inputs of length n. We can easily resolve this discrepancy by padding inputs of length $\ell(n)$ up to n.

[8] Again, the input c to $\mathsf{D}_{\ell(n)}$ has length $\ell(n)$ while $\mathcal{F}_n$ takes inputs of length n. As above, we resolve the discrepancy by padding inputs of length $\ell(n)$ up to n.

In Case 1, we set the tampering function $\{f_k\}_k$ to use the circuit described above to decode a random codeword with prob 3/4 and then chooses a random encoding of 0 or 1 appropriately. Additionally, f_k only responds if $k \in \mathcal{S}$. Clearly, f_k succeeds with non-negligible probability ϵ. Since the ϵ function remains the same, we know that δ and ℓ, $\mathcal{S}$, $\mathcal{S}'$ remain the same.

In this case, as in the previous proof, we can switch to a simulated tampering function Sim, which responds with $f_{\ell(n)}$ on query input length $\ell(n)$ and hardcodes all responses for all possible queries R makes to f_k with input lengths $k = k(n) \in O(\log(n))$.

Note that since we are in Case 1, for infinitely many input lengths–input lengths $n \in \mathcal{S}''$–to R, R^{Sim}, is a circuit in $\mathcal{F}_n$, since $\mathcal{F}_n$ strongly composes. Additionally, the behavior of R^{Sim} is identical to the behavior of $R^{\{f_k\}_k}$. Moreover, since f_k succeeds with non-negligible ϵ, by assumption on R, it means that for all $n \in \mathcal{S}'$, $R^{f_{\ell(n)}}$ agrees with (Ψ, L) with probability $1/2+1/n^c$. But then we must have that for infinitely many $n \in \mathcal{S}'$–input lengths $n \in \mathcal{S}''$–R^{Sim} agrees with (Ψ, L) with probability $1/2 + 1/n^c$ and $R^{\mathsf{Sim}} \in \mathcal{F}_n$. So (Ψ, L) is $(\delta'(n))$-easy for $\mathcal{F}$, where

$$\delta'(n) := \begin{cases} \frac{1}{n^c} & \text{if } n \in \mathcal{S}'' \\ 0 & \text{if } n \notin \mathcal{S}'' \end{cases}$$

In Case 2, we set the tampering function $\{f_k\}_k$ to decode the query submitted by the reduction R and respond with a random encoding from the hardcore set described above (if it exists), which decodes to 0 or 1 as appropriate. Specifically, the hardcore set $\mathcal{H}$ is defined as follows: f_k sets n^* to be equal to the lexicographically first element in the (finite) set $\ell^{-1}(k) \cap \mathcal{S}''$[9], and chooses the lexicographically first set $\mathcal{H}$ of size $\epsilon'(n^*) \cdot 2^{\ell(n^*)} = \epsilon'(n^*) \cdot 2^k$ for which every adversary in $\mathcal{F}_n$ outputs $\mathsf{D}_{\ell(n^*)}(c)$ with probability at most $1/2 + \epsilon'(n^*)$, when c is chosen at random from $\mathcal{H}$. If $\ell^{-1}(k) \cap \mathcal{S}' = \emptyset$ or there is no such hardcore set $\mathcal{H}$, then f_k applies the trivial breaking strategy described above (decoding the input and responding with a random encoding of 0 or 1 as appropriate). Moreover, f_k responds only if $k \in \mathcal{S}$. Since the ϵ function remains the same in this case as well, the δ function also remains the same. Thus, for $n \in \mathcal{S}'$, $R^{f_{\ell(n)}}$ must still agree with (Ψ, L) with probability $1/2 + 1/n^c$.

In this case, as in the previous proof, we can switch to a simulated tampering function Sim that does not decode but rather chooses a random codeword from the hardcode set $\mathcal{H}$ (which again we can hardcode in using lookup circuits as before). Moreover, for queries R makes to Sim with input lengths $k = k(n) \in O(\log(n))$, all responses for all possible queries c are hardcoded into Sim. Now, for infinitely many $n \in \mathcal{S}'$–input lengths $n \in \mathcal{S}''$–R's behavior should be $t(n) \cdot \epsilon'(n)$-close when interacting with $\{f_k\}_k$ versus Sim, since otherwise in each hybrid step we can construct a distinguishing circuit in $\mathcal{F}_n$ (as in the previous proof) contradicting the guaranteed hardness of the hardcore set. Finally, we must argue that for infinitely many $n \in \mathcal{S}'$–input lengths $n \in \mathcal{S}''$–R composed with Sim is in the class $\mathcal{F}$. But due to the fact that $\mathcal{F}$ is $(\mathcal{F}, t)$-closed under strong composition, this occurs whenever the reduction is instantiated with security parameter $n \in \mathcal{S}''$, where n is the lexicographically first element in the set $\ell^{-1}(\ell(n)) \cap \mathcal{S}''$. Since n is always contained in $\ell^{-1}(\ell(n))$, since the size of $\ell^{-1}(\ell(n)) \cap \mathcal{S}'$ is finite and since the size of $\mathcal{S}''$ is infinite, there will be infinitely many $n \in \mathcal{S}''$ for which this event occurs. Thus, for infinitely many $n \in \mathcal{S}''$ (denote this set of values by $\widetilde{\mathcal{S}}$, $R^{\{f_k\}_k}$ agrees with (Ψ, L) with probability $1/2 + 1/n^c$ and R^{Sim} is $t(n) \cdot \epsilon'(n) \leq 1/2n^c$-close to $R^{\{f_k\}_k}$. So we conclude that (Ψ, L) is $(\delta'(n))$-easy for $\mathcal{F}$, where

$$\delta'(n) := \begin{cases} \frac{1}{2n^c} & \text{if } n \in \widetilde{\mathcal{S}} \\ 0 & \text{if } n \notin \widetilde{\mathcal{S}} \end{cases}$$

◀

[9] Note that it is finite since $\ell^{-1}(k) \cap \mathcal{S}'$ is finite and $\mathcal{S}'' \subseteq \mathcal{S}'$.

The following corollary holds since NC is (NC, t)-closed under strong composition and Impagliazzo's HCS holds for NC.

▶ **Corollary 38.** *If for every non-negligible $\epsilon = \epsilon(\cdot)$, there is an $(\mathsf{nu}-\mathsf{NC}, \epsilon, \delta)$-black-box-reduction, for some non-negligible $\delta = \delta(\cdot)$, making $t(n)$ queries from a (single bit) non-malleable code for $\mathsf{nu}-\mathsf{NC}$, $(\mathsf{E}, \mathsf{D}) = \{(\mathsf{E}_n, \mathsf{D}_n)\}_{n=1}^{\infty}$, to a distributional problem, $(\Psi, L) = \{(\Psi_n, L_n)\}_{n=1}^{\infty}$, then (Ψ, L) is $(\delta'(n))$-easy for NC, for some non-negligible $\delta' = \delta'(\cdot)$.*

References

1. Masayuki Abe, Jens Groth, and Miyako Ohkubo. Separating Short Structure-Preserving Signatures from Non-interactive Assumptions. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 628–646. Springer, Heidelberg, December 2011. doi:10.1007/978-3-642-25385-0_34.
2. Divesh Aggarwal, Shashank Agrawal, Divya Gupta, Hemanta K. Maji, Omkant Pandey, and Manoj Prabhakaran. Optimal Computational Split-state Non-malleable Codes. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A, Part II*, volume 9563 of *LNCS*, pages 393–417. Springer, Heidelberg, January 2016. doi:10.1007/978-3-662-49099-0_15.
3. Divesh Aggarwal, Yevgeniy Dodis, Tomasz Kazana, and Maciej Obremski. Non-malleable Reductions and Applications. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *47th ACM STOC*, pages 459–468. ACM Press, June 2015. doi:10.1145/2746539.2746544.
4. Divesh Aggarwal, Yevgeniy Dodis, and Shachar Lovett. Non-malleable codes from additive combinatorics. In David B. Shmoys, editor, *46th ACM STOC*, pages 774–783. ACM Press, May/June 2014. doi:10.1145/2591796.2591804.
5. Divesh Aggarwal, Nico Döttling, Jesper Buus Nielsen, Maciej Obremski, and Erick Purwanto. Continuous Non-Malleable Codes in the 8-Split-State Model. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part I*, volume 11476 of *LNCS*, pages 531–561. Springer, Heidelberg, May 2019. doi:10.1007/978-3-030-17653-2_18.
6. Divesh Aggarwal, Stefan Dziembowski, Tomasz Kazana, and Maciej Obremski. Leakage-Resilient Non-malleable Codes. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part I*, volume 9014 of *LNCS*, pages 398–426. Springer, Heidelberg, March 2015. doi:10.1007/978-3-662-46494-6_17.
7. Divesh Aggarwal and Maciej Obremski. Inception makes non-malleable codes shorter as well! Cryptology ePrint Archive, Report 2019/399, 2019. URL: https://eprint.iacr.org/2019/399.
8. Shashank Agrawal, Divya Gupta, Hemanta K. Maji, Omkant Pandey, and Manoj Prabhakaran. A Rate-Optimizing Compiler for Non-malleable Codes Against Bit-Wise Tampering and Permutations. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part I*, volume 9014 of *LNCS*, pages 375–397. Springer, Heidelberg, March 2015. doi:10.1007/978-3-662-46494-6_16.
9. Shashank Agrawal, Divya Gupta, Hemanta K. Maji, Omkant Pandey, and Manoj Prabhakaran. Explicit Non-malleable Codes Against Bit-Wise Tampering and Permutations. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 538–557. Springer, Heidelberg, August 2015. doi:10.1007/978-3-662-47989-6_26.
10. Miklós Ajtai, János Komlós, and Endre Szemerédi. An O(n log n) Sorting Network. In David S. Johnson, Ronald Fagin, Michael L. Fredman, David Harel, Richard M. Karp, Nancy A. Lynch, Christos H. Papadimitriou, Ronald L. Rivest, Walter L. Ruzzo, and Joel I. Seiferas, editors, *Proceedings of the 15th Annual ACM Symposium on Theory of Computing, 25-27 April, 1983, Boston, Massachusetts, USA*, pages 1–9. ACM, 1983. doi:10.1145/800061.808726.
11. Marcin Andrychowicz, Ivan Damgård, Stefan Dziembowski, Sebastian Faust, and Antigoni Polychroniadou. Efficient Leakage Resilient Circuit Compilers. In Kaisa Nyberg, editor, *CT-RSA 2015*, volume 9048 of *LNCS*, pages 311–329. Springer, Heidelberg, April 2015. doi:10.1007/978-3-319-16715-2_17.

12 Benny Applebaum. *Cryptography in Constant Parallel Time*. Information Security and Cryptography. Springer, 2014. doi:10.1007/978-3-642-17367-7.

13 Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography in NC^0. In *45th FOCS*, pages 166–175. IEEE Computer Society Press, October 2004. doi:10.1109/FOCS.2004.20.

14 Benny Applebaum and Pavel Raykov. On the Relationship Between Statistical Zero-Knowledge and Statistical Randomized Encodings. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part III*, volume 9816 of *LNCS*, pages 449–477. Springer, Heidelberg, August 2016. doi:10.1007/978-3-662-53015-3_16.

15 Marshall Ball, Dana Dachman-Soled, Siyao Guo, Tal Malkin, and Li-Yang Tan. Non-Malleable Codes for Small-Depth Circuits. In Mikkel Thorup, editor, *59th FOCS*, pages 826–837. IEEE Computer Society Press, October 2018. doi:10.1109/FOCS.2018.00083.

16 Marshall Ball, Dana Dachman-Soled, Mukul Kulkarni, Huijia Lin, and Tal Malkin. Non-Malleable Codes Against Bounded Polynomial Time Tampering. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part I*, volume 11476 of *LNCS*, pages 501–530. Springer, Heidelberg, May 2019. doi:10.1007/978-3-030-17653-2_17.

17 Marshall Ball, Dana Dachman-Soled, Mukul Kulkarni, and Tal Malkin. Non-malleable Codes for Bounded Depth, Bounded Fan-In Circuits. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 881–908. Springer, Heidelberg, May 2016. doi:10.1007/978-3-662-49896-5_31.

18 Marshall Ball, Dana Dachman-Soled, Mukul Kulkarni, and Tal Malkin. Non-malleable Codes from Average-Case Hardness: AC^0, Decision Trees, and Streaming Space-Bounded Tampering. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part III*, volume 10822 of *LNCS*, pages 618–650. Springer, Heidelberg, April/May 2018. doi:10.1007/978-3-319-78372-7_20.

19 Marshall Ball, Siyao Guo, and Daniel Wichs. Non-Malleable Codes for Decision Trees. *IACR Cryptology ePrint Archive*, 2019:379, 2019.

20 Boaz Barak and Mohammad Mahmoody-Ghidary. Merkle puzzles are optimal - an $O(n^2)$-query attack on any key exchange from a random oracle. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 374–390. Springer, Heidelberg, August 2009. doi:10.1007/978-3-642-03356-8_22.

21 Nir Bitansky, Shafi Goldwasser, Abhishek Jain, Omer Paneth, Vinod Vaikuntanathan, and Brent Waters. Time-Lock Puzzles from Randomized Encodings. In Madhu Sudan, editor, *ITCS 2016*, pages 345–356. ACM, January 2016. doi:10.1145/2840728.2840745.

22 Harry Buhrman and Ronald de Wolf. Complexity measures and decision tree complexity: a survey. *Theor. Comput. Sci.*, 288(1):21–43, 2002. doi:10.1016/S0304-3975(01)00144-X.

23 Nishanth Chandran, Vipul Goyal, Pratyay Mukherjee, Omkant Pandey, and Jalaj Upadhyay. Block-Wise Non-Malleable Codes. In Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi, editors, *ICALP 2016*, volume 55 of *LIPIcs*, pages 31:1–31:14. Schloss Dagstuhl, July 2016. doi:10.4230/LIPIcs.ICALP.2016.31.

24 Nishanth Chandran, Bhavana Kanukurthi, and Rafail Ostrovsky. Locally Updatable and Locally Decodable Codes. In Yehuda Lindell, editor, *TCC 2014*, volume 8349 of *LNCS*, pages 489–514. Springer, Heidelberg, February 2014. doi:10.1007/978-3-642-54242-8_21.

25 Nishanth Chandran, Bhavana Kanukurthi, and Srinivasan Raghuraman. Information-Theoretic Local Non-malleable Codes and Their Applications. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A, Part II*, volume 9563 of *LNCS*, pages 367–392. Springer, Heidelberg, January 2016. doi:10.1007/978-3-662-49099-0_14.

26 Eshan Chattopadhyay, Vipul Goyal, and Xin Li. Non-malleable extractors and codes, with their many tampered extensions. In Daniel Wichs and Yishay Mansour, editors, *48th ACM STOC*, pages 285–298. ACM Press, June 2016. doi:10.1145/2897518.2897547.

27 Eshan Chattopadhyay, Bhavana Kanukurthi, Sai Lakshmi Bhavana Obbattu, and Sruthi Sekar. Privacy Amplification from Non-malleable Codes. Cryptology ePrint Archive, Report 2018/293, 2018. URL: https://eprint.iacr.org/2018/293.

28 Eshan Chattopadhyay and Xin Li. Non-malleable codes and extractors for small-depth circuits, and affine functions. In Hamed Hatami, Pierre McKenzie, and Valerie King, editors, *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 1171–1184. ACM, 2017. doi:10.1145/3055399.3055483.

29 Eshan Chattopadhyay and Xin Li. Non-malleable codes and extractors for small-depth circuits, and affine functions. In Hamed Hatami, Pierre McKenzie, and Valerie King, editors, *49th ACM STOC*, pages 1171–1184. ACM Press, June 2017.

30 Eshan Chattopadhyay and Xin Li. Non-Malleable Extractors and Codes for Composition of Tampering, Interleaved Tampering and More. Cryptology ePrint Archive, Report 2018/1069, 2018. URL: https://eprint.iacr.org/2018/1069.

31 Eshan Chattopadhyay and David Zuckerman. Non-malleable Codes against Constant Split-State Tampering. In *55th FOCS*, pages 306–315. IEEE Computer Society Press, October 2014. doi:10.1109/FOCS.2014.40.

32 Binyi Chen, Yilei Chen, Kristina Hostáková, and Pratyay Mukherjee. Continuous Space-Bounded Non-malleable Codes from Stronger Proofs-of-Space. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 467–495. Springer, Heidelberg, August 2019. doi:10.1007/978-3-030-26948-7_17.

33 Mahdi Cheraghchi and Venkatesan Guruswami. Capacity of non-malleable codes. In Moni Naor, editor, *ITCS 2014*, pages 155–168. ACM, January 2014. doi:10.1145/2554797.2554814.

34 Mahdi Cheraghchi and Venkatesan Guruswami. Non-malleable Coding against Bit-Wise and Split-State Tampering. In Yehuda Lindell, editor, *TCC 2014*, volume 8349 of *LNCS*, pages 440–464. Springer, Heidelberg, February 2014. doi:10.1007/978-3-642-54242-8_19.

35 Sandro Coretti, Antonio Faonio, and Daniele Venturi. Rate-Optimizing Compilers for Continuously Non-Malleable Codes. Cryptology ePrint Archive, Report 2019/055, 2019. URL: https://eprint.iacr.org/2019/055.

36 Jean-Sébastien Coron. Security Proof for Partial-Domain Hash Signature Schemes. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 613–626. Springer, Heidelberg, August 2002. doi:10.1007/3-540-45708-9_39.

37 Dana Dachman-Soled and Mukul Kulkarni. Upper and Lower Bounds for Continuous Non-Malleable Codes. In Dongdai Lin and Kazue Sako, editors, *PKC 2019, Part I*, volume 11442 of *LNCS*, pages 519–548. Springer, Heidelberg, April 2019. doi:10.1007/978-3-030-17253-4_18.

38 Dana Dachman-Soled, Mukul Kulkarni, and Aria Shahverdi. Tight Upper and Lower Bounds for Leakage-Resilient, Locally Decodable and Updatable Non-malleable Codes. In Serge Fehr, editor, *PKC 2017, Part I*, volume 10174 of *LNCS*, pages 310–332. Springer, Heidelberg, March 2017. doi:10.1007/978-3-662-54365-8_13.

39 Dana Dachman-Soled, Mukul Kulkarni, and Aria Shahverdi. Local Non-malleable Codes in the Bounded Retrieval Model. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018, Part II*, volume 10770 of *LNCS*, pages 281–311. Springer, Heidelberg, March 2018. doi:10.1007/978-3-319-76581-5_10.

40 Dana Dachman-Soled, Feng-Hao Liu, Elaine Shi, and Hong-Sheng Zhou. Locally Decodable and Updatable Non-malleable Codes and Their Applications. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part I*, volume 9014 of *LNCS*, pages 427–450. Springer, Heidelberg, March 2015. doi:10.1007/978-3-662-46494-6_18.

41 Akshay Degwekar, Vinod Vaikuntanathan, and Prashant Nalini Vasudevan. Fine-Grained Cryptography. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part III*, volume 9816 of *LNCS*, pages 533–562. Springer, Heidelberg, August 2016. doi:10.1007/978-3-662-53015-3_19.

42 Stefan Dziembowski, Tomasz Kazana, and Maciej Obremski. Non-malleable Codes from Two-Source Extractors. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 239–257. Springer, Heidelberg, August 2013. doi:10.1007/978-3-642-40084-1_14.

43 Stefan Dziembowski, Krzysztof Pietrzak, and Daniel Wichs. Non-Malleable Codes. *J. ACM*, 65(4):20:1–20:32, April 2018. Extended abstract appeared in Innovations in Computer Science (ICS) 2010. doi:10.1145/3178432.

44 Antonio Faonio, Jesper Buus Nielsen, Mark Simkin, and Daniele Venturi. Continuously Non-malleable Codes with Split-State Refresh. In Bart Preneel and Frederik Vercauteren, editors, *ACNS 18*, volume 10892 of *LNCS*, pages 121–139. Springer, Heidelberg, July 2018. doi:10.1007/978-3-319-93387-0_7.

45 Sebastian Faust, Kristina Hostáková, Pratyay Mukherjee, and Daniele Venturi. Non-Malleable Codes for Space-Bounded Tampering. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part II*, volume 10402 of *LNCS*, pages 95–126. Springer, Heidelberg, August 2017. doi:10.1007/978-3-319-63715-0_4.

46 Sebastian Faust, Pratyay Mukherjee, Jesper Buus Nielsen, and Daniele Venturi. Continuous Non-malleable Codes. In Yehuda Lindell, editor, *TCC 2014*, volume 8349 of *LNCS*, pages 465–488. Springer, Heidelberg, February 2014. doi:10.1007/978-3-642-54242-8_20.

47 Sebastian Faust, Pratyay Mukherjee, Jesper Buus Nielsen, and Daniele Venturi. A Tamper and Leakage Resilient von Neumann Architecture. In Jonathan Katz, editor, *PKC 2015*, volume 9020 of *LNCS*, pages 579–603. Springer, Heidelberg, March/April 2015. doi:10.1007/978-3-662-46447-2_26.

48 Sebastian Faust, Pratyay Mukherjee, Daniele Venturi, and Daniel Wichs. Efficient Non-malleable Codes and Key-Derivation for Poly-size Tampering Circuits. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 111–128. Springer, Heidelberg, May 2014. doi:10.1007/978-3-642-55220-5_7.

49 Marc Fischlin and Dominique Schröder. On the Impossibility of Three-Move Blind Signature Schemes. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 197–215. Springer, Heidelberg, May/June 2010. doi:10.1007/978-3-642-13190-5_10.

50 Georg Fuchsbauer, Momchil Konstantinov, Krzysztof Pietrzak, and Vanishree Rao. Adaptive Security of Constrained PRFs. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part II*, volume 8874 of *LNCS*, pages 82–101. Springer, Heidelberg, December 2014. doi:10.1007/978-3-662-45608-8_5.

51 Sanjam Garg, Raghav Bhaskar, and Satyanarayana V. Lokam. Improved Bounds on Security Reductions for Discrete Log Based Signatures. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 93–107. Springer, Heidelberg, August 2008. doi:10.1007/978-3-540-85174-5_6.

52 Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In Lance Fortnow and Salil P. Vadhan, editors, *43rd ACM STOC*, pages 99–108. ACM Press, June 2011. doi:10.1145/1993636.1993651.

53 Yael Gertner, Sampath Kannan, Tal Malkin, Omer Reingold, and Mahesh Viswanathan. The Relationship between Public Key Encryption and Oblivious Transfer. In *41st FOCS*, pages 325–335. IEEE Computer Society Press, November 2000. doi:10.1109/SFCS.2000.892121.

54 Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to Construct Random Functions (Extended Abstract). In *25th FOCS*, pages 464–479. IEEE Computer Society Press, October 1984. doi:10.1109/SFCS.1984.715949.

55 Russell Impagliazzo. Hard-Core Distributions for Somewhat Hard Problems. In *36th FOCS*, pages 538–545. IEEE Computer Society Press, October 1995. doi:10.1109/SFCS.1995.492584.

56 Russell Impagliazzo and Steven Rudich. Limits on the Provable Consequences of One-Way Permutations. In *21st ACM STOC*, pages 44–61. ACM Press, May 1989. doi:10.1145/73007.73012.

57 Zahra Jafargholi and Daniel Wichs. Tamper Detection and Continuous Non-malleable Codes. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part I*, volume 9014 of *LNCS*, pages 451–480. Springer, Heidelberg, March 2015. doi:10.1007/978-3-662-46494-6_19.

58 Bhavana Kanukurthi, Sai Lakshmi Bhavana Obbattu, and Sruthi Sekar. Four-State Non-malleable Codes with Explicit Constant Rate. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part II*, volume 10678 of *LNCS*, pages 344–375. Springer, Heidelberg, November 2017. doi:10.1007/978-3-319-70503-3_11.

59 Bhavana Kanukurthi, Sai Lakshmi Bhavana Obbattu, and Sruthi Sekar. Non-malleable Randomness Encoders and Their Applications. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part III*, volume 10822 of *LNCS*, pages 589–617. Springer, Heidelberg, April/May 2018. doi:10.1007/978-3-319-78372-7_19.

60 Aggelos Kiayias, Feng-Hao Liu, and Yiannis Tselekounis. Practical Non-Malleable Codes from l-more Extractable Hash Functions. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016*, pages 1317–1328. ACM Press, October 2016. doi:10.1145/2976749.2978352.

61 Aggelos Kiayias, Feng-Hao Liu, and Yiannis Tselekounis. Non-Malleable Codes for Partial Functions with Manipulation Detection. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part III*, volume 10993 of *LNCS*, pages 577–607. Springer, Heidelberg, August 2018. doi:10.1007/978-3-319-96878-0_20.

62 Xin Li. Improved non-malleable extractors, non-malleable codes and independent source extractors. In Hamed Hatami, Pierre McKenzie, and Valerie King, editors, *49th ACM STOC*, pages 1144–1156. ACM Press, June 2017. doi:10.1145/3055399.3055486.

63 Xin Li. Non-Malleable Extractors and Non-Malleable Codes: Partially Optimal Constructions. Cryptology ePrint Archive, Report 2018/353, 2018. URL: https://eprint.iacr.org/2018/353.

64 Feng-Hao Liu and Anna Lysyanskaya. Tamper and Leakage Resilience in the Split-State Model. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 517–532. Springer, Heidelberg, August 2012. doi:10.1007/978-3-642-32009-5_30.

65 Rafail Ostrovsky, Giuseppe Persiano, Daniele Venturi, and Ivan Visconti. Continuously Non-Malleable Codes in the Split-State Model from Minimal Assumptions. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part III*, volume 10993 of *LNCS*, pages 608–639. Springer, Heidelberg, August 2018. doi:10.1007/978-3-319-96878-0_21.

66 Pascal Paillier and Damien Vergnaud. Discrete-Log-Based Signatures May Not Be Equivalent to Discrete Log. In Bimal K. Roy, editor, *ASIACRYPT 2005*, volume 3788 of *LNCS*, pages 1–20. Springer, Heidelberg, December 2005. doi:10.1007/11593447_1.

67 Rafael Pass. Limits of provable security from standard assumptions. In Lance Fortnow and Salil P. Vadhan, editors, *43rd ACM STOC*, pages 109–118. ACM Press, June 2011. doi:10.1145/1993636.1993652.

68 Peter M. R. Rasmussen and Amit Sahai. Expander Graphs are Non-Malleable Codes. Cryptology ePrint Archive, Report 2018/929, 2018. URL: https://eprint.iacr.org/2018/929.

69 Omer Reingold, Luca Trevisan, and Salil P. Vadhan. Notions of Reducibility between Cryptographic Primitives. In Moni Naor, editor, *TCC 2004*, volume 2951 of *LNCS*, pages 1–20. Springer, Heidelberg, February 2004. doi:10.1007/978-3-540-24638-1_1.

70 Yannick Seurin. On the Exact Security of Schnorr-Type Signatures in the Random Oracle Model. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 554–571. Springer, Heidelberg, April 2012. doi:10.1007/978-3-642-29011-4_33.

71 Daniel R. Simon. Finding Collisions on a One-Way Street: Can Secure Hash Functions Be Based on General Assumptions? In Kaisa Nyberg, editor, *EUROCRYPT'98*, volume 1403 of *LNCS*, pages 334–345. Springer, Heidelberg, May/June 1998. doi:10.1007/BFb0054137.

Cryptography from Information Loss

Marshall Ball
Columbia University, New York, NY, USA
marshall@cs.columbia.edu

Elette Boyle
IDC Herzliya, Israel
eboyle@alum.mit.edu

Akshay Degwekar
MIT, Cambridge, MA, USA
degwekarakshay@gmail.com

Apoorvaa Deshpande
Brown University, Providence, RI, USA
apoorvaa_deshpande@brown.edu

Alon Rosen
IDC Herzliya, Israel
alon.rosen@idc.ac.il

Vinod Vaikuntanathan
MIT, Cambridge, MA, USA
vinodv@mit.edu

Prashant Nalini Vasudevan
UC Berkeley, CA, USA
prashvas@berkeley.edu

Abstract

Reductions between problems, the mainstay of theoretical computer science, efficiently map an instance of one problem to an instance of another in such a way that solving the latter allows solving the former.[1] The subject of this work is "lossy" reductions, where the reduction loses some information about the input instance. We show that such reductions, when they exist, have interesting and powerful consequences for lifting hardness into "useful" hardness, namely cryptography.

Our first, conceptual, contribution is a definition of lossy reductions in the language of mutual information. Roughly speaking, our definition says that a reduction C is t-lossy if, for any distribution X over its inputs, the mutual information $I(X;\mathsf{C}(X)) \leq t$. Our treatment generalizes a variety of seemingly related but distinct notions such as worst-case to average-case reductions, randomized encodings (Ishai and Kushilevitz, FOCS 2000), homomorphic computations (Gentry, STOC 2009), and instance compression (Harnik and Naor, FOCS 2006).

We then proceed to show several consequences of lossy reductions:

1. We say that a language L has an f-reduction to a language L' for a Boolean function f if there is a (randomized) polynomial-time algorithm C that takes an m-tuple of strings $X = (x_1, \ldots, x_m)$, with each $x_i \in \{0,1\}^n$, and outputs a string z such that with high probability,

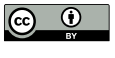
   $$L'(z) = f(L(x_1), L(x_2), \ldots, L(x_m))$$

   Suppose a language L has an f-reduction C to L' that is t-lossy. Our first result is that one-way functions exist if L is worst-case hard and one of the following conditions holds:
   - f is the OR function, $t \leq m/100$, and L' is the same as L
   - f is the Majority function, and $t \leq m/100$
   - f is the OR function, $t \leq O(m \log n)$, and the reduction has no error

   This improves on the implications that follow from combining (Drucker, FOCS 2012) with (Ostrovsky and Wigderson, ISTCS 1993) that result in *auxiliary-input* one-way functions.
2. Our second result is about the stronger notion of t-compressing f-reductions – reductions that only output t bits. We show that if there is an average-case hard language L that has a t-compressing Majority reduction to some language for $t = m/100$, then there exist collision-resistant hash functions.

   This improves on the result of (Harnik and Naor, STOC 2006), whose starting point is a cryptographic primitive (namely, one-way functions) rather than average-case hardness, and whose assumption is a compressing OR-reduction of SAT (which is now known to be false unless the polynomial hierarchy collapses).

[1] Such reductions are called many-one or Karp reductions. To be sure, there are more general types of reductions, such as oracle reductions (or Cook reductions), which we do not deal with in this paper.



11th Innovations in Theoretical Computer Science Conference (ITCS 2020).
Editor: Thomas Vidick; Article No. 81; pp. 81:1–81:27

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Along the way, we define a non-standard one-sided notion of average-case hardness, which is the notion of hardness used in the second result above, that may be of independent interest.

2012 ACM Subject Classification Theory of computation → Computational complexity and cryptography; Theory of computation → Problems, reductions and completeness; Theory of computation → Cryptographic primitives

Keywords and phrases Compression, Information Loss, One-Way Functions, Reductions, Generic Constructions

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.81

Funding Marshall Ball is supported by an IBM Research PhD Fellowship. This research is based upon work supported in part by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA) via Contract No. 2019-1902070006. Elette Boyle is supported in part by ISF grant 1861/16 and AFOSR Award FA9550-17-1-0069. Akshay Degwekar is supported in part by ISF grant 1861/16, AFOSR Award FA9550-17-1-0069, NSF Grants CNS-1413920 and CNS-1350619, and by the Defense Advanced Research Projects Agency (DARPA) and the U.S. Army Research Office under contracts W911NF-15-C-0226 and W911NF-15-C-0236. Alon Rosen is supported by ISF grant No. 1399/17 and via Project PROMETHEUS (Grant 780701). Vinod Vaikuntanathan is supported in part by NSF Grants CNS-1350619, CNS-1718161 and CNS-1414119, an MIT-IBM grant, a Microsoft Faculty Fellowship and a DARPA Young Faculty Award. Prashant Vasudevan is supported in part from AFOSR Award FA9550-19-1-0200, AFOSR YIP Award, NSF CNS Award 1936826, DARPA and SPAWAR under contract N66001-15-C-4065, a Hellman Award and research grants by the Okawa Foundation, Visa Inc., and Center for Long-Term Cybersecurity (CLTC, UC Berkeley). This work was done when Akshay Degwekar was a student at MIT, and in part while Marshall Ball, Akshay Degwekar, Apoorvaa Deshpande, and Prashant Vasudevan were visiting the FACT Center in IDC Herzliya. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either express or implied, of ODNI, IARPA, the U.S. Government, or other funding agencies. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein.

1 Introduction

Consider a polynomial-time reduction R from a language L to another L'.[2] That is, R takes as input $x \in \{0,1\}^n$ for any n and produces an output $y \in \{0,1\}^{m(n)}$ for some polynomial m such that $x \in L$ if and only if $y \in L'$. Such a reduction relates the computational complexities of L and L'; its existence says that the complexity of L is at most the complexity of L', upto some additional polynomial running time. In general, the reduction says little more than this. If the reduction has certain additional properties, however, it tells us more about the complexity of L and starts becoming more useful. This work is about an important such class of reductions, namely *lossy reductions* that forget information about the input instance.

Losing Information through Compression. One way to lose information about the input instance is by compressing it. Continuing the discussion above, if $m(n) = O(\log n)$, the reduction enables L to be decided in non-uniform polynomial-time. More generally, L can be decided on n-bit inputs by a circuit of size roughly $2^{m(n)}$. Such *instance compressing*

[2] Much of our discussion also applies to promise problems and search problems.

reductions, those that have $m(n) \ll n$, have been an important part of the study of fixed parameter tractable algorithms, where they are called kernelizations (see [11] and the references therein). A classical example is the vertex cover problem on n-node graphs parameterized by the size of the cover k, where there is a $\mathsf{poly}(n)$-time algorithm that takes the graph as input and outputs a smaller graph of size $\mathsf{poly}(k)$ which has a cover of size k if and only if the original graph does.

Harnik and Naor [18] showed a variety of cryptographic applications of such compressing reductions. For example, assuming that the Boolean Satisfiability problem (SAT) has good enough instance compression, they showed how to construct a collision-resistant hash function starting from any one-way function, a task known to be impossible to do in a black-box manner [28]. Furthermore, if this compression has some additional properties, they also show how to get public-key encryption schemes and oblivious transfer protocols from one-way functions. However, Fortnow and Santhanam [14] and later Drucker [12] showed that, unless the polynomial hierarchy collapses, such instance compression algorithms for SAT cannot exist.

This Work: Losing Information without Compression. Compression is one way to lose information, but not the only one. In this paper, we study cryptographic implications of *lossy reductions* – randomized algorithms that forget information about the input instance but not necessarily by compressing them. Such reductions are abundant in the study of average-case complexity and cryptography.

An example is a reduction $\mathsf{R}(x)$ that outputs a sample from a distribution that is (almost) completely determined by whether x is contained in L (and, as before, $\mathsf{R}(x) \in L'$ if and only if $x \in L$). Such a reduction loses all information about x other than its membership in L. Thus, it relates the complexity of deciding L in the *worst-case* to the complexity of deciding L' *on average* over a specific distribution[3]. In other words, a reduction that loses all information about its input except membership in L is a *worst-case to average-case reduction* from L to L'.

From a different vantage point, such reductions are equivalent to *randomized encodings* [19], a powerful and versatile notion in cryptography (see the survey by Applebaum [1] and the references therein). In particular, randomized encodings have been used to delegate computations [3], for secure computation [31, 7, 20], for parallel cryptography [2], for key-dependent message security and much more.

Given this relevance to cryptography of reductions that lose all information except membership, both as average-case reductions and as randomized encodings, we ask whether anything similar can be said of reductions that are almost this way. That is, *are reductions that lose* almost *all other information except membership useful for cryptography*?

Lossy Reductions and Cryptography. Our first contribution is a definition of such lossy reductions through the lens of mutual information. For a function $t : \mathbb{N} \to \mathbb{R}^+$, we say that a reduction C from a language L to a language L' is *t-lossy* if for any random variable X over bit-strings of length n (capturing an input distribution), the mutual information $I(X; \mathsf{C}(X))$ is at most $t(n)$. Note that this definition does not fix an input distribution, rather quantifies over all of them. In this sense, it is reminiscent of the prior-free information complexity definition of [10]. Roughly speaking, if X has little entropy, we don't care; if X has more than $t(n)$ bits of entropy, we want the reduction to reveal at most $t(n)$ of it.

[3] The distribution is given by picking any $x_N \notin L$ and any $x_Y \in L$, and taking the equal convex combination of the distributions sampled by $\mathsf{R}(x_N)$ and $\mathsf{R}(x_Y)$.

Whereas Harnik and Naor showed that instance compression can be used to construct other cryptographic primitives *from* one-way functions, we investigate the possibility of using compression or information loss to construct one-way functions from simpler hardness properties. Some results along these lines are already known, or are implicit in prior work: (1) Drucker [12] showed that the existence of certain kinds of compression for a language L implies that L has a statistical zero-knowledge proof.[4] Together with a result of Ostrovsky [25], this implies that if such a language is *average-case hard*, then one-way functions exist; (2) Replacing average-case with worst-case hardness in the above gives us *auxiliary input* one-way functions, a weaker object [26]; (3) on the other hand, if the reduction is a randomized encoding, then replacing average-case with worst-case hardness in the above gives us one-way functions [5].

We demonstrate a number of other similar sufficient conditions to elevate worst-case hardness to one-way functions. To give the reader a taste of what is to come, one of our results is a way to achieve the best of (1), (2) and (3) above, showing how to elevate worst-case hardness of L into one-way functions if L has a lossy reduction to L'.

▶ **Informal Theorem 1.1.** *Suppose there is a perfect reduction from L to L' that is $O(\log n)$-lossy on n-bit inputs. If L is worst-case hard, then One-Way Functions exist.*

The above follows as a corollary to Informal Theorem 1.4 described later. As noted earlier, randomized encodings may be seen roughly as a 1-lossy reduction from L to some problem L', and the worst-case hardness of a problem that has randomized encodings implies the existence of a one-way function. The above theorem says that it is in fact sufficient for the reduction to be $O(\log n)$-lossy (rather than 1-lossy) for this conclusion to hold, if it has the additional property that it does not make any errors – that is, a YES instance of L is never mapped to a NO instance of L', and also the other way round. This may also be interpreted as saying that the implication to OWFs for randomized encodings still holds if the privacy guarantee of the encodings is much weaker, as long as the correctness is perfect.

f-Reductions and OWFs. Our main theorems are about the implications of lossy reductions for the composition of a simple Boolean function with membership in a language. Let $f = \{f^n\}$ denote a family of (partial) Boolean functions, where for some polynomial m, the function f^n takes $m(n)$ bits as input. We denote by $f \circ L$ the composition of f with L – on $m(n)$ inputs $x_1, \ldots, x_{m(n)} \in \{0,1\}^n$, this function is computed as $f(L(x_1), \ldots, L(x_{m(n)}))$. We refer to a reduction from $f \circ L$ to some problem as an f-reduction of L.

Such f-reductions that are also compressing (in terms of bit length) have been the subject of considerable past work in the study of parametrized complexity [18, 9, 14, 13] (see also further references in [13]), especially for simple functions f like AND and OR. Most relevant to our work are the results of Drucker [13], who showed that if there is a sufficiently compressing reduction from $\mathrm{OR}_m \circ L$ to any problem L', then L is contained in SZK (where m is some polynomial and OR_m is the family of OR function on $m(n)$ inputs). As noted earlier, this implied membership in SZK lets us lift the average-case hardness of L to a one-way function, or its worst-case hardness to an auxiliary-input OWF.

Our starting point is the observation that Drucker's proofs still work if the reduction were just lossy and not necessarily compressing. We prove the following theorems that show three different sufficient conditions for lossy reductions to be useful in lifting worst-case

[4] Drucker's results were stated for the stronger notion of compression in terms of bit-length, but his proofs imply the same results for notion of lossy reductions as well.

hardness directly to one-way functions. Each of the three imply OWFs along different paths, as described briefly below. These paths and the pointers to the relevant parts of the paper are presented in Figure 1.

▶ **Informal Theorem 1.2.** *Suppose, for some polynomial m, and promise problem L, there is a reduction from $\mathrm{OR}_m \circ L$ to L that is $(m(n)/100)$-lossy on $(n \cdot m(n))$-bit inputs. If L is worst-case hard, then One-Way Functions exist.*

We prove this by first showing that Drucker's techniques actually imply a certain kind of worst-case to average-case reduction to L'. And if L is worst-case hard, then this reduction lets us conclude that L' is *one-sided average-case hard*, a notion explained later in this section. Thus, if L' is L itself, then it is both contained in SZK and is one-sided average-case hard. Finally, we show that the existence of any such problem implies the existence of a OWF.

The next theorem we prove is that a similarly lossy Majority-reduction for L by itself lets us lift the worst-case hardness of L to a OWF, without worrying about what it reduces to.

▶ **Informal Theorem 1.3.** *Suppose, for some polynomial m, and promise problem L, there is a reduction from $\mathrm{MAJ}_m \circ L$ to some problem that is $(m(n)/100)$-lossy on $(n \cdot m(n))$-bit inputs. If L is also worst-case hard, then One-Way Functions exist.*

In this case, we extend Drucker's techniques to show that such a reduction from $\mathrm{MAJ}_m \circ L$ implies a *two-sided* average-case reduction from L to some problem L' – where the YES and NO parts of the distribution over instances of L' that L is reduced to can be sampled separately. In different terms, this implies that L has statistical randomized encodings, and the conclusion follows from the fact that the worst-case hardness of a problem that has randomized encodings implies one-way functions [5].

Finally, the following theorem says that we can make do with OR_m-reductions that are not as lossy as before – only $O(m(n)\log n)$-lossy instead of $m(n)/100$ – if the reduction is perfect, meaning that it never maps YES instances of $\mathrm{OR}_m \circ L$ to NO instances of L', and also the other way. Note that a reduction that is $\Omega(m(n)\log n)$-lossy could still preserve some information about all of its $m(n)$ inputs – for instance, whether each x_i is a YES or NO instance of L, etc..

▶ **Informal Theorem 1.4.** *Suppose, for some polynomial m, and promise problems L and L', there is a* perfect *reduction from $\mathrm{OR}_m \circ L$ to L' that is $O(m(n)\log n)$-lossy on $(n \cdot m(n))$-bit inputs. If L is worst-case hard, then One-Way Functions exist.*

We prove this by showing that a perfect reduction from $\mathrm{OR}_m \circ L$ implies a one-sided average-case reduction that is perfect in the sense that the NO and YES distributions that are generated by the reduction are disjoint. On the other hand, the hardness that is implied by the reduction is quite weak due to it not being lossy enough – it only says that the NO distribution is no better than $(1 - 1/\mathsf{poly}(n))$-distinguishable from the YES distribution for a given algorithm. However, the perfectness lets us show that the sampler for the NO distribution itself is a weak one-way function (with the input as its random string).

Collision-Resistance from Compression. Finally, we show that compressing reductions – a strong form of lossy reductions where the output length is smaller than the input length – can be used to lift one-sided average-case hardness to collision-resistant hash functions.

▶ **Informal Theorem 1.5.** *Suppose, for some polynomial m, and promise problem L, there is a perfect reduction from $\mathrm{OR}_m \circ L$ to some problem that compresses $(n \cdot m(n))$-bit inputs to $m(n)/100$ bits. If L is also one-sided average-case hard with perfect sampling*[5]*, then Collision-Resistant Hash Functions exist.*

Our construction builds directly on the construction of collision-resistant hash functions from homomorphic commitments by Ishai et al [21]. In our construction, the keys of the hash function for hashing strings of length m correspond to a set of $2m$ instances (grouped into m pairs) sampled from the NO distribution. Each bit of an input x is used to select an instance from the corresponding pair, and the hash function is computed by running the compressing OR-reduction on this set.

The construction in [21] may be seen to use essentially the same approach, where instead of using a compressing OR-reduction for a hard problem, they use the homomorphism of a commitment scheme. The security of the commitment scheme there is the analogue of the one-sided average-case hardness of L, and our observation is that, while homomorphism is one way to compress (XOR-compression in their case), it is not necessary, and any compressing OR-reduction is sufficient to use with the construction.

One-Sided Average-Case Hardness. In the process of studying the implications of OR-compression of a problem to itself, we introduce the notion of one-sided average-case hardness, which we describe next. Recall that the worst-case hardness of a problem L says that for any polynomial-time algorithm A that attempts to decide L, there exists some input x on which it is wrong. Perhaps the simplest notion of average-case hardness flips the quantifiers here and says that there exists a (samplable) distribution over inputs such that any algorithm fails to decide L with large advantage when inputs are sampled from this distribution.

A stronger notion of average-case hardness that we call *two-sided* average-case hardness (and one that directly relates to one-way functions), says that there are two (samplable) distributions, one over the just the YES instances of L and another over just the NO instances, such that no algorithm A can distinguish between them. One-sided average-case hardness is a notion between this and worst-case hardness, and says that there exists one samplable distribution N over (mostly) NO instances such that for any algorithm A, there exists a distribution Y_A over (mostly) YES instances such that A cannot distinguish between N and Y_A.

One-sided average-case hardness implies worst-case hardness, is implied by two-sided average-case hardness, and does not seem to be related in this manner to plain average-case hardness. Ostrovsky [25] proved that plain average-case hardness of a problem in SZK implies OWFs. We prove the following theorem that, to our knowledge, is incomparable to it.

▶ **Informal Theorem 1.6.** *If there is a problem in SZK that is one-sided average-case hard, then One-Way Functions exist.*

The theorem is proven using a reduction to (the complement of) the Statistical Difference problem that is complete for SZK [27]. We also show that the existence of an analogous notion of worst-case to average-case reduction – called one-sided average-case reduction – from a problem L to any other problem implies that L is contained in SZK. Thus, along with the worst-case hardness of L, they imply auxiliary-input OWFs analogous to how two-sided average-case hardness implies OWFs. Such reductions are closely related to the notion of semi-private randomized encodings [4].

[5] The perfect sampling condition implies that the YES and NO distributions that come up in the definition of the hardness are contained completely within the YES and NO parts of the problem.

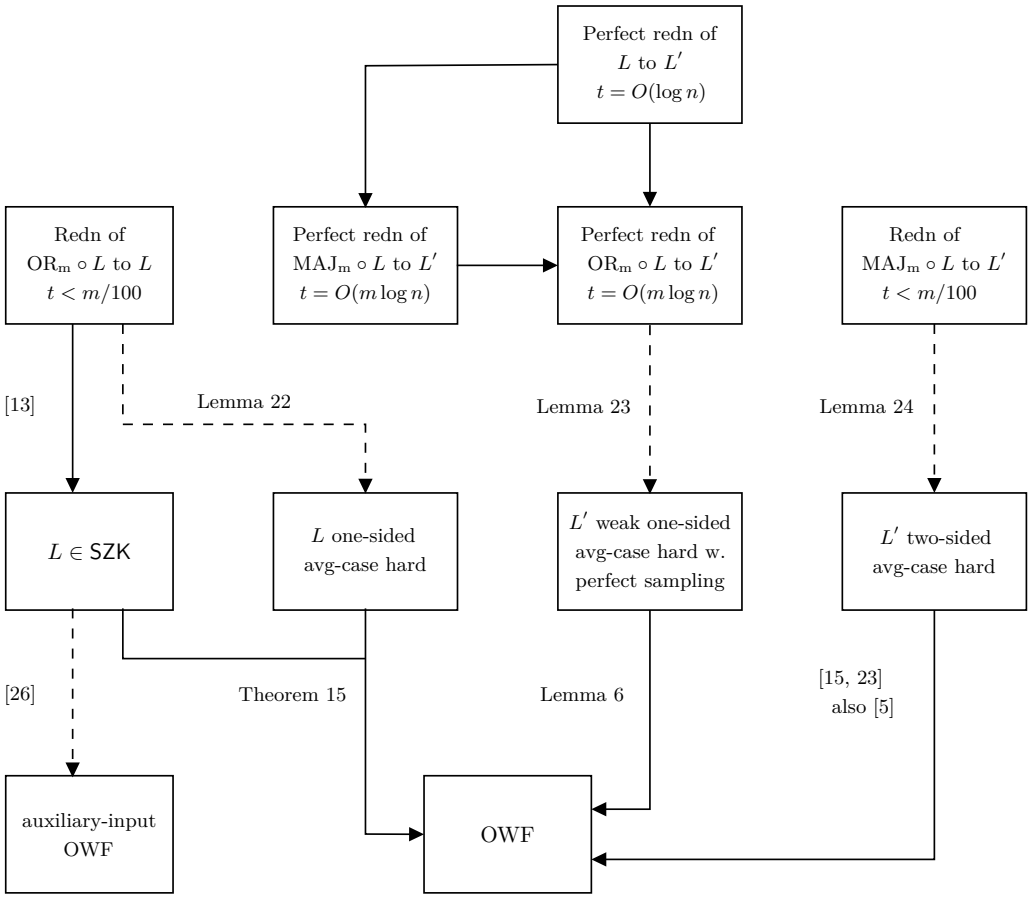


Figure 1 The many paths to One-Way Functions. L and L' are promise problems. The solid arrows denote implications, and the dashed arrows denote implication conditioned on the worst-case hardness of L. In all cases, m and t are polynomials in n, and the reductions lose all but $t(n)$ bits of information on a set of $m(n)$ inputs of size n – that is, t-lossy according to Definition 21 (when reducing L to L', $m(n)$ is set to 1).

1.1 Notation

We will always deal with promise problems and partial functions rather than languages and total functions, as these come up naturally in our discussions, and are also more general. Given a promise problem L, we denote by L_Y and L_N the corresponding sets of YES and NO inputs. Abusing notation, we also denote by L the partial function indicating membership in this promise problem. In the other direction, note that any partial Boolean function $f : \{0,1,\perp\}^n \to \{0,1,\perp\}$ has a corresponding promise problem, which we also sometimes denote by f. The symbol $\perp$ is set to be the output of a partial function on an input on which it is otherwise undefined. In general, we take U_S to be the random variable distributed uniformly over the set S, where S is clear from context we simply write U.

Random variables are capitalized (X), and algorithms are sans serif (A). $\mathsf{A}(X)$ denotes the random variable resulting from sampling x according to X and running A on it. $\mathrm{H}(X)$ is the Shannon entropy of X, and $\mathrm{I}(X;Y)$ is the mutual information between X and Y. We take $\Delta(X;Y)$ to denote the total variation distance between X and Y.

Unless otherwise specified, all algorithms in our work are non-uniform, and assumptions and claims of hardness are also against non-uniform algorithms. Further, we always use notions of hardness that are strong in the following sense – when we say, for instance, that a problem L is hard against polynomial-time algorithms, we mean that for any polynomial-time algorithm A, there is an $n_{\mathsf{A}} \in \mathbb{Z}$ such that for any $n \geq n_{\mathsf{A}}$, the algorithm A fails to decide L correctly on some instance of size n. This is to be contrasted against the more standard notion of hardness, infinitely-often hardness, which would say that there is an infinite sequence of n's such that A fails to decide L on some instance of size n. All of our lemmas and theorems, however, may also be stated in terms of infinitely-often hardness and follow from the same proofs, resulting in infinitely-often cryptographic objects as well.

1.2 Outline of the Paper

In Section 2, we define the various kinds of computational hardness that we will be using, the corresponding reductions, and some lemmas relating the reductions, the hardness, and one-way functions. In Section 3, we prove define the class SZK and prove that one-sided average-case hardness in SZK implies one-way functions, and some associated lemmas. In Section 4, we define our notion of lossy reductions, and prove that certain kinds of lossy reductions along with worst-case hardness imply one-way functions. In Section 5, we prove that one-sided average-case hardness and OR-compressing reductions imply collision-resistance hash functions.

2 Computational Hardness

In this section, we define the various kinds of computational hardness of promise problems that we will be employing in later discussions, and also different kinds of reductions between problems that have implications for such hardness. The proofs of the lemmas in this section are in Appendix A. We start by defining the weakest and simplest form of hardness.

▶ **Definition 1** (Worst-Case Hardness). *A problem L is* worst-case hard *if, for any polynomial-time algorithm A and all large enough n, there is an input $x \in (L_Y \cup L_N) \cap \{0,1\}^n$ such that $\Pr[\mathsf{A}(x) = L(x)] < 2/3$.*

Worst-case hardness says that for any algorithm A (and large enough instance size n), there is an input x on which it is wrong. We next define perhaps the simplest form of average-case hardness, which mostly just swaps these quantifiers. It says that there is a distribution over inputs x such that any algorithm A is wrong on average over this distribution.

▶ **Definition 2** (Average-Case Hardness). *A problem L is* average-case hard *if there is a polynomial-time sampling algorithm D such that:*

- *D mostly samples instances from L_Y and L_N. That is, for all large enough n,*

$$\Pr_{x \leftarrow \mathsf{D}(1^n)}[x \notin L_Y \cup L_N] \leq 0.1$$

- *The membership in L of most instances drawn from D is hard to decide. That is, for any polynomial-time algorithm A, for all large enough n,*

$$\Pr_{x \leftarrow \mathsf{D}(1^n)}[\mathsf{A}(x) = L(x) \vee L(x) = \perp] \leq \frac{1}{2} + 0.15$$

Note that the constant 2/3 in Definition 1 is somewhat arbitrary – if there is an algorithm A that does in fact decide L in the worst-case with success probability bounded away from 1/2 for all n, this can be amplified by repetition to get an algorithm A' with success probability close to 1.

In Definition 2, however, the constants are not arbitrary, at least in this straighforward sense. There are known hardness amplification theorems (see [6, Chapter 19]) that can take a problem L that is somewhat average-case hard as above, and obtain another problem L' and a distribution under which L' is much harder, but these do not say that L itself is any harder than initially supposed. Our choice of the constants we use in our definitions are such that the hardness is mild enough to be implied by the hypotheses that we later start with, and are yet strong enough to have interesting implications for cryptography, as shown by our theorems.

We next define a stronger and natural notion of average-case hardness that is known to be closely related to One-Way Functions. It separates the hard distribution into YES and NO parts, and asks that each of these parts be efficiently samplable.

▶ **Definition 3** (Two-Sided Average-Case Hardness). *A problem L is* two-sided average-case hard *if there are two polynomial-time sampling algorithms Y and N such that:*

- *Y and N mostly sample instances from L_Y and L_N, respectively. That is, for all large enough n,*

$$\Pr_{x \leftarrow \mathsf{Y}(1^n)}[x \notin L_Y] \leq 0.1$$
$$\Pr_{x \leftarrow \mathsf{N}(1^n)}[x \notin L_N] \leq 0.1$$

- *The outputs of Y and N are computationally indistinguishable. That is, for any polynomial-time algorithm A, for all large enough n,*

$$\left| \Pr_{x \leftarrow \mathsf{Y}(1^n)}[\mathsf{A}(x) = 1] - \Pr_{x \leftarrow \mathsf{N}(1^n)}[\mathsf{A}(x) = 1] \right| \leq 0.3$$

It is known that any two-sided average-case hard problem implies the existence of a One-Way Function. This follows from the fact that such hardness implies the existence of a family of statistically far distributions that are computationally indistinguishable (N and Y). Goldreich [15] showed that this is equivalent to the existence of OWFs if the indistinguishability was with negligible advantage, and this was later extended by Naor and Rothblum [23] to hold for weaker indistinguishability that covers the constants used above (see [8] for an alternative proof), leading to the following lemma. A theorem that is roughly equivalent to the combination of Lemmas 4 and 9 was also proven by Applebaum and Raykov [5].

▶ **Lemma 4.** *There is a problem that is two-sided average-case hard if and only if One-Way Functions exist.*

Further, note that the existence of a OWF also implies (as can be seen by the PRG that can be constructed from it) that there is some language that is two-sided average-case hard, but with much stronger guarantees – with negligible functions (in n) in place of the constants 0.1 and 0.3 in Definition 3. In this sense, for the purposes of its relevance to cryptography, the constants in the definition above are also somewhat arbitrary.

Finally, we introduce a notion of average-case hardness that is intermediate between worst-case and two-sided average-case hardness and, to our knowledge, is incomparable to plain average-case hardness. Recall that average-case hardness was obtained by swapping

the quantifiers in worst-case hardness, and two-sided average-case hardness then came out of separating and fixing the YES and NO parts. The following definition fixes just the NO distribution, and requires that for any algorithm A, there exists some YES distribution that it cannot distinguish from this fixed NO distribution.

▶ **Definition 5** (One-Sided Average-Case Hardness). *A problem L is* one-sided average-case hard *if there is a polynomial-time sampling algorithm N such that:*

- N *mostly samples instances from L_N. That is, for all large enough n,*

$$\Pr_{x \leftarrow \mathsf{N}(1^n)} [x \notin L_N] \leq 0.1$$

- *For any polynomial-time algorithm, there is some distribution that is mostly over L_Y that it cannot distinguish from the output of N. That is, for every polynomial-time algorithm A, there is a (possibly inefficient) sampler Y_A such that:*
  - Y_A *mostly samples instances from L_Y. That is, for all large enough n,*

$$\Pr_{x \leftarrow \mathsf{Y}_\mathsf{A}(1^n)} [x \notin L_Y] \leq 0.1$$

  - *The outputs of N and Y_A are indistinguishable to A. That is, for all large enough n,*

$$\left| \Pr_{x \leftarrow \mathsf{N}(1^n)} [\mathsf{A}(x) = 1] - \Pr_{x \leftarrow \mathsf{Y}_\mathsf{A}(1^n)} [\mathsf{A}(x) = 1] \right| \leq 0.3$$

We say that the hardness is strong *if the distinguishing advantage is negligible in n, and* weak *if it is only required to be less than $(1 - 1/n^c)$ for some constant c. We say that L is one-sided average-case hard with* perfect sampling *if N and Y_A only output samples in L_N and L_Y, respectively.*

This kind of hardness turns out to be somewhat related to a weaker kind of OWF called auxiliary-input OWF – see Corollary 17. We do not know how to amplify this kind of hardness, so the choice of constants in its definition is not arbitrary.

We consider separately the weak one-sided average-case hardness with perfect sampling, which requires that the YES and NO distribution be contained completely within the respective parts of the problem, but places a much weaker requirement on their indistinguishability. This variant implies the existence of OWFs, though following an approach different from that taken in Lemma 4.

▶ **Lemma 6.** *If there is a problem that is weak one-sided average-case hard with perfect sampling, then One-Way Functions exist.*

2.1 Reductions

Reductions relate the complexities of different problems, and while there are more involved forms of reductions that still do so, in our work we will be using the simple notion of Karp reductions. Roughly, a Karp reduction from L to L' takes an instance x and produces an instance x' whose membership in L' is completely determined by that of x in L. Later definitions below also ask for some additional properties.

▶ **Definition 7** (Karp Reduction). *A polynomial-time algorithm R is a* Karp reduction *from a problem L to a problem L' if for all large enough n:*

$$x \in L_Y \cap \{0,1\}^n \implies \Pr[\mathsf{R}(x) \in L'_Y] \geq 0.9$$
$$x \in L_N \cap \{0,1\}^n \implies \Pr[\mathsf{R}(x) \in L'_N] \geq 0.9$$

In this case, L is said to reduce *to L'. If the above probabilities are both 1, then A is said to be a* perfect *Karp reduction.*

A worst-case reduction like the one above from L to L' allows us to conclude that L' is worst-case hard if L is worst-case hard. Throughout the rest of this work, we will be drawing conclusions about the two-sided and one-sided average-case hardness of L' that follow from analogous notions of worst-to-average-case reductions defined below.

▶ **Definition 8** (Two-Sided Average-Case Karp Reduction). *A polynomial-time algorithm* R *is a* two-sided average-case Karp reduction *from a problem* L *to a problem* L' *if there are two polynomial-time samplers* Y *and* N *such that the following hold for all large enough* n*:*

- Y *and* N *mostly sample instances from* L'_Y *and* L'_N*, respectively. That is,*

$$\Pr_{x \leftarrow \mathsf{Y}(1^n)}[x \notin L'_Y] \leq 0.1$$

$$\Pr_{x \leftarrow \mathsf{N}(1^n)}[x \notin L'_N] \leq 0.1$$

- *For any* $x \in L_N \cap \{0,1\}^n$*, the output of* $\mathsf{R}(x)$ *is close to that of* $\mathsf{N}(1^n)$*. That is, for such* x*,*

$$\Delta\left(\mathsf{R}(x); \mathsf{N}(1^n)\right) \leq 0.1$$

- *For any* $x \in L_Y \cap \{0,1\}^n$*, the output of* $\mathsf{R}(x)$ *is close to that of* $\mathsf{Y}(1^n)$*. That is, for such* x*,*

$$\Delta\left(\mathsf{R}(x); \mathsf{Y}(1^n)\right) \leq 0.1$$

A two-sided average-case reduction from L to L' allows us to conclude that L' is two-sided average-case hard if L is worst-case hard.

▶ **Lemma 9.** *Suppose there is a two-sided average-case Karp reduction from a language* L *to a language* L'*, and* L *is worst-case hard. Then,* L' *is two-sided average-case hard.*

Finally, we define one-sided average-case reductions, which similarly relate one-sided average-case hardness to worst-case hardness.

▶ **Definition 10** (One-Sided Average-Case Karp Reduction). *A polynomial-time algorithm* R *is a* one-sided average-case Karp reduction *from a problem* L *to a problem* L' *if there is a polynomial-time sampler* N *such that the following hold for all large enough* n*:*

- N *mostly samples instances from* L'_N*. That is,*

$$\Pr_{x \leftarrow \mathsf{N}(1^n)}[x \notin L'_N] \leq 0.1$$

- *For any* $x \in L_N \cap \{0,1\}^n$*, the output of* $\mathsf{R}(x)$ *is close to that of* $\mathsf{N}(1^n)$*. That is, for such* x*,*

$$\Delta\left(\mathsf{R}(x); \mathsf{N}(1^n)\right) \leq 0.1$$

- *For any* $x \in L_Y \cap \{0,1\}^n$*, the output of* $\mathsf{R}(x)$ *is mostly contained in* L'_Y*. That is, for such* x*,*

$$\Pr_{x' \leftarrow \mathsf{R}(x)}[x' \notin L'_Y] \leq 0.1$$

The reduction is said to be strong *if for any* $x \in L_N \cap \{0,1\}^n$*, the distance* $\Delta\left(\mathsf{R}(x); \mathsf{N}(1^n)\right)$ *is negligible in* n*, and is* weak *if this is at most* $1 - 1/n^c$ *for some constant* c*. The reduction is said to have* perfect sampling *if the probability that the outputs of* N *or* $\mathsf{R}(x)$ *when* $x \in L_Y$ *are not in* L'_N *and* L'_Y*, respectively, are* 0*.*

▶ **Lemma 11.** *Suppose there is a one-sided average-case Karp reduction from a language L to a language L', and L is worst-case hard. Then, L' is one-sided average-case hard. Further, if the reduction is weak and has perfect sampling, then the hardness is also weak and has perfect sampling.*

3 Statistical Zero Knowledge

One of the paths we take to showing the existence of OWFs is through a one-sided average-case hard problem that also has a statistical zero-knowledge proof. The class SZK of problems that have statistical zero-knowledge proofs has been widely studied in the past, partly owing to its connections to cryptography (see [29] and discussions and references therein). Due to a completeness theorem of Sahai and Vadhan [27], we may equivalently define this class in the following manner that is more convenient for us.

▶ **Definition 12** (Statistical Zero Knowledge [27])**.** Statistical Zero Knowledge *(*SZK*) is the class of promise problems which have a perfect Karp reduction to the* Statistical Difference problem *(*SD*), which is defined over pairs of circuits (C_0, C_1), as follows:*

$$\mathrm{SD}_Y = \{(C_0, C_1) \mid \Delta(C_0(U)); C_1(U))) > 2/3\}$$
$$\mathrm{SD}_N = \{(C_0, C_1) \mid \Delta(C_0(U)); C_1(U))) < 1/3\}$$

where n is the output length of both C_0 and C_1, and the U's above represent uniform distributions over the appropriate input domains.

We will also use the following two results regarding SZK.

▶ **Lemma 13** (Polarization [27])**.** *There exists an efficient procedure* Polarize *that when given two circuits C_0, C_1 and a parameter 1^λ as input outputs C_0', C_1' such that*

- *If $\Delta(C_0(U); C_1(U)) > 2/3$, then $\Delta(C_0'(U); C_1'(U)) > 1 - 2^{-\lambda}$.*
- *If $\Delta(C_0(U); C_1(U)) < 1/3$, then $\Delta(C_0(U); C_1(U)) < 2^{-\lambda}$.*

▶ **Theorem 14** (SZK closed under complement [24])**.** *If $\Pi = (\Pi_Y, \Pi_N) \in \mathsf{SZK}$, then $\overline{\Pi} = (\overline{\Pi}_Y, \overline{\Pi}_N) \in \mathsf{SZK}$ where $\overline{\Pi}_Y = \Pi_N$ and $\overline{\Pi}_N = \Pi_Y$.*

As noted in Lemma 4, two-sided average-case hardness of any problem implies the existence of OWFs. Ostrovsky [25] showed that *plain* average-case hardness of any problem in SZK also implies the existence of OWFs. Furthermore, Ostrovsky and Wigderson [26] observe that the *worst-case* hardness of any problem in SZK implies the existence of *auxiliary-input* one-way functions.[6] We show the following incomparable theorem.

▶ **Theorem 15.** *If there is a problem in* SZK *that is one-sided average-case hard, then One-Way Functions exist.*

[6] Roughly, auxiliary-input one-way functions (against non-uniform adversaries) exist if there is polynomial p such that for every family of poly size circuits, $\{\mathsf{A}_n\}_{n\in\mathbb{N}}$ there is family of circuits of size $p(n)$, $\{\mathsf{F}_n\}_{n\in\mathbb{N}}$, such that A_n fails to invert F_n when given a description of F_n as an auxiliary input. Contrast this with the notion of non-uniform one-way functions we use throughout: a (fixed) family of poly-sized circuits, $\{\mathsf{F}_n\}_{n\in\mathbb{N}}$ is a non-uniform one-way function if for *any* poly-sized circuit family $\{\mathsf{F}_n\}$ is hard to invert. So far as we know, the latter is strictly stronger. Interestingly however, if one is concerned with security against *uniform* adversaries, the quantifiers can be switched via diagonalization arguments and the two notions are equivalent.

Proof. We show that such hardness implies the existence of two samplable distributions that are α-statistically far but β-computationally indistinguishable, where there is a noticeable gap between α and β. (Here, we will take $\alpha = 4/5 - \text{negl}(n)$ and $\beta = 3/4$.) The rest follows from the results of Goldreich and Naor-Rothblum [15, 23].

The completeness of the complement of Statistical Difference(by Theorem 14) implies that for any problem L in SZK, there is a reduction R that takes input x and outputs two circuits that sample distributions that are far if $x \in L_N$, and negligibly close if $x \in L_Y$. Let L be the one-sided average-case hard problem in SZK and let N be the sampler for its fixed NO distribution. The two distributions we want are sampled by the samplers D_0 and D_1 below given security parameter n:

- $\mathsf{D}_0(1^n)$: Sample $x \leftarrow \mathsf{N}(1^n)$. Compute $(C_0, C_1) \leftarrow \mathsf{R}(x)$. Compute circuits $(C'_0, C'_1) \leftarrow \text{Polarize}(1^n, C_0, C_1)$. Pick random r of appropriate length and output $((C'_0, C'_1), C_0(r))$.
- $\mathsf{D}_1(1^n)$: Same as above, but output $((C'_0, C'_1), C'_1(r))$ at the end.

By definition, the event that $\mathsf{N}(1^n)$ outputs something not in L_N happens with probability at most $1/10$. By the definition of $\overline{\text{SD}}$, conditioned on this event not happening, for any fixed output C_0, C_1 we have that $\Delta(C_0(U); C_1(U)) > 2/3$. If this is the case, it follows from Lemma 13 that $\Delta(C'_0(U); C'_1(U)) > 1 - 2^{-n}$ It follows that, *conditioned on this event not happening*, we have $\Delta(\mathsf{D}_0(1^n); \mathsf{D}_1(1^n)) > 1 - \text{negl}(n)$. By standard manipulations[7], we get that $\Delta(\mathsf{D}_0(1^n); \mathsf{D}_1(1^n)) > .8 - \text{negl}(n)$.

To see that these distributions are 3/4-computationally indistinguishable, consider any distinguisher A for them. Suppose A has more than 3/4 advantage in distinguishing $\mathsf{D}_0(1^n)$ from $\mathsf{D}_1(1^n)$. But then consider A' that attempts to solve L by on input x running the reduction,R, and polarization procedure, $\text{Polarize}(1^n, \cdot, \cdot)$ to get C'_0, C'_1, flipping a coin $b \leftarrow U_{\{0,1\}}$, and outputting 1 if $\mathsf{A}(C'_0, C'_1, C'_b(U)) = b$. If A' is given inputs from $N(1^n)$, then what it feeds A is identically distributed to either $D_0(1^n)$ (if $b = 0$) or $D_1(1^n)$ (if $b = 1$). It follows from A's advantage that $Pr[\mathsf{A}'(N(1^n))] > \frac{1+3/4}{2} = .875$.

On the other hand, by virtue of the fact that L is one-sided average case hard there is a distribution $\mathsf{Y}_{\mathsf{A}'}$ which is .3-indistinguishable from N to A'. Consider $\mathsf{A}'(\mathsf{Y}_{\mathsf{A}'})$. First of all $\Pr[\mathsf{Y}_{\mathsf{A}'} \in L_Y] \geq .9$. Moreover, for any such $x \in L_Y$, $\mathsf{R}(x) \in \overline{\text{SD}}_N$. In this case, $\text{Polarize}(1^n)$ will output (C'_0, C'_1) such that $\Delta(C'_0(U); C'_1(U)) < 2^{-n}$. Therefore, with probability at least .9, the two circuits A' gives to A correspond to distributions with negligible distance from one another. It follows from standard manipulations that A, regardless of efficiency, can distinguish with probability at most $.1 + \text{negl}(n)$. Thus, $\Pr[\mathsf{A}'(\mathsf{Y}_{\mathsf{A}'}) = 1] \leq \frac{1+.15}{2} = .575$.

But then $\Pr[\mathsf{A}'(\mathsf{N}_{\mathsf{A}'}) = 1] - \Pr[\mathsf{A}'(\mathsf{Y}_{\mathsf{A}'}) = 1] > .3$, which violates the assumption on $\mathsf{Y}_{\mathsf{A}'}$ following from the average-case hardness of L. ◀

The above theorem talks about the implication of a problem that is both one-sided hard and is in SZK. The following lemma, which was implicitly used by Drucker [13], says that membership of a problem in SZK is implied by the existence of any one-sided average-case reduction *from* it. A stronger version of this lemma (in different terminology) was also proven by Applebaum and Raykov [5].

▶ **Lemma 16.** *If there is a one-sided average-case Karp reduction from a problem L to any other problem, then L is in* SZK.

[7] For any random variables X, Y, Z and any event E, $\Delta(X;Y) \in [\Pr[Z \in E]\Delta(X|Z \in E; Y|Z \in E) \pm \Pr[Z \notin E]]$.

Proof of Lemma 16. We show this by reduction to the Statistical Difference problem, and appealing to its completeness of SZK and the closure of the class under complement. Suppose R is the one-sided average-case Karp reduction and N is the canonical NO distribution from Definition 10. The reduction is, given input x for L, to output the pair of circuits $(\mathsf{N}(1^n;\cdot), \mathsf{R}(x;\cdot))$ – each of these takes the randomness for the respective algorithm as input and produces the corresponding output. The properties of the one-sided reduction guarantee that if $x \in L_N$, then $\Delta(\mathsf{N}(1^n); \mathsf{R}(x))$ is at most 0.1, while if $x \in L_Y$, then this is at least 0.9. ◀

Because worst-case hard languages in SZK imply auxiliary-input one-way functions [26], the following is an immediate consequence of Lemma 16. We refer the reader to [26] for the definition of auxiliary-input one-way functions.

▶ **Corollary 17.** *If L is worst-case hard and there is a one-sided average-case Karp reduction from L to any other problem, then auxiliary-input one-way functions exist.*

Starting with a stronger hypothesis – a *two-sided* average-case reduction from L – we show membership in SRE, which is the class of problems that have statistical randomized encodings (and is a subset of SZK). We refer the reader to [5] for the definitions of randomized encodings and this class.

▶ **Lemma 18.** *If there is a two-sided average-case Karp reduction from a problem L to any other problem, then L is in* SRE*.*

This lemma is proven in the same way as Lemma 16, with the two fixed distributions of the randomized encodings taken to be Y and N from the two-sided average-case reduction (see Definition 8).

4 Lossy Reductions

In this section, we define our notions of lossy reductions, and show their implications for OWFs. We start with a definition of what it means for a generic algorithm to lose information about its input. All algorithms in this section are randomized and non-uniform unless specified otherwise.

▶ **Definition 19** (Lossy Algorithm)**.** *An algorithm* C *is said to* t-lossy on n-bit inputs *for some $n \in \mathbb{N}$ and $t \in \mathbb{R}^+$ if, for any random variable X over $\{0,1\}^n$,*

$$\mathrm{I}(X; \mathsf{C}(X)) \leq t$$

Note that being t-lossy means, in a sense, that the algorithm loses $(n - t(n))$ bits of information. For convenience, we overload the above terminology for the following special cases of reductions.

▶ **Definition 20** (Lossy Reduction)**.** *For a function $t : \mathbb{N} \to \mathbb{R}^+$, a Karp reduction from a problem L to a problem L' is said to be* t-lossy *if, for every $n \in \mathbb{N}$, it is $t(n)$-lossy on n-bit inputs.*

For a polynomial m, consider a family $f = \left\{f_n : \{0,1,\perp\}^{m(n)} \to \{0,1,\perp\}\right\}$ of partial functions. Define the problem $f \circ L$ on inputs from $\{0,1\}^{n \times m(n)}$ as the composition of f with $m(n)$ copies of L; that is, for $x_1, \ldots, x_{m(n)} \in \{0,1\}^n$, define $(f \circ L)(x_1, \ldots, x_{m(n)}) = f(L(x_1), \ldots, L(x_{m(n)}))$. We also overload the terminology for reductions from such compositions as follows.

▶ **Definition 21** (Lossy f-Reduction). *Let $m : \mathbb{N} \to \mathbb{N}$ and $t : \mathbb{N} \to \mathbb{R}^+$ be polynomials, and $f = \left\{f_n : \{0,1,\bot\}^{m(n)} \to \{0,1,\bot\}\right\}$ be a family of partial functions. For a problem L, a Karp reduction from $f \circ L$ to a problem L' is said to be* t-lossy *if, for every $n \in \mathbb{N}$, it is $t(n)$-lossy on $(n \cdot m(n))$-bit inputs. We say in this case that L has a t-lossy f-reduction to L'.*

In the last two definitions, the problem L is said to have a t-lossy *self-reduction* (or self-f-reduction) if L' is the same as L. During many of our discussions, the size parameter n will be fixed, and when it is, we will use just t and m to denote the numbers $t(n)$ and $m(n)$.

We first consider the family of OR functions – for some polynomial m, the family $\mathrm{OR}_m = \left\{\mathrm{OR}^n : \{0,1,\bot\}^{m(n)} \to \{0,1,\bot\}\right\}_{n\in\mathbb{N}}$, where OR^n is a function from $\{0,1,\bot\}^{m(n)}$ to $\{0,1,\bot\}$ that is the Boolean OR of its inputs if they are all from $\{0,1\}$, and is $\bot$ if any of its inputs is $\bot$. Drucker [13] showed that any problem that is OR_m-compressible to $O(m(n)\log n)$ bits is contained in SZK, where his notion of compression to t bits was that the output length of the reduction is at most t bits (similar to Definition 31). We observe that his proof works almost as is for reductions that are lossy in the sense of Definition 21 (but that may not be compressing). We isolate the following lemma that is implicit in [13], which will be useful for us, and state it in terms of average-case Karp reductions.

▶ **Lemma 22.** *Suppose, for some polynomial m and problems L, L', there is a reduction from $\mathrm{OR}_m \circ L$ to L'. If this reduction is $(m/100)$-lossy and $m(n) > 100$ for all large enough n, then there is a one-sided average-case Karp reduction from L to L'.*

As noted above, Drucker shows membership in SZK as long as the compression (or loss) is to $O(m \log n)$ bits, but the implication of the one-sided average-case reduction as in Lemma 22 seems to follow only if the loss is to somewhat less than m bits. In the case where the reduction from $\mathrm{OR}_m \circ L$ does not make any errors, however, we can recover a weak one-sided average-case reduction with perfect sampling even if the loss is only to $O(m \log n)$ bits.

▶ **Lemma 23.** *Suppose, for some polynomial m and problems L, L', there is a* perfect *reduction from the problem $\mathrm{OR}_m \circ L$ to L'. If this reduction is $O(m \log n)$-lossy, then there is a weak one-sided average-case Karp reduction with perfect sampling from L to L'.*

Define MAJ_m in the same way as OR_m, but with the Majority function. We extend Drucker's results and show the following stronger conclusion from lossy Majority-reduction – that it implies a two-sided average-case reduction (or a randomized encoding, following Lemma 18).

▶ **Lemma 24.** *Suppose, for some polynomial m and problems L, L', there is a reduction from the problem $\mathrm{MAJ}_m \circ L$ to L'. If this reduction is $m/100$-lossy and $m(n) > 100$ for all large enough n, then there is a two-sided average-case Karp reduction from L to L'.*

The proofs of these lemmas are in Appendix B.

4.1 Lossy Reductions and One-Way Functions

We now state and prove our main theorems, showing that certain lossy reductions for a problem L can be used to lift its worst-case hardness to a One-Way Function. Each of our three theorems is proven by following a different path, as explained in the respective proofs. The first says that good enough lossy self-OR-reduction implies a OWF.

▶ **Theorem 25.** *Suppose, for some polynomial m and a problem L, there is a reduction from $\mathrm{OR}_m \circ L$ to L that is $(m/100)$-lossy. If L is worst-case hard and $m(n) > 100$ for all large enough n, then One-Way Functions exist.*

Proof of Theorem 25. This theorem is proven by showing that the hypothesis implies that L is both one-sided average-case hard and contained in SZK which, as seen in Section 3, implies a OWF. The proof is as follows:

- Lemma 22 implies that there is a one-sided average-case reduction from L to itself.
- Lemma 16 and the one-sided reduction imply that $L \in \mathsf{SZK}$.
- Together with the worst-case hardness of L and the one-sided reduction, Lemma 11 implies that L is one-sided average-case hard.
- Theorem 15, along with the above two conclusions, implies that OWFs exist. ◀

Our second theorem uses an incomparable hypothesis – that there is a lossy MAJ-reduction, but not necessarily to L itself.

▶ **Theorem 26.** *Suppose, for some polynomial m and a problem L, there is a reduction from $\mathrm{MAJ}_m \circ L$ to some problem that is $(m/100)$-lossy. If L is worst-case hard and $m(n) > 100$ for all large enough n, then One-Way Functions exist.*

Proof of Theorem 26. This theorem is proven by showing that such a reduction implies the existence of a two-sided average-case reduction from L, and using this to go from worst-case hardness of L to OWFs. Suppose the reduction is to a problem L'. The proof is as follows:

- Lemma 24 implies that there is a two-sided average-case reduction from L to L'.
- Together with the worst-case hardness of L and the two-sided reduction, Lemma 9 implies that L' is two-sided average-case hard.
- Lemma 4, along with the above two-sided hardness, implies that OWFs exist. ◀

The third theorem also uses a hypothesis incomparable to the other two – it assumes that the lossy reduction is perfect, but works even if the loss is to more bits than the number of instances it takes as input.

▶ **Theorem 27.** *Suppose, for a polynomial m and a problem L, there is a* perfect *reduction from $\mathrm{OR}_m \circ L$ to some problem that is $O(m \log n)$-lossy. If L is also worst-case hard, then One-Way Functions exist.*

We draw the following interesting corollaries of this theorem, both of which follow from a direct implication of lossy perfect OR-reduction by their respective hypotheses. The first is a statement along the lines of Lemma 4, but incomparable to it. Lemma 4 may be rephrased as saying that a 1-lossy reduction from a worst-case hard language implies OWFs, and Corollary 28 relaxes the requirement to $O(\log n)$-lossiness, but requires the reduction to be perfect.

▶ **Corollary 28.** *Suppose there is a* perfect *reduction from a problem L to another problem that is $O(\log n)$-lossy. If L is also worst-case hard, then One-Way Functions exist.*

The second corollary is the analogue of Theorem 27 for Majority reductions.

▶ **Corollary 29.** *Suppose, for a polynomial m and a problem L, there is a* perfect *reduction from $\mathrm{MAJ}_m \circ L$ to some problem that is $O(m \log n)$-lossy. If L is also worst-case hard, then One-Way Functions exist.*

Proof of Theorem 27. This theorem is proven by showing that a perfect lossy OR-reduction leads to a weak one-sided average-case reduction with perfect sampling, which along with worst-case hardness leads to OWFs. Suppose the reduction is to a problem L'. The proof is as follows:

- Lemma 23 implies that there is a weak one-sided average-case reduction from L to L' with perfect sampling.
- Together with the worst-case hardness of L and the above reduction, Lemma 11 implies that L' is weak one-sided average-case hard with perfect sampling.
- Lemma 6, along with the above hardness, implies that OWFs exist. ◀

5 Collision Resistance

In this section, we show that that one-sided average-case hardness of a language L combined with compressiblility of $\mathrm{OR}_m \circ L$ (in the standard sense of compressed output length), implies the existence of *collision-resistant hash functions.*

The reduction builds directly on the construction of collision-resistant hash functions from homomorphic encryption due to Ishai, Kushilevitz, and Ostrovsky [21] (more generally, from any one-round private information retrieval (PIR) protocol, or homomorphic one-way commitments). Indeed, these cryptographic notions can be viewed precisely as providing compressing reductions for related languages whose average-case hardness is implied by security: e.g., additively homomorphic encryption corresponds directly to self-compression from $(\mathrm{XOR}_m \circ L)$ to L, for the (average-case hard) language L where L_Y is the set of ciphertexts encrypting the bit 1, and L_N is the set of ciphertexts of 0.

We begin by formally defining collision-resistant hash function families and the required form of output-length compression.

▶ **Definition 30.** Collision-resistant hash functions *exist if there exist* $\ell, \ell' : \mathbb{N} \to \mathbb{N}$ *with* $\ell(n) > \ell'(n)$*, an index set* $I \subseteq \{0,1\}^*$*, and probabilistic polynomial-time algorithms of the form:*

- $\mathit{Gen}(1^n)$*, which outputs an index* $s \in I$*,*
- $\mathit{Eval}(s, y)$*, which given index* $s \in I$ *and input* $y \in \{0,1\}^{\ell(n)}$*, outputs* $h_s(y) \in \{0,1\}^{\ell'(n)}$*,*

for which finding collisions is computationally hard. That is, for every polynomial-time (non-uniform) A*, there exists a negligible function* ν *for which*

$$\Pr_{\substack{s \leftarrow \mathit{Gen}(1^n) \\ (y,y') \leftarrow \mathsf{A}(s)}} [(y \neq y') \wedge (\mathit{Eval}(s,y) = \mathit{Eval}(s,y'))] \leq \nu(n).$$

▶ **Definition 31** (Compressing Reduction)**.** *An algorithm* C *is said to* compress n-bit inputs to t bits *for some* $n, t \in \mathbb{N}$ *if for any* $x \in \{0,1\}^n$ *the output bit length is* $|\mathsf{C}(x)| \leq t$*.*

For a function $t : \mathbb{N} \to \mathbb{N}$*, a Karp reduction from a problem* L *to a problem* L' *is said to* t-compressing *if, for every* $n \in \mathbb{N}$*, it is* $t(n)$*-compressing on* n*-bit inputs.*

We now present the main theorem of the section. Note for simplicity, we present the reduction with respect to a *perfect* compressing reduction of $\mathrm{OR}_m \circ L$ (see Definition 7), as well as assuming *perfect* sampling for the one-sided average-case hard language L (that is, for any polynomial-time A, the probability that the outputs of N or Y_A are not in L_N and L_Y, respectively, are set to 0).

▶ **Theorem 32.** *Suppose, for some polynomial* m *and a problem* L*, there is a perfect reduction from* $\mathrm{OR}_m \circ L$ *to* L' *that compresses to* $m/100$ *bits. If* L *is strongly one-sided average-case hard with perfect sampling and* $m(n) > 100$ *for all large enough* n*, then Collision-Resistant Hash Functions exist.*

Recall that existence of a compressing reduction from $(\mathrm{MAJ}_{2m+1} \circ L)$ to L', or from $(\mathrm{AND}_m \circ L)$ to L', each directly imply equivalent compressing reduction from $(\mathrm{OR}_m \circ L)$ to L', up to constant factors of compression. This yields the following corollary.

▶ **Corollary 33.** *The following variations of Theorem 32 additionally hold. Let m be polynomial and L a problem. Then collision-resistant hash functions exist assuming existence of a compressing reduction of the following corresponding kinds (to $m/100$ bits), in addition to the specified hardness requirements on L:*

1. *If there is a compressing reduction from $\mathrm{MAJ}_m \circ L$ to L' and L is strongly one-sided average-case hard with perfect sampling.*
2. *If there is a compressing reduction from $\mathrm{AND}_m \circ L$ to L' and the complement language $\mathsf{co}L$ is strongly one-sided average-case hard with perfect sampling.*

At a high level, the construction mirrors the approach of [21], as follows. Each hash function in the family will be keyed by a collection s of $2m$ randomly sampled no-instances of L (via N). The corresponding hash function $\mathsf{Eval}(s, \cdot)$ takes as input a bit-string $y \in \{0,1\}^m$, and outputs the $m/100$-bit instance of L' generated by applying the compressing reduction on the $(OR_m \circ L)$ instance defined by the m no-instances selected by the bits of y. A successful collision-finder can be used to gain contradictorily high advantage in deciding L (more specifically, in distinguishing a sample of N from the (possibly inefficient) Y_A), by embedding the challenge instance x into a random location $i^* \in [m]$ of the key, and seeing whether the colliding inputs $y \neq y'$ differ in this index i^*. If x was sampled from N, then the concocted key is properly distributed, independent of i^*, and thus y, y' must differ in position i^* with noticeable probability. On the other hand, if x was sampled from Y_A, then any $(\mathrm{OR}_m \circ L)$ instance containing x is a *yes* instance, whereas any instance not containing x is (by construction) a *no* instance; by perfect correctness of the compressing reduction, it thus cannot be the case that any two such inputs y, y' that differ in position i^* could collide to an identical output in L'.

Proof of Theorem 32. Given such a pair of languages L, L', we construct the desired collision-resistant hash function family.

Denote by R the weakly compressing reduction from $(\mathrm{OR}_m \circ L)$ to L'. By one-sided average-case hardness of L (as per Definition 5), there exists a polynomial-time sampling algorithm N which samples instances from L_N. Let $\ell(n) = m$ and $\ell'(n) = m/100$ be the input and (compressed) output length of the hash function. Consider the following algorithms.

- $\mathsf{Gen}(1^n)$: Independently sample $\ell(n) = m$ instances from N. That is, for $(i, b) \in [m] \times \{0,1\}$, let $x_{i,b} \leftarrow \mathsf{N}(1^n)$. Output $s = (x_{i,b})_{i \in [m], b \in \{0,1\}}$.
- $\mathsf{Eval}(s, y)$: Parse $s = (x_{i,b})_{i \in [m], b \in \{0,1\}}$ and $y = (y_1, \ldots, y_m) \in \{0,1\}^m$. Output the R-compression of the $\mathrm{OR}_m \circ L$ instance selected by y. That is, output $\mathsf{R}\left((x_{i,y_i})_{i \in [\ell]}\right) \in \{0,1\}^{m/100}$.

We prove that the above constitutes a collision-resistant hash function family, by showing that any successful collision finder would violate one-sided average-case hardness of L.

Suppose there exists a (non-uniform) polynomial-time algorithm A for which

$$\Pr_{\substack{s \leftarrow \mathsf{Gen}(1^n) \\ (y,y') \leftarrow \mathsf{A}(s)}} [(y \neq y') \wedge (\mathsf{Eval}(s, y) = \mathsf{Eval}(s, y'))] = \epsilon.$$

Consider the following associated algorithm A', which receives as input a bit string $x \in \{0,1\}^n$ and outputs a bit (corresponding to a prediction for $x \in L_Y$ or L_N). Intuitively, A' embeds the input x into a random index of the hash function description, runs the collision-finder A, and outputs 1 if A successfully finds a collision which differs in the embedded index.

Algorithm 1 Algorithm $\mathsf{A}'(x)$.

1. Select a random index $(i^*, b^*) \leftarrow [m] \times \{0,1\}$; let $x_{i^*,b^*} := x$.
   For every $(i,b) \neq (i^*,b^*)$, sample $x_{i,b} \leftarrow \mathsf{N}$. Set $s = (x_{i,b})_{i\in[m],b\in\{0,1\}}$.
2. Execute $(y, y') \leftarrow \mathsf{A}(s)$.
3. If it holds that: (1) $y \neq y'$, (2) $\mathsf{Eval}(s,y) = \mathsf{Eval}(s,y')$, and (3) $y_{i^*} \neq y'_{i^*}$, then output 1.
   Else, output a randomly selected bit $c \leftarrow \{0,1\}$.

By the strong one-sided average-case hardness of L, there exists a negligible function ν' and (possibly inefficient) sampler algorithm $\mathsf{Y}_{A'}$ corresponding to A', for which

$$\left| \Pr_{x\leftarrow\mathsf{N}(1^n)}[\mathsf{A}'(x)=1] - \Pr_{x\leftarrow\mathsf{Y}_{\mathsf{A}'}(1^n)}[\mathsf{A}'(x)=1] \right| \leq \nu'(n). \tag{1}$$

Claim 1: $\Pr_{x\leftarrow\mathsf{N}(1^n)}[\mathsf{A}'(x)=1] \geq \epsilon/m$. Given $x \leftarrow \mathsf{N}(1^n)$, the value of s as generated by A' is identically distributed to that of $\mathsf{Gen}(1^n)$, independent of the selected choice of $i^* \in [m]$. This implies that

$$\begin{aligned}\Pr_{x\leftarrow\mathsf{N}(1^n)}[\mathsf{A}'(x)=1] &= \Pr_{\substack{s\leftarrow\mathsf{Gen}(1^n)\\(y,y')\leftarrow\mathsf{A}(s)}}[(y\neq y') \wedge (\mathsf{Eval}(s,y)=\mathsf{Eval}(s,y')) \wedge (y_{i^*}\neq y'_{i^*})] \\ &\geq \Pr_{\substack{s\leftarrow\mathsf{Gen}(1^n)\\(y,y')\leftarrow\mathsf{A}(s)}}[(y\neq y') \wedge (\mathsf{Eval}(s,y)=\mathsf{Eval}(s,y'))] \cdot \frac{1}{m} = \frac{\epsilon}{m}.\end{aligned}$$

Claim 2: $\Pr_{x\leftarrow\mathsf{Y}_{\mathsf{A}'}(1^n)}[\mathsf{A}'(x)=1] = 0$. Assuming perfect correctness $\Pr_{x\leftarrow\mathsf{Y}_{\mathsf{A}'}}[x \in L_Y] = 1$ and $\Pr_{x\leftarrow\mathsf{N}}[x \in L_N] = 1$, then the embedded instance satisfies $x_{i^*,b^*} \in L_Y$, whereas the ambient instances are in $x_{i,b} \in L_N$ for $(i,b) \neq (i^*,b^*)$. This means for any $y, y' \in \{0,1\}^m$ in which $y_{i^*} \neq y'_{i^*}$, the corresponding selected $\mathrm{OR}_m \circ L$ instances *necessarily* disagree: $(x_{i,y_i})_{i\in[m]} \in (\mathrm{OR}_m \circ L)_Y$ and $(x_{i,y'_i})_{i\in[m]} \in (\mathrm{OR}_m \circ L)_N$, or vice versa. However, by perfect correctness of the compression algorithm R, this implies $\mathsf{R}((x_{i,y_i})_{i\in[m]}) \neq \mathsf{R}((x_{i,y'_i})_{i\in[m]})$. Thus, for any $y, y' \in \{0,1\}^m$ with $y_{i^*} \neq y'_{i^*}$, it must be the case that $\mathsf{Eval}(s,y) \neq \mathsf{Eval}(s,y')$.

Combining the two above claims together with Equation (1) implies that the collision-finding success probability ϵ of A must be bounded above by a negligible value. The theorem follows. ◀

References

1. Benny Applebaum. Cryptographic Hardness of Random Local Functions - Survey. *Computational Complexity*, 25(3):667–722, 2016. doi:10.1007/s00037-015-0121-8.
2. Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography in NC^0. In *45th Symposium on Foundations of Computer Science (FOCS 2004), 17-19 October 2004, Rome, Italy, Proceedings*, pages 166–175. IEEE Computer Society, 2004. doi:10.1109/FOCS.2004.20.
3. Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. From Secrecy to Soundness: Efficient Verification via Secure Computation. In Samson Abramsky, Cyril Gavoille, Claude Kirchner, Friedhelm Meyer auf der Heide, and Paul G. Spirakis, editors, *Automata, Languages and Programming, 37th International Colloquium, ICALP 2010, Bordeaux, France, July 6-10, 2010, Proceedings, Part I*, volume 6198 of *Lecture Notes in Computer Science*, pages 152–163. Springer, 2010. doi:10.1007/978-3-642-14165-2_14.
4. Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Minimizing Locality of One-Way Functions via Semi-private Randomized Encodings. *J. Cryptology*, 31(1):1–22, 2018. doi:10.1007/s00145-016-9244-6.

5 Benny Applebaum and Pavel Raykov. On the Relationship Between Statistical Zero-Knowledge and Statistical Randomized Encodings. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part III*, volume 9816 of *Lecture Notes in Computer Science*, pages 449–477. Springer, 2016. doi:10.1007/978-3-662-53015-3_16.

6 Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009. URL: http://www.cambridge.org/catalogue/catalogue.asp?isbn=9780521424264.

7 Donald Beaver, Silvio Micali, and Phillip Rogaway. The Round Complexity of Secure Protocols (Extended Abstract). In Harriet Ortiz, editor, *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, May 13-17, 1990, Baltimore, Maryland, USA*, pages 503–513. ACM, 1990. doi:10.1145/100216.100287.

8 Itay Berman, Akshay Degwekar, Ron D. Rothblum, and Prashant Nalini Vasudevan. Statistical Difference Beyond the Polarizing Regime. *Electronic Colloquium on Computational Complexity (ECCC)*, 26:38, 2019. URL: https://eccc.weizmann.ac.il/report/2019/038.

9 Hans L. Bodlaender, Rodney G. Downey, Michael R. Fellows, and Danny Hermelin. On problems without polynomial kernels. *J. Comput. Syst. Sci.*, 75(8):423–434, 2009. doi:10.1016/j.jcss.2009.04.001.

10 Mark Braverman, Ankit Garg, Denis Pankratov, and Omri Weinstein. From information to exact communication. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 151–160. ACM, 2013. doi:10.1145/2488608.2488628.

11 Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013. doi:10.1007/978-1-4471-5559-1.

12 Andrew Drucker. New Limits to Classical and Quantum Instance Compression. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012*, pages 609–618, 2012.

13 Andrew Drucker. New Limits to Classical and Quantum Instance Compression. *SIAM J. Comput.*, 44(5):1443–1479, 2015. doi:10.1137/130927115.

14 Lance Fortnow and Rahul Santhanam. Infeasibility of instance compression and succinct PCPs for NP. *J. Comput. Syst. Sci.*, 77(1):91–106, 2011. doi:10.1016/j.jcss.2010.06.007.

15 Oded Goldreich. A Note on Computational Indistinguishability. *Inf. Process. Lett.*, 34(6):277–281, 1990. doi:10.1016/0020-0190(90)90010-U.

16 Oded Goldreich. *The Foundations of Cryptography - Volume 1: Basic Techniques*. Cambridge University Press, 2001.

17 Oded Goldreich, Russell Impagliazzo, Leonid A. Levin, Ramarathnam Venkatesan, and David Zuckerman. Security Preserving Amplification of Hardness. In *FOCS*, pages 318–326. IEEE Computer Society, 1990.

18 Danny Harnik and Moni Naor. On the Compressibility of *NP* Instances and Cryptographic Applications. *SIAM J. Comput.*, 39(5):1667–1713, 2010. doi:10.1137/060668092.

19 Yuval Ishai and Eyal Kushilevitz. Randomizing Polynomials: A New Representation with Applications to Round-Efficient Secure Computation. In *41st Annual Symposium on Foundations of Computer Science, FOCS 2000, 12-14 November 2000, Redondo Beach, California, USA*, pages 294–304, 2000.

20 Yuval Ishai and Eyal Kushilevitz. Perfect Constant-Round Secure Computation via Perfect Randomizing Polynomials. In Peter Widmayer, Francisco Triguero Ruiz, Rafael Morales Bueno, Matthew Hennessy, Stephan Eidenbenz, and Ricardo Conejo, editors, *Automata, Languages and Programming, 29th International Colloquium, ICALP 2002, Malaga, Spain, July 8-13, 2002, Proceedings*, volume 2380 of *Lecture Notes in Computer Science*, pages 244–256. Springer, 2002. doi:10.1007/3-540-45465-9_22.

21 Yuval Ishai, Eyal Kushilevitz, and Rafail Ostrovsky. Sufficient Conditions for Collision-Resistant Hashing. In Joe Kilian, editor, *Theory of Cryptography, Second Theory of Cryptography Conference, TCC 2005, Cambridge, MA, USA, February 10-12, 2005, Proceedings*, volume 3378 of *Lecture Notes in Computer Science*, pages 445–456. Springer, 2005. doi:10.1007/978-3-540-30576-7_24.

22 Richard J. Lipton and Neal E. Young. Simple strategies for large zero-sum games with applications to complexity theory. In Frank Thomson Leighton and Michael T. Goodrich, editors, *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing, 23-25 May 1994, Montréal, Québec, Canada*, pages 734–740. ACM, 1994. doi:10.1145/195058.195447.

23 Moni Naor and Guy N. Rothblum. Learning to impersonate. In William W. Cohen and Andrew Moore, editors, *Machine Learning, Proceedings of the Twenty-Third International Conference (ICML 2006), Pittsburgh, Pennsylvania, USA, June 25-29, 2006*, volume 148 of *ACM International Conference Proceeding Series*, pages 649–656. ACM, 2006. doi:10.1145/1143844.1143926.

24 Tatsuaki Okamoto. On Relationships between Statistical Zero-Knowledge Proofs. *J. Comput. Syst. Sci.*, 60(1):47–108, 2000. doi:10.1006/jcss.1999.1664.

25 Rafail Ostrovsky. One-Way Functions, Hard on Average Problems, and Statistical Zero-Knowledge Proofs. In *Proceedings of the Sixth Annual Structure in Complexity Theory Conference, Chicago, Illinois, USA, June 30 - July 3, 1991*, pages 133–138, 1991.

26 Rafail Ostrovsky and Avi Wigderson. One-Way Fuctions are Essential for Non-Trivial Zero-Knowledge. In *Second Israel Symposium on Theory of Computing Systems, ISTCS 1993, Natanya, Israel, June 7-9, 1993, Proceedings*, pages 3–17. IEEE Computer Society, 1993. doi:10.1109/ISTCS.1993.253489.

27 Amit Sahai and Salil P. Vadhan. A complete problem for statistical zero knowledge. *J. ACM*, 50(2):196–249, 2003. doi:10.1145/636865.636868.

28 Daniel R. Simon. Finding Collisions on a One-Way Street: Can Secure Hash Functions Be Based on General Assumptions? In Kaisa Nyberg, editor, *Advances in Cryptology - EUROCRYPT '98, International Conference on the Theory and Application of Cryptographic Techniques, Espoo, Finland, May 31 - June 4, 1998, Proceeding*, volume 1403 of *Lecture Notes in Computer Science*, pages 334–345. Springer, 1998. doi:10.1007/BFb0054137.

29 Salil Pravin Vadhan. *A study of statistical zero-knowledge proofs*. PhD thesis, Massachusetts Institute of Technology, 1999.

30 Andrew Chi-Chih Yao. Theory and Applications of Trapdoor Functions (Extended Abstract). In *FOCS*, pages 80–91. IEEE Computer Society, 1982.

31 Andrew Chi-Chih Yao. How to Generate and Exchange Secrets (Extended Abstract). In *27th Annual Symposium on Foundations of Computer Science, Toronto, Canada, 27-29 October 1986*, pages 162–167. IEEE Computer Society, 1986. doi:10.1109/SFCS.1986.25.

A OWFs and One/Two-Sided Average-Case Hardness

In this section, we restate and prove the lemmas stated in Section 2 about average-case hardness and reductions, and their connections to one-way functions.

A.1 Proof of Lemma 6

▶ **Lemma 6.** *If there is a problem that is weak one-sided average-case hard with perfect sampling, then One-Way Functions exist.*

Proof. Let L be weak one-sided average-case hard with perfect sampling, and N the poly-time sampling algorithm, such that there is a constant c such that for any poly-time distinguisher D there is Y_D such that for almost all n, $|\Pr_r[\mathsf{D}(\mathsf{N}(1^n;r))=1] - \Pr[\mathsf{Y}_D(1^n;r)=1]| < 1 - 1/n^c$. Moreover, $\mathrm{Supp}(\mathsf{N}) \subseteq L_N$ and $\mathrm{Supp}(\mathsf{Y}) \subseteq L_Y$.

Let F be defined for output length n as $\mathsf{F}_n(x) := \mathsf{N}(1^n; x)$. We will show that F is a weak-one-way function. The lemma then follows from classical results. [30, 17, 16]

Suppose, for the sake of contradiction, there exists an efficient A that can invert F_n with probability $> 1 - 1/n^c$, for infinitely many n. Then, consider the (efficient) distinguisher D that simply checks if A was succesful. In particular, D on input x simply:

1. $y \leftarrow \mathsf{A}(x)$
2. If $\mathsf{F}_n(y) = x$, output 1. Otherwise, output 0.

Because A has advantage $1 - 1/n^c$ on F_n, $\Pr_r[\mathsf{A}(\mathsf{N}(1^n; r)) = 1] > 1 - 1/n^c$, for infinitely many n. Now by our assumption on L, there exists Y_D such that $|\Pr_r[\mathsf{D}(\mathsf{N}(1^n; r)) = 1] - \Pr_r[\mathsf{D}(\mathsf{Y}_D(1^n; r)) = 1]| < 1 - 1/n^c$ and $\mathrm{Supp}(\mathsf{Y}_D) \cap \mathrm{Supp}(\mathsf{N}) = \emptyset$, for almost all n. It follows from the latter condition that A will never find preimages (under F) for outputs from Y_D, because they don't exist. It follows that $\Pr_r[\mathsf{D}(Y_D(1^n; r))] = 0$.

But then, $\Pr[\mathsf{D}(\mathsf{N}) = 1] - \Pr[\mathsf{D}(\mathsf{Y}_D) = 1] > 1 - 1/n^c$ for infinitely many n, which contradicts the weak one-sided average-case hardness of L. ◀

A.2 Proof of Lemma 9

▶ **Lemma 9.** *Suppose there is a two-sided average-case Karp reduction from a language L to a language L', and L is worst-case hard. Then, L' is two-sided average-case hard.*

Let Y, N be the samplers guaranteed by the two-sided-ness of the reduction from L to L', R. Let Y, N denote the random variable distributed according to Y and N evaluated on random inputs, respectively.

It suffices to show that for every efficient A, $|\Pr[\mathsf{A}(Y) = 1] - \Pr[\mathsf{A}(Y) = 1]| \leq 3/10$ (for almost all n).

Suppose not, and let A be a counter-example to the above in that $\Delta(\mathsf{A}(Y); \mathsf{A}(N)) > 3/10$ for infinitely many n. Moreover, suppose without loss of generality that $\Pr[\mathsf{A}(Y) = 1] > \Pr[\mathsf{A}(N) = 1]$. Then define A' to be the algorithm for L that on input x: (i) runs the reduction to get $z \leftarrow \mathsf{R}(x)$, then (ii) runs $b \leftarrow \mathsf{A}(z)$, and finally (iii) outputs b. Let R_x be the random variable distributed according to $\mathsf{R}(x)$. By definition, it is the case that $\Delta(R_x; Y) \leq 1/10$ for all $x \in L_Y$ and similarly $\Delta(R_x; N) \leq 1/10$ for all $x \in L_N$. It follows that $\Delta(\mathsf{A}(R_x); \mathsf{A}(N)) \leq 1/10$ for all $x \in L_Y$ and similarly $\Delta(\mathsf{A}(R_x); \mathsf{A}(Y)) \leq 1/10$ for all $x \in L_N$.

Consequently,

$$\begin{aligned} x \in L_Y &\implies \Pr[\mathsf{A}'(x) = 1] = \Pr[\mathsf{A}(R_x) = 1] \geq \Pr[\mathsf{A}(Y) = 1] - 1/10 \\ x \in L_N &\implies \Pr[\mathsf{A}'(x) = 1] = \Pr[\mathsf{A}(R_x) = 1] \leq \Pr[\mathsf{A}(N) = 1] + 1/10 \\ &\qquad\qquad \leq \Pr[\mathsf{A}(Y) = 1] - 2/10. \end{aligned}$$

It follows from standard amplification that L is not worst-case hard, contradicting our assumption on L.

A.3 Proof of Lemma 11

▶ **Lemma 11.** *Suppose there is a one-sided average-case Karp reduction from a language L to a language L', and L is worst-case hard. Then, L' is one-sided average-case hard. Further, if the reduction is weak and has perfect sampling, then the hardness is also weak and has perfect sampling.*

Let R be the one-sided average-case Karp reduction from L to L' with a corresponding NO-instance sampler N. In order to show that L' is one-sided average-case hard, we need to show a NO-instance sampler for L' and, for any polynomial-time A, a YES-distribution Y_A that it cannot distinguish from it. Our NO-instance sampler will be the N from the reduction itself, which satisfies the property that its samples are in L'_N except with probability 0.1.

Given a polynomial-time algorithm A, we claim that, for all large enough n, there exists an $x_y \in L_Y \cap \{0,1\}^n$ such that A cannot distinguish between $\mathsf{R}(x_y)$ and $\mathsf{N}(1^n)$ with advantage more than 0.3. If this were not the case, that is, if for every $x \in L_Y \cap \{0,1\}^n$, the algorithm A could distinguish between $\mathsf{R}(x)$ and $\mathsf{N}(1^n)$ with advantage more than 0.3, then the algorithm A' that, on input x, estimates the probability that $\mathsf{A}(\mathsf{R}(x)) = 1$ up to error say 0.01 could be used to decide L on all instances $x \in (L_Y \cup L_N) \cap \{0,1\}^n$, as for all $x \in L_N$, the reduction promises that $\mathsf{R}(x)$ is at most 0.1-far from $\mathsf{N}(1^n)$. The distribution $\mathsf{Y}_\mathsf{A}(1^n)$ is simply $\mathsf{R}(x_y)$ for such an x_y, and the reduction guarantees that $\mathsf{R}(x_y)$ is contained in L_Y except with probability 0.1, as necessary.

Note that if R is a weak one-sided average-case reduction with perfect sampling, then for any $x \in L_Y, \Pr[\mathsf{R}(x) \notin L'_Y] = 0$, $\Pr[\mathsf{N}(1^n) \notin L'_N] = 0$, and there is a constant c such that for any $x \in L_N$, $\Delta(\mathsf{N}(1^n); \mathsf{R}(x)) \leq 1 - n^{-c}$. Then, again, for any efficient A it follows that $\Delta(\mathsf{A}(\mathsf{N}(1^n)); \mathsf{A}(\mathsf{R}(x))) \leq 1 - n^{-c}$, for any $x \in L_Y$. Now, we can modify the A' above to approximate $\Pr[\mathsf{A}(\mathsf{R}(x)) = 1]$ to within a $\frac{1}{2n^c}$ factor. We claim, as above, that there is some $x_y \in L_Y$ $\Delta(\mathsf{A}(\mathsf{R}(x)); \mathsf{A}(\mathsf{N}(1^n))) \leq 1 - \frac{1}{4n^c}$. If not then for every $x \in L_Y \cup L_N$, $\Delta(\mathsf{A}(\mathsf{R}(x)); \mathsf{A}(\mathsf{N}(1^n))) > 1 - \frac{1}{4n^c}$ and A' will output such that $\mathsf{A}'(x) = L(x)$ with overwhelming probability. It follows that such an x_y exists and again we can take $\mathsf{Y}_A(1^n) = \mathsf{R}(x_y)$. Because $x \in L_Y, \Pr[\mathsf{R}(x) \notin L'_Y] = 0$, it follows that $\Pr[\mathsf{Y}_A(1^n) \notin L'_Y] = 0$.

B Proofs for Section 4

In this section, we restate and prove the lemmas used in Section 4. The proofs of Lemmas 22 and 23 are based on the following lemma that is implicit in [13], in particular following from the proofs of Theorem 7.1 and Claim 7.2 there. While the statement there is in terms of the compression (where the output length of the reduction itself is restricted to t bits), it may be verified that the proof only uses lossiness as in Definitions 19 and 21.

▶ **Lemma 34** ([13]). *Suppose, for some polynomials $m : \mathbb{N} \to \mathbb{N}$ and $t : \mathbb{N} \to \mathbb{R}$, a problem L has a t-lossy OR_m-reduction to a problem L', and the reduction has error $\epsilon(n) < 0.5$. Let*

$$\delta(n) = \min\left\{\sqrt{\frac{\ln 2}{2} \cdot \frac{t(n)+1}{m(n)}}, 1 - 2^{-\frac{t(n)}{m(n)}-3}\right\}$$

Then, for any constant c, there are polynomial-time algorithms A and B such that:

- $\Pr[\mathsf{A}(1^n) \notin L'_N] \leq \epsilon$
- *For any $x \in L_N \cap \{0,1\}^n$, $\Delta(\mathsf{A}(1^n); \mathsf{B}(x)) \leq \delta(n) + n^{-c}$*
- *For any $x \in L_Y \cap \{0,1\}^n$, $\Pr[\mathsf{B}(x) \notin L'_Y] \leq \epsilon$*

We now use this to prove the following lemmas.

▶ **Lemma 22.** *Suppose, for some polynomial m and problems L, L', there is a reduction from $\mathrm{OR}_m \circ L$ to L'. If this reduction is $(m/100)$-lossy and $m(n) > 100$ for all large enough n, then there is a one-sided average-case Karp reduction from L to L'.*

Proof of Lemma 22. The reduction from the hypothesis has error at most 0.1, and is t-lossy for $t = m/100$. Applying Lemma 34 with $c = 1$ gives us algorithms A and B as in its statement, with $\delta(n) < \sqrt{\ln 2/100}$ for large enough n. We get a one-sided Karp reduction from L to L' with the algorithm B as the reduction and A the corresponding NO-instance sampler. ◀

▶ **Lemma 23.** *Suppose, for some polynomial m and problems L, L', there is a* perfect *reduction from the problem $\mathrm{OR}_m \circ L$ to L'. If this reduction is $O(m \log n)$-lossy, then there is a weak one-sided average-case Karp reduction with perfect sampling from L to L'.*

Proof of Lemma 23. Here, the reduction has error 0, and is t-lossy for $t = c'm \log n$ for some constant c'. Applying Lemma 34 with $c = c' + 2$, we get $\delta(n) = 1 - 2^{-(c' \log n + 3)}$, which is less than $1 - 1/n^{c-1}$ for large enough n. We get a weak-perfect one-sided reduction from L to L' with the algorithm B as the reduction and A as the corresponding NO-instance sampler. ◀

Closely along the lines of Drucker [13], we show the following analogue of Lemma 34 for Majority reductions, and use it to prove Lemma 24. We defer the proof of this lemma to Appendix B.1.

▶ **Lemma 35.** *Suppose, for some polynomials $m : \mathbb{N} \to \mathbb{N}$ and $t : \mathbb{N} \to \mathbb{R}$, a problem L has a t-lossy MAJ_m-reduction to a problem L', and the reduction has error $\epsilon(n) < 0.5$. Let*

$$\delta(n) = \min\left\{\sqrt{\frac{(t+1)\ln 2}{m+1}}, 1 - 2^{-\frac{2t}{m+2}-3}\right\}$$

Then, for any constant c, there are polynomial-time algorithms Y, N and R such that:

- $\Pr[\mathsf{N}(1^n) \notin L'_N] \leq \epsilon$
- $\Pr[\mathsf{Y}(1^n) \notin L'_Y] \leq \epsilon$
- *For any $x \in L_N \cap \{0,1\}^n$, $\Delta(\mathsf{N}(1^n); \mathsf{R}(x)) \leq \delta(n) + n^{-c}$*
- *For any $x \in L_Y \cap \{0,1\}^n$, $\Delta(\mathsf{Y}(1^n); \mathsf{R}(x)) \leq \delta(n) + n^{-c}$*

▶ **Lemma 24.** *Suppose, for some polynomial m and problems L, L', there is a reduction from the problem $\mathrm{MAJ}_m \circ L$ to L'. If this reduction is $m/100$-lossy and $m(n) > 100$ for all large enough n, then there is a two-sided average-case Karp reduction from L to L'.*

Proof of Lemma 24. The reduction we start with has error 0.1 and is t-lossy for $t = m/100$ bits. Applying Lemma 35 with $c = 1$, we get algorithms R, Y and N with $\delta < \sqrt{\ln 2/100}$ for all large enough n. We get a two-sided Karp reduction from L to L' with the algorithm Y as the YES-instance sampler, N as the NO-instance sampler, and R as the reduction. ◀

B.1 Proof of Lemma 35

Throughout this subsection, we will be concerned with a (possibly randomized) mapping $\mathsf{F} : (\{0,1\}^n)^m \to \{0,1\}^*$ for some odd m and $n \in \mathbb{N}$. The following notation will be useful in our discussions. Suppose D_0 and D_1 are distributions over $\{0,1\}^n$. By $\mathsf{F}(D_0^k, D_1^\ell)$, we denote the execution of F on m inputs, k of which are sampled from D_0, and ℓ from D_1, and the whole set of samples is randomly permuted before being input to F. For an $x \in \{0,1\}^n$, $\mathsf{F}(D_0^k, D_1^\ell, x)$ is similar, except along with the $(k + \ell)$ samples, x is also inserted. We will use the following concept of distributional stability that is similar to the one used by Drucker [13], but specialised for pairs of distirbutions.

▶ **Definition 36.** *Let $m, n, k \in \mathbb{N}$, with $m = 2k + 1$, and $\delta \in [0, 1]$. A (possibly randomized) mapping $\mathsf{F} : (\{0,1\}^n)^m \to \{0,1\}^*$ is said to be* δ-distributionally stable *(denoted δ-DS) over a pair of distributions (D_0, D_1) over $\{0,1\}^n$ if the following holds:*

$$\mathop{\mathbb{E}}_{x \leftarrow D_0} \left[\Delta\left(\mathsf{F}(D_0^k, D_1^k, x); \mathsf{F}(D_0^{k+1}, D_1^k)\right)\right] \leq \delta$$

The proof of Lemma 35 follows from the following propositions.

▶ **Proposition 37.** *Let $m, n, k \in \mathbb{N}$, and $t \in \mathbb{R}^+$, with $m = 2k + 1$. Any (possibly randomized) mapping $\mathsf{F} : (\{0,1\}^n)^m \to \{0,1\}^*$ that is t-lossy on nm-bit inputs is δ-distributionally stable over any pair of distributions (D_0, D_1) over $\{0,1\}^n$, where $\delta = \min\left\{\sqrt{\frac{(t+1)\ln 2}{m+1}}, 1 - 2^{-\frac{2t}{m+2}-3}\right\}$.*

▶ **Proposition 38.** *Let $m, n, k \in \mathbb{N}$, with $m = 2k + 1$. Suppose $\mathsf{F} : (\{0,1\}^n)^m \to \{0,1\}^*$ is δ-DS over all pairs of distributions over $\{0,1\}^n$. For any pair of disjoint sets $S_0, S_1 \subseteq \{0,1\}^n$, any $\ell > 0$, and any $\nu \in (0, 1)$, there exists a distibution K over $S_0^\ell \times S_1^\ell$ such that:*

- *K is samplable in time $\mathsf{poly}(n, \ell, 1/\nu^2)$*
- *For any $x \in S_0$,*

$$\mathop{\mathbb{E}}_{(T_0,T_1)\leftarrow K} \left[\Delta\left(\mathsf{F}(U_{T_0}^k, U_{T_1}^k, x); \mathsf{F}(U_{T_0}^{k+1}, U_{T_1}^k)\right)\right] \leq \delta + 2m/\ell + \nu$$

- *For any $x \in S_1$,*

$$\mathop{\mathbb{E}}_{(T_0,T_1)\leftarrow K} \left[\Delta\left(\mathsf{F}(U_{T_1}^k, U_{T_0}^k, x); \mathsf{F}(U_{T_1}^{k+1}, U_{T_0}^k)\right)\right] \leq \delta + 2m/\ell + \nu$$

where U_T is the uniform distribution over the multiset T.

Proof of Lemma 35. Given a reduction A from $\mathrm{MAJ}_m \circ L$ to L' that is t-lossy and the constant c, we construct the algorithms Y, N, and R as required by the lemma. Fix a value of n and, for convenience, denote $m(n)$ and $t(n)$ by just m $(= 2k + 1)$ and t.

First, we apply Proposition 37 to A, which tells us that it is δ-DS over any pair of distributions over $\{0,1\}^n$, with δ as in the statement. This lets us apply Proposition 38 taking the sets S_0 and S_1 to be $L_N \cap \{0,1\}^n$ and $L_Y \cap \{0,1\}^n$, respectively, and with $\ell = 4mn^c$ and $\nu = 1/2n^c$ there. Proposition 38 then gives us a samplable distribution K over $L_N^\ell \times L_Y^\ell$.

The algorithm N, on input 1^n, samples (T_0, T_1) from the K obtained as above, and outputs a sample from $\mathsf{A}(U_{T_0}^{k+1}, U_{T_1}^k)$. The algorithm R is the same, except it outputs $\mathsf{A}(U_{T_1}^{k+1}, U_{T_0}^k)$. The algorithm R, on input x, similarly samples (T_0, T_1) and outputs $\mathsf{A}(U_{T_0}^k, U_{T_1}^k, x)$.

That the outputs of $\mathsf{N}(1^n)$ and $\mathsf{Y}(1^n)$ are contained in L'_N except with error ϵ follows from the fact that $T_0 \subseteq L_N$, $T_1 \subseteq L_Y$, and A has error at most ϵ.

Next, for any $x \in L_N \cap \{0,1\}^n$, we are interested in the following distance:

$$\begin{aligned}
\Delta\left(\mathsf{N}(1^n); \mathsf{R}(x)\right) &= \Delta\left(\mathsf{A}(U_{T_0}^{k+1}, U_{T_1}^k); \mathsf{A}(U_{T_0}^k, U_{T_1}^k, x)\right) \\
&\leq \Delta\left((T_0, T_1, \mathsf{A}(U_{T_0}^{k+1}, U_{T_1}^k)); (T_0, T_1, \mathsf{A}(U_{T_0}^k, U_{T_1}^k, x))\right) \\
&= \mathop{\mathbb{E}}_{(T_0,T_1)\leftarrow K} \left[\Delta\left(\mathsf{A}(U_{T_0}^{k+1}, U_{T_1}^k); \mathsf{A}(U_{T_0}^k, U_{T_1}^k, x)\right)\right] \\
&\leq \delta + 2m/\ell + \nu = \delta + 3/4n^c
\end{aligned}$$

where the first inequality follows from the data processing inequality, and the second from Proposition 38. The analogous guarantee for $x \in L_Y \cap \{0,1\}^n$ and $\mathsf{Y}(1^n)$ is shown in the same way. This proves the lemma. ◀

Proof of Proposition 37. We prove this by reduction to the simpler distributional stability lemma proved by Drucker [13]. Recall that $\mathsf{F}(D_0^{k+1}, D_1^k)$ is computed by sampling $x_1, \ldots, x_{k+1} \leftarrow D_0$ and $y_1, \ldots y_k \leftarrow D_1$, and a random permutation π over $[2k+1]$, and then computing the output as $\mathsf{F}(\pi(x_1, \ldots, x_{k+1}, y_1, \ldots, y_k))$. Alternatively, we can write this as the random variable $F(\Pi(\overline{X}, \overline{Y}))$. Similarly, we can write $\mathsf{F}(D_0^k, D_1^k, x)$ as $\mathsf{F}(\Pi(x, \overline{X'}, \overline{Y'}))$. The distance we are interested in is now:

$$\Delta\left(\mathsf{F}(\Pi(\overline{X}, \overline{Y})); \mathsf{F}(\Pi(x, \overline{X'}, \overline{Y}))\right)$$

We split the process of sampling Π into two parts. First, we sample the part of the permutation that maps the last k inputs (corresponding to $\overline{Y}$) to get an intermediate random variable $\widehat{\Pi}$, which is uniform over the set of permutations that all map their last k inputs to the same places, and then sample the final permutation π from this variable. We have:

$$\begin{aligned}\Delta\left(\mathsf{F}(\Pi(\overline{X}, \overline{Y})); \mathsf{F}(\Pi(x, \overline{X'}, \overline{Y}))\right) &\leq \Delta\left((\widehat{\Pi}, \mathsf{F}(\Pi(\overline{X}, \overline{Y}))); (\widehat{\Pi}, \mathsf{F}(\Pi(x, \overline{X'}, \overline{Y})))\right) \\ &= \mathop{\mathbb{E}}_{\widehat{\pi} \leftarrow \widehat{\Pi}} \left[\Delta\left(\mathsf{F}(\Pi_{\widehat{\pi}}(\overline{X}, \overline{Y})); \mathsf{F}(\Pi_{\widehat{\pi}}(x, \overline{X'}, \overline{Y}))\right)\right]\end{aligned}$$

where the first inequality is by the data processing inequality, and in the last expectation $\Pi_{\widehat{\Pi}}$ represents Π sampled conditioned on the last k inputs being mapped according to $\widehat{\pi}$.

For any $\widehat{\pi}$ sampled from $\widehat{\Pi}$, define the mapping $\mathsf{G}_{\widehat{\pi}}$ on input $(x_1, \ldots, x_{k+1})$ as the one obtained by sampling $(y_1, \ldots, y_k) \leftarrow \overline{Y}$, and running F with the y_i's in the positions assigned by $\widehat{\pi}$, and the x_i's in lexicographic order in the rest. We can now apply [13, Lemma 6.2], which is paraphrased below, to the quantity inside the first expectation with $\mathsf{G}_{\widehat{\pi}}$ as the mapping to get the proposition.

▶ **Proposition 39** ([13, Lemma 6.2])**.** *Let $m, n \in \mathbb{N}$, and $t \in \mathbb{R}^+$. Any (possibly randomized) mapping $\mathsf{F} : (\{0,1\}^n)^m \to \{0,1\}^*$ that is t-lossy on nm-bit inputs satisfies the following for any distribution D over $\{0,1\}^n$:*

$$\mathop{\mathbb{E}}_{x \leftarrow D}\left[\Delta\left(\mathsf{F}(D^k, x); \mathsf{F}(D^{k+1})\right)\right] \leq \delta$$

where $\delta = \min\left\{\sqrt{\frac{\ln 2}{2} \cdot \frac{t+1}{m}}, 1 - 2^{-\frac{t}{m}-3}\right\}$.

While the lemma in [13] was stated for compression as measured by output length, it may be verified that its proof there only uses lossiness. ◀

In order to prove Proposition 38, we need the following proposition that is proven in the same way as [13, Lemma 6.3].

▶ **Proposition 40.** *Suppose F is δ-DS over all pairs of distributions over $\{0,1\}^n$. For a pair of distributions (D_0, D_1), denote by $\widehat{D}_{0,\ell}$ (respectively $\widehat{D}_{1,\ell}$) the empirical distribution formed by taking ℓ samples from D_0 (respectively D_1). (Note that these are themselves random variables.) Then,*

$$\mathop{\mathbb{E}}_{\substack{\widehat{D}_{0,\ell}, \widehat{D}_{1,\ell} \\ x \leftarrow D_0}}\left[\Delta\left(\mathsf{F}(\widehat{D}_{0,\ell}^k, \widehat{D}_{1,\ell}^k, x); \mathsf{F}(\widehat{D}_{0,\ell}^{k+1}, \widehat{D}_{1,\ell}^k)\right)\right] \leq \delta + 2m/\ell$$

Proof of Proposition 38. We proceed by considering the following two-player zero-sum game:

- Player 1 chooses a pair of multisets $T_0 \subseteq S_0$ and $T_1 \subseteq S_1$, both of size ℓ
- Player 2 chooses a string $x \in S_0 \cup S_1$

- If $x \in S_0$, Player 2 gets a payoff of $\Delta\left(\mathsf{F}(U^k_{T_0}, U^k_{T_1}, x); \mathsf{F}(U^{k+1}_{T_0}, U^k_{T_1})\right)$, and if $x \in S_1$, it gets a payoff of $\Delta\left(\mathsf{F}(U^k_{T_1}, U^k_{T_0}, x); \mathsf{F}(U^{k+1}_{T_1}, U^k_{T_0})\right)$

The rest of the proof follows that of [13, Lemmas 6.4 and 6.5]. For any randomized strategy of Player 2, which is a distribution X over $S_0 \cup S_1$, consider the randomized strategy of Player 1 (T_0, T_1) where each element of T_0 (respectively T_1) is sampled from the distribution X restricted to S_0 (respectively S_1), denoted $X|_{S_0}$ (denoted $X|_{S_1}$). In the case where X is not supported in S_0 (respectively S_1), the set T_0 is chosen arbitrarily from S_0 (respectively T_1 from S_1). The payoff of this strategy for Player 2 is calculated as follows:

$$\begin{aligned}\Pr_{x \leftarrow X}\left[x \in S_0\right] \mathop{\mathbb{E}}_{x \leftarrow X|_{S_0}, T_0, T_1} \left[\Delta\left(\mathsf{F}(U^k_{T_0}, U^k_{T_1}, x); \mathsf{F}(U^{k+1}_{T_0}, U^k_{T_1})\right)\right] \\ + \Pr_{x \leftarrow X}\left[x \in S_1\right] \mathop{\mathbb{E}}_{x \leftarrow X|_{S_1}, T_0, T_1} \left[\Delta\left(\mathsf{F}(U^k_{T_1}, U^k_{T_0}, x); \mathsf{F}(U^{k+1}_{T_1}, U^k_{T_0})\right)\right]\end{aligned}$$

Noting that the elements of T_0 and T_1 are sampled from $X|_{S_0}$ and $X|_{S_1}$, respectively, and that F is δ-DS over all pairs of distributions over $\{0,1\}^n$, Proposition 40 implies that both the expectations above are at most $\delta + 2m/\ell$.

Thus, for any strategy of Player 2, there is a strategy of Player 1 such that Player 2's payoff is at most $\delta + 2m/\ell$. By the minimax theorem, there is a randomized strategy of Player 1 – a distribution over multisets (T_0, T_1) – such that for every move $x \in S_0 \cup S_1$ of Player 2, its payoff is at most $\delta + 2m/\ell$. Further, by the results of Lipton and Young [22, Theorem 2], there is a randomized strategy of Player 1 that corresponds to uniformly sampling from a set of pure strategies of size $O(n/\nu^2)$ and restricts Player 2's payoff to $\delta + 2m/\ell + \nu$. This proves the proposition. ◀

Affine Determinant Programs: A Framework for Obfuscation and Witness Encryption

James Bartusek
UC Berkeley, CA, USA
bartusek.james@gmail.com

Yuval Ishai
Technion – Israel Institute of Technology, Haifa, Israel
yuvali@cs.technion.ac.il

Aayush Jain
UCLA, Los Angeles, CA, USA
aayushjain@cs.ucla.edu

Fermi Ma
Princeton University, NJ, USA
NTT Research, Palo Alto, CA, USA
fermima@alum.mit.edu

Amit Sahai
UCLA, Los Angeles, CA, USA
sahai@cs.ucla.edu

Mark Zhandry
Princeton University, NJ, USA
NTT Research, Palo Alto, CA, USA
mzhandry@princeton.edu

Abstract

An *affine determinant program* $\mathsf{ADP} : \{0,1\}^n \to \{0,1\}$ is specified by a tuple $(\mathbf{A}, \mathbf{B}_1, \ldots, \mathbf{B}_n)$ of square matrices over $\mathbb{F}_q$ and a function $\mathsf{Eval} : \mathbb{F}_q \to \{0,1\}$, and evaluated on $\mathbf{x} \in \{0,1\}^n$ by computing $\mathsf{Eval}(\det(\mathbf{A} + \sum_{i\in[n]} x_i \mathbf{B}_i))$.

In this work, we suggest ADPs as a new framework for building general-purpose obfuscation and witness encryption. We provide evidence to suggest that constructions following our ADP-based framework may one day yield secure, practically feasible obfuscation.

As a proof-of-concept, we give a candidate ADP-based construction of indistinguishability obfuscation ($i\mathcal{O}$) for all circuits along with a simple witness encryption candidate. We provide cryptanalysis demonstrating that our schemes resist several potential attacks, and leave further cryptanalysis to future work. Lastly, we explore practically feasible applications of our witness encryption candidate, such as public-key encryption with near-optimal key generation.

2012 ACM Subject Classification Theory of computation → Cryptographic primitives

Keywords and phrases Obfuscation, Witness Encryption

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.82

Funding YI was supported by ERC Project NTSC (742754), NSF-BSF grant 2015782, BSF grant 2018393, and a joint Israel-India grant. MZ, FM, AS, and AJ were supported by the Defense Advanced Research Projects Agency through the ARL under Contract W911NF-15-C- 0205. MZ and FM were also supported under NSF grant 1616442. AS and AJ were also supported in part by a DARPA/ARL SAFEWARE award, NSF Frontier Award 1413955, and NSF grant 1619348, BSF grant 2012378, a Xerox Faculty Research Award, a Google Faculty Research Award, an equipment grant from Intel, and an Okawa Foundation Research Grant. AJ was also supported by a Google PhD Fellowship in Privacy and Security. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense, the National Science Foundation, the U.S. Government or Google.

11th Innovations in Theoretical Computer Science Conference (ITCS 2020).
Editor: Thomas Vidick; Article No. 82; pp. 82:1–82:39

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

Program obfuscation "scrambles" a program, preserving functionality while hiding implementation details. A theoretical study of program obfuscation was initiated by Barak et al. [6], who demonstrated that the natural notion of virtual black-box (VBB) obfuscation is impossible to achieve for all circuits. This initially led to skepticism that general cryptographically useful program obfuscation might not be realizable.

The situation changed several years ago when Garg et al. [17] gave the first candidate construction of *indistinguishability obfuscation* – a weaker notion not subject to the impossibility – and, along with subsequent work [32], showed how to use this weaker notion in a multitude of cryptographically useful ways. In fact, today indistinguishability obfuscation (iO) is widely regarded as "crypto complete," capable of achieving essentially any cryptographic task.[1]

Initially all candidate obfuscators required cryptographic multilinear maps [16, 13, 20], a powerful new mathematical tool. Very recently, two new lines of work have emerged on constructing obfuscation without multilinear maps. The first line of work builds upon the connection between obfuscation and functional encryption (see [1] and the references therein); the second builds on a number of new mathematical methods [21]. While these new directions expand the set of methodologies for achieving general-purpose obfuscation, the set of viable approaches for building general-purpose obfuscation remains extremely limited.

We view the situation as being reminiscent of the early days of public key cryptography, which also required new mathematical tools for the time. Some, such as Diffie-Hellman and RSA, have withstood the test of time, whereas others, such as the Merkle–Hellman knapsack cryptosystem, were ultimately found to be insecure. But even the failures yielded interesting insights, such as new cryptanalysis techniques. Therefore, given the importance of obfuscation, we believe it is crucial to investigate many potential lines of inquiry for realizing obfuscation. Furthermore, we believe it is particularly important to emphasize *simplicity* in our search for new approaches, since even successful cryptanalyses of simple proposals are likely to inform the scientific quest for secure and efficient obfuscation. (This perspective is part of a larger research agenda exploring new sources of hardness in cryptography [31].)

This Work

In this work, we propose a new framework for building obfuscation through *affine determinant programs* (ADP). An ADP consists of a tuple of square matrices $(\mathbf{A}, \mathbf{B}_1, \ldots, \mathbf{B}_n)$ over $\mathbb{F}_p$, where evaluation on an input $\mathbf{x} \in \{0,1\}^n$ involves computing the affine combination $\mathbf{A} + \sum_{i \in [n]} x_i \mathbf{B}_i$ and taking the determinant. While ADPs have appeared before in the literature (e.g. to build randomized encodings [28]), to the best of our knowledge no prior work has considered their application to general obfuscation. We believe ADPs are a promising route toward obfuscation for several reasons:

- ADPs are conceptually very simple.
- We can interpret the recent work of Bartusek et al. [8] as an ADP for conjunction obfuscation, with provable security under the LPN assumption. Thus, ADPs certainly provide security for obfuscation in some situations.
- As we will show, ADPs may offer a route toward implementable applications.

[1] To be slightly more precise, building cryptography from iO also requires one-way functions.

Therefore, we initiate the study of ADPs for obfuscation, giving the following results:

- **A New Framework for Building Obfuscation.** We suggest ADPs as a route toward building obfuscation without multilinear maps. To facilitate this approach, we provide a number of techniques for protecting ADPs against attacks, and study the security of these techniques by developing new cryptanalysis techniques.
- **A Candidate Witness Encryption Scheme.** We examine a special case of obfuscation, called witness encryption, and give a very simple candidate construction. As an application, we use our scheme to build a public key encryption scheme with low-complexity key generation. Our approach yields for the first time *implementable* applications of witness encryption.
- **Towards Candidate iO Constructions.** We then turn to the much more difficult problem of constructing full indistinguishability obfuscation. We develop some initial ideas, including cryptanalytic methods, leading to an initial candidate construction.

1.1 Technical Overview

One of the main approaches to general-purpose program obfuscation, pioneered in [17], can be seen as following a particular recipe:

1. Construct an algebraic randomized encoding for the input program, which can be seen as offering a sort of "one-time" security.
2. Place the algebraic encoding "in the exponent" of a multilinear map, using the multilinear map operations to evaluate the encoding.

In most of the literature, the randomized encoding used is a matrix branching program, which can be obtained using Barrington's theorem and Kilian re-randomization. Another common example due to [37] converts a circuit into an algebraic circuit.

Affine Determinant Programs

In this work, we consider another variant of randomized encodings from the literature (though not yet considered in the *obfuscation* literature) which we call *Affine Determinant Programs* (ADPs). Here, the program consists of a matrix M whose entries are affine functions mapping inputs x to a field $\mathbb{F}_p$. We will denote by $M(x)$ the entry-wise evaluation of M on input x. There is also a fixed evaluation function $\mathsf{Eval} : \mathbb{F} \to \{0, 1\}$. To evaluate the program on input x, we simply compute $\mathsf{Eval}(\det(M(x))$.

We note that, just like Barrington's theorem or the algebraic circuit approach to constructing algebraic randomized encodings, ADPs can also plausibly be used in conjunction with multilinear maps (and appropriate re-randomization) to yield secure obfuscation[2]. However, the central question we consider in this work is:

Can we obfuscate with ADP*s in the clear, without using multilinear maps?*

Witness Encryption – An Initial Idea

As a starting point, we consider the easier task of constructing a secure *witness encryption* [19] scheme. In witness encryption, a ciphertext c is encrypted with respect to an instance x of some NP language L. The ciphertext may be decrypted using any witness π that attests

[2] There is a potential issue that the standard multilinear map interface does not allow for efficiently computing determinants over encoded values. However, the first two multilinear map candidates [16, 13] *do* allow for determinant computations.

that $x \in L$. Security stipulates that, for any false instance $x \notin L$, encryptions relative to x completely hide the message from computationally bounded adversaries. Note that the instance x is considered public.

We build a new framework for witness encryption for the NP-complete subset-sum problem, where instances x are vector/scalar pairs $(v, t) \in \mathbb{Z}^n \times \mathbb{Z}$, and witnesses are 0/1 vectors $w \in \{0,1\}^n$ such that $v \cdot w = t$. Under standard NP reductions, our construction extends to any NP language.

Testing the validity of a subset-sum witness can be trivially expressed as an ADP of dimension one: $M_{v,t}(w) = -t + v \cdot w$ is an affine function that is 0 if and only if w is a witness for (v, t). We can introduce randomization and extend this to a matrix program by choosing a random $R \in \mathbb{F}^{k \times k}$ and setting $M_{v,t}(w) = (-t + v \cdot w)R$. Then, with overwhelming probability over the choice of R, for $w \in \{0,1\}^n$, $\det(M_{v,t}(w)) = 0$ if and only if w is a valid witness.

While the above allows for testing if a witness is valid, we need a way to actually encode a message in the output. There are many ways to do this, but perhaps the simplest is to encrypt 0 by sampling the distribution above, and encrypting 1 by sampling a uniformly random ADP. Let $M_{v,t,m}$ be the resulting ADP. A random ADP over a large field will have $\det(M(w)) \neq 0$ with high probability for *any* w. Therefore for valid witnesses w, we have that $\det(M_{v,t,m}(w)) = 0$ if and only if $m = 0$.

Witness Encryption – Resisting Attacks Through Structured "Noise"

It is not difficult to see that the idea so far is completely insecure. The attacker can learn m by, for example, choosing a w^* that satisfies $v \cdot w^* = t$, but where w^* now consists of arbitrary field elements rather than 0/1. Such a w^* can be found trivially using linear algebra. In this case, we will still have that $\det(M(w)) = 0$ if and only if $m = 0$, thus revealing m. We call attacks of this form *invalid input attacks*.

Alternatively, one could simply inspect a single entry of M and notice that in the case $m = 0$, the entry is a multiple of the (known) function $(-t + v \cdot w)$, whereas if $m = 1$ the entry is a random affine function and is most likely not a multiple of $(-t + v \cdot w)$.

To remedy these issues, we will *add* as "noise" a randomized "all-accept" ADP M_{AA}, which is an ADP that accepts every 0/1 input, but rejects (with high probability) any input that is not 0/1. The encryption of 0 will then be

$$M(w) = M_{AA}(w) + M_{v,t,m}(w).$$

Since the all-accept program rejects any input that is not 0/1, meaning $M_{AA}(w)$ is full rank, it is also the case that $M(w)$ is full rank with high probability. Therefore, our witness encryption scheme will reject any invalid inputs, just as a random ADP will. This shows that our witness encryption scheme is at least immune to invalid input attacks.

Of course, as suggested by our noise analogy above, the security of this proposal depends on how "noisy" the all-accept program M_{AA} is. In Section 5 we discuss two simple approaches, and how these approaches evade our attempts at cryptanalysis.

Applications

As suggested in [19], one application of witness encryption is to build a public key encryption scheme where public keys are extremely short and generated by a lightweight process. In particular, given a PRG G, the secret key will be a seed s and the public key will be the output $x = G(s)$. To encrypt a message m, simply witness encrypt m to the statement "x

has a pre-image under G". Correctness is immediate, and security follows from a simple two-step hybrid: first replace x with a random string, which with overwhelming probability will not be in the image of G. Then invoke witness encryption security to show that m is hidden. Under a stronger assumption of *extractable* witness encryption due to [23], G can even be taken to be an arbitrary one-way function.

We show how to optimize our witness encryption candidate for use with Goldreich's one-way function, which is a very simple one-way function where every output bit depends on only 5 input bits. Thus, public-key generation is *linear-time* in the secret key length. For a plausible setting of parameters, we obtain 300-bit public keys, and 50 megabyte ciphertexts. We note that while our ciphertexts are too large for any practical use, they are well within the realm of implementability. In contrast, existing approaches to building witness encryption using multilinear maps are extremely impractical and their concrete efficiency has never even been estimated, to the best of our knowledge.

Indistinguishability Obfuscation

We now turn to the task of using ADPs to build general-purpose indistinguishability obfuscation ($i\mathcal{O}$). We note that this case is much more challenging than witness encryption. First, there is the obvious question of converting general computations into ADPs. Second, the security requirements for witness encryption are much weaker: it only needs security to hold in the evasive setting, where the adversary cannot find accepting inputs, and the adversary is allowed to know almost the entire program, save for a single message bit.

Fortunately, there are multiple ways to convert a circuit in NC^1 into an ADP, outlined in Section 4, and $i\mathcal{O}$ for NC^1 can be bootstrapped into $i\mathcal{O}$ for all circuits following known techniques [17]. However, simply taking such a program and adding an all-accept program as in our witness encryption construction is insufficient, as doing so leads to various attacks, which we outline in Section 9.

As mentioned earlier, the common approach starting with [17] to obfuscating circuits in NC^1 first applies Barrington's theorem [7] to obtain a representation of the circuit as a polynomial-size branching program. Then, the branching program is represented as a *matrix branching program*, and the matrix entries are encoded in the exponent of a multilinear map.

In this work, we still make use of branching programs as the underlying computational model, but instead appeal to the works of [28, 4], which give a representation of any deterministic branching program as an ADP $(\mathbf{A}, \mathbf{B}_1, \ldots, \mathbf{B}_n)$. In particular, this representation has the property that for any binary input $\mathbf{x} \in \{0,1\}^n$, $\det(\mathbf{A} + \sum_i x_i \mathbf{B}_i) = 1$ if $\mathbf{x}$ induces an accepting path in the branching program, and $\det(\mathbf{A} + \sum_i x_i \mathbf{B}_i) = 0$ otherwise.

We argue that this ADP representation of branching programs has several qualitative advantages over the matrix branching program approach. Notably, the earlier matrix branching program approach gives rise to so-called "mixed input" attacks, since multiple matrices are associated with each input bit. In contrast, the ADP encoding associates a single matrix with each input bit.

In this work, we initiate a systematic study of various randomization strategies that can be applied to ADPs that encode branching programs, in the hope that such safeguards will lead to a secure obfuscation scheme. As a first attempt, we left- and right-randomize with matrices $\mathbf{R}, \mathbf{S}$ over $\mathbb{F}_p$ such that $\det(\mathbf{R}) \cdot \det(\mathbf{S}) = 1$, preserving the correctness property of the ADP, while scrambling the individual matrix entries. However, this safeguard is not sufficient, due to the existence of "polynomial extension attacks," discussed in Section 9.1. Thus, we consider methods of scrambling the underlying ADP encoding first, before applying the left and right randomization as a final step.

In particular, we make use of the fact that on honest evaluation, the determinant will always be in $\{0, 1\}$. This motivates the possibility of adding random *even-valued* noise to the matrices in the ADP, which shifts the determinant by an even value on every input. Correctness is then preserved by simply having the evaluator compute the parity of the determinant. Parameters must be set carefully so that the size of the field over which these determinant calculations are being carried out is large enough so that determinants on honest inputs do not wrap around the modulus. This is discussed in detail in Section 8.1.

Unfortunately, as we show in Section 9.3, this even-valued error does not yet give a secure obfuscation scheme. In particular, there exist attacks on $i\mathcal{O}$ that use the fact that the adversary may know the underlying structure of the branching program it is attacking to learn non-trivial information about the random even-valued error. This in turn motivates the need to inject entropy into the structure of the branching program itself, before encoding it as an ADP and applying the safeguards described above. We term this process "random local substitution," where each individual "step" of the branching program is randomized, and discuss one concrete method of doing this in Section 8.1.1. However, we stress that there is a vast space of transformations to explore here, and our proposal is only the first of many simple possibilities to explore.

In summary, we present the following generic recipe for obfuscating a deterministic branching program BP:

- Apply random local *functionality-preserving* transformations to BP, producing BP'.
- Represent BP' as an ADP, following [28, 4].
- Add random even-valued error to each matrix in ADP, fixing the evaluation function to compute the parity of the determinant.
- Left- and right-randomize with matrices $\mathbf{R}, \mathbf{S}$ such that $\det(\mathbf{R}) \cdot \det(\mathbf{S}) = 1$.

In the body, we give an instantiation of the above recipe, resulting in a candidate $i\mathcal{O}$ scheme for all circuits (noting that it suffices to obfuscate the class NC^1, which can be simulated by polynomial-size branching programs). However, we view the recipe itself as the main contribution of this section, and hope that it will motivate other candidates and more cryptanalysis, with the eventual goal of obtaining efficient and secure obfuscation without the use of multilinear maps.

2 Preliminaries

Let $\mathbb{Z}, \mathbb{N}$ be the set of integers and positive integers respectively. For $n \in \mathbb{N}$, we let $[n]$ denote the set $\{1, \ldots, n\}$. For $p \in \mathbb{N}$, denote $\mathbb{Z}/p\mathbb{Z}$ by $\mathbb{Z}_p$, and denote the finite field of order p by $\mathbb{F}_p$. A vector $\mathbf{v} \in \mathbb{F}_p^n$ (represented in column form by default) is written as a bold lower-case letter and we denote its ith element by $v_i \in \mathbb{F}_p$. A matrix $\mathbf{A} \in \mathbb{F}_p^{n \times m}$ is written as a bold capital letter and we denote the entry at position (i, j) by $(\mathbf{A})_{i,j}$. For any set of matrices $\mathbf{A}_1, \ldots, \mathbf{A}_n$ of potentially varying dimensions, let $\mathsf{diag}(\mathbf{A}_1, \ldots, \mathbf{A}_n)$ be the block diagonal matrix with the $\mathbf{A}_i$ on the diagonal, and zeros elsewhere.

We use the usual Landau notations. A function $f(n)$ is said to be negligible if it is $n^{-\omega(1)}$ and we denote it by $f(n) := \mathsf{negl}(n)$. A probability $p(n)$ is said to be overwhelming if it is $1 - n^{-\omega(1)}$. We write $D_1 \approx_S D_2$ if there exists a negligible function $\mathsf{negl}(n)$ such that the statistical distance between D_1 and D_2, denoted $SD(D_1, D_2)$, is bounded above by $\mathsf{negl}(n)$. Recall for any two distribution D_1 and D_2 taking values in some set $\mathcal{S}$ the statistical distance or $SD(D_1, D_2)$ is defined as:

$$SD(D_1, D_2) = 1/2 \sum_{x \in \mathcal{S}} |\Pr[D_1 = x] - \Pr[D_2 = x]|$$

We write $D_1 \approx_C D_2$ if no computationally-bounded adversary can distinguish between D_1 and D_2 except with advantage $\mathsf{negl}(n)$.

2.1 Randomized Encodings

A randomized encoding of the function $f(x)$ is a function $\hat{f}(x,r)$ such that a sample from the output distribution of $\hat{f}(x,r)$ for a uniformly chosen r reveals $f(x)$ and no additional information about x. We formalize and generalize this below.

▶ **Definition 1** (Randomized Encodings [5]). *Let $f : \{0,1\}^n \to \{0,1\}^\ell$ be a function. We say that $\hat{f} : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^s$ is a δ-correct, ϵ-private randomized encoding of f, if it satisfies the following:*

- *δ-**correctness**. There exists an algorithm B, called a decoder, such that for any input $\mathbf{x} \in \{0,1\}^n$,*

$$\Pr[B(\hat{f}(x, U_m)) \neq f(x)] \leq \delta.$$

- *ϵ-**privacy**. There exists a randomized simulator algorithm Sim such that for any $\mathbf{x} \in \{0,1\}^n$,*

$$SD(\mathsf{Sim}(f(x)), \hat{f}(x, U_m)) \leq \epsilon.$$

2.2 Witness Encryption

The following definition is taken almost verbatim from [19].

▶ **Definition 2.** *A witness encryption scheme for an NP language L (with corresponding witness relation R) consists of the following two polynomial-time algorithms:*

- *$\mathsf{Encrypt}(1^\lambda, x, m)$ takes as input a security parameter 1^λ, an unbounded-length string x, and a message $m \in \{0,1\}$, and outputs a ciphertext ct.*
- *$\mathsf{Decrypt}(ct, w)$ takes as input a ciphertext ct and an unbounded-length string w, and outputs a message m or the symbol $\perp$.*

These algorithms satisfy the following two conditions:

- ***Correctness.** For any security parameter λ, for any $m \in \{0,1\}$, and for any $x \in L$ such that $R(x,w)$ holds, we have that*

$$\Pr[\mathsf{Decrypt}(\mathsf{Encrypt}(1^\lambda, x, m), w) = m] = 1.$$

- ***Soundness Security.** For any PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for any $x \notin L$, we have:*

$$|\Pr[\mathcal{A}(\mathsf{Encrypt}(1^\lambda, x, 0)) = 1] - \Pr[\mathcal{A}(\mathsf{Encrypt}(1^\lambda, x, 1)) = 1]| < \mathsf{negl}(\lambda).$$

We also consider an "extractable" version of witness encryption, introduced by [23]. In extractable witness encryption, if an adversary that can break semantic security for an instance x, an extractor can extract the witness for x. Formally, in an extractable witness encryption, the following definition (taken verbatim from [23]) replaces the standard soundness security notion.

- **Extractable Security.** A witness encryption scheme for a language $L \in \mathsf{NP}$ is secure if for all PPT adversary $\mathcal{A}$ and all polynomials q, there exists a PPT extractor E and a polynomial $p(\cdot)$ such that for all auxiliary inputs z and for all $x \in \{0,1\}^*$, the following holds:

$$\Pr[m \leftarrow \{0,1\}; ct \leftarrow \mathsf{WE.Encrypt}(1^\lambda, x, m) : \mathcal{A}(x, ct, z) = b] \geq \frac{1}{2} + \frac{1}{q(|x|)}$$
$$\implies \Pr[\mathsf{E}(x, z) = w : (x, w) \in R_L] \geq \frac{1}{p(|x|)}.$$

2.3 Indistinguishability Obfuscation

▶ **Definition 3** (Indistinguishability Obfuscator [6]). *A uniform PPT machine $i\mathcal{O}$ is an indistinguishability obfuscator for a circuit class $\{\mathcal{C}_\lambda\}$ if the following conditions are satisfied:*

- *(Strong Functionality Preservation) For all security parameters $\lambda \in \mathbb{N}$, for all $C \in \mathcal{C}_\lambda$,*

$$\Pr_{C' \leftarrow i\mathcal{O}(\lambda, C)}[\forall x,\ C'(x) = C(x)] \geq 1 - \mathsf{negl}(\lambda).$$

- *For any non-uniform PPT distinguisher D, there exists a negligible function α such that the following holds: for all $\lambda \in \mathbb{N}$, for all pairs of circuits $C_0, C_1 \in \mathcal{C}_\lambda$, we have that if $C_0(x) = C_1(x)$ for all inputs x and $|C_0| = |C_1|$ (where $|C|$ denotes the size of a circuit), then*

$$|\Pr[D(i\mathcal{O}(\lambda, C_0)) = 1] - \Pr[D(i\mathcal{O}(\lambda, C_1)) = 1]| \leq \alpha(\lambda).$$

3 Affine Determinant Programs: Syntax and Definitions

An *affine determinant program* $(\mathsf{ADP}) : \{0,1\}^n \to \{0,1\}$ is parameterized by an input length n, a width k, and a finite field $\mathbb{F}_p$. An ADP is comprised of an affine function $\mathsf{M} : \{0,1\}^n \to \mathbb{F}_p^{k \times k}$ along with an evaluation function $\mathsf{Eval} : \mathbb{F}_p \to \{0,1\}$. The affine function M is specified by an $(n+1)$-tuple of $k \times k$ matrices $\mathsf{M} = (\mathbf{A}, \mathbf{B}_1, \ldots, \mathbf{B}_n)$ over $\mathbb{F}_p$ so that

$$\mathsf{M}(\mathbf{x}) := \mathbf{A} + \sum_{i \in [n]} x_i \mathbf{B}_i.$$

On input $\mathbf{x} \in \{0,1\}^n$, $\mathsf{ADP}_{\mathsf{M},\mathsf{Eval}}(\mathbf{x})$ is computed as

$$\mathsf{Eval}(\det(\mathsf{M}(\mathbf{x}))).$$

We will typically use one of the following Eval functions.

- $\mathsf{Eval}_0(y) = \begin{cases} 1 & \text{if } y = 0 \\ 0 & \text{if } y \neq 0 \end{cases}$.
- $\mathsf{Eval}_{\neq 0}(y) = \begin{cases} 1 & \text{if } y \neq 0 \\ 0 & \text{if } y = 0 \end{cases}$.
- $\mathsf{Eval}_{\text{parity}}(y) = y \bmod 2$.

3.1 Encoding Functions as ADPs

We will consider a randomized procedure ADP-Encode which takes as input a description $\langle f \rangle$ of a function $f : \{0,1\}^n \to \{0,1\}$ and produces $\mathsf{ADP}_{\mathsf{M},\mathsf{Eval}}$ such that $\mathsf{ADP}_{\mathsf{M},\mathsf{Eval}} \equiv f$. Formally, the algorithm works as follows.

- $\mathsf{ADP\text{-}Encode}(1^\lambda, \langle f \rangle)$: This algorithm takes in a security parameter 1^λ and a canonical description $\langle f \rangle$ of a function $f \in \mathcal{F}_n$. It outputs width-k $\mathsf{ADP}_{\mathsf{M},\mathsf{Eval}}$ over $\mathbb{F}_p$ where $k = \mathsf{poly}(\lambda, |\langle f \rangle|)$ and $p = 2^{\mathsf{poly}(\lambda)}$.

In this work, Eval will be fixed as part of the description of ADP-Encode and will be independent of the particular input $\langle f \rangle$. We will therefore view ADP-Encode as an algorithm outputting $\mathsf{M} = (\mathbf{A}, \mathbf{B}_1, \ldots, \mathbf{B}_n)$ where $\mathbf{A}, \mathbf{B}_1, \ldots, \mathbf{B}_n \in \mathbb{F}_p^{k \times k}$.

3.2 One-time Security

We say that ADP-Encode satisfies *one-time security* if the matrix $\mathsf{M}(\mathbf{x})$ can be simulated given only $f(x)$.

▶ **Definition 4** (One-time Security)**.** ADP-Encode *is* one-time secure *for* $\mathcal{F}_n$ *if there exists a PPT simulator* Sim *such that for any* $f \in \mathcal{F}_n$ *and any* $\mathbf{x} \in \{0,1\}^n$*,*

$$\mathsf{M}(\mathbf{x}) \approx_S \mathsf{Sim}(f(x))$$

where $\mathsf{M} \leftarrow \mathsf{ADP\text{-}Encode}(1^\lambda, \langle f \rangle)$.

We stress that while $\mathsf{M} = (\mathbf{A}, \mathbf{B}_1, \ldots, \mathbf{B}_n)$, the simulator is only required to simulate a single evaluation of $\mathsf{M}(\mathbf{x}) = \mathbf{A} + \sum_{i \in [n]} x_i \mathbf{B}_i$.

▶ **Theorem 5** ([28])**.** *There exists an algorithm* ADP-Encode *which satisfies correctness and one-time security for all branching programs.*

▶ Remark 6. One-time security implies that $\mathsf{M}(\mathbf{x})$ is a randomized encoding [27] of $f(x)$.

Achieving One-time Security Generically

When the evaluation function is Eval_0 or $\mathsf{Eval}_{\neq 0}$, there is a simple generic transformation that turns any functionality-preserving ADP-Encode procedure into one that additionally satisfies one-time security.

We first import the following lemma from [26].

▶ **Lemma 7** ([26])**.** *For any fixed* $\mathbf{A} \in \mathbb{F}_q^{k \times k}$ *and uniformly random invertible* $\mathbf{R}, \mathbf{S} \leftarrow \mathbb{F}_q^{k \times k}$*, the distribution of* $\mathbf{R} \cdot \mathbf{A} \cdot \mathbf{S}$ *depends only on* $\mathrm{rank}(\mathbf{A})$.

Let ADP-Encode be an encoding procedure which uses Eval_0 or $\mathsf{Eval}_{\neq 0}$, and outputs

$$\mathsf{M} = (\mathbf{A}, \mathbf{B}_1, \ldots, \mathbf{B}_n) \in (\mathbb{F}_p^{k \times k})^{n+1}.$$

▷ Claim 8 (One-Time Security for $\mathsf{Eval}_0, \mathsf{Eval}_{\neq 0}$). Suppose ADP-Encode has the property that when $\det(\mathsf{M}(\mathbf{x})) = 0$, then $\mathrm{rank}(\mathsf{M}(\mathbf{x})) = k'$ for some fixed $k' < k$. Furthermore, ADP-Encode uses Eval_0 or $\mathsf{Eval}_{\neq 0}$. Let $\mathsf{ADP\text{-}Encode}'$ be an algorithm which runs ADP-Encode, samples uniformly random invertible $\mathbf{R}, \mathbf{S} \leftarrow \mathbb{F}_p^{k \times k}$, and outputs

$$\mathbf{R} \cdot \mathsf{M} \cdot \mathbf{S} := (\mathbf{R} \cdot \mathbf{A} \cdot \mathbf{S}, \mathbf{R} \cdot \mathbf{B}_1 \cdot \mathbf{S}, \ldots, \mathbf{R} \cdot \mathbf{B}_n \cdot \mathbf{S}).$$

Then $\mathsf{ADP\text{-}Encode}'$ has identical functionality to ADP-Encode, and is additionally one-time secure.

Proof. Identical functionality holds since $\det(\mathsf{M}(\mathbf{x})) = 0$ implies $\det(\mathbf{R} \cdot \mathsf{M}(\mathbf{x}) \cdot \mathbf{S}) = 0$, and $\det(\mathsf{M}(\mathbf{x})) \neq 0$ implies $\det(\mathbf{R} \cdot \mathsf{M}(\mathbf{x}) \cdot \mathbf{S}) \neq 0$ since $\mathbf{R}, \mathbf{S}$ are invertible.

One-time security follows from the fact that given $f(x)$, $\mathbf{R} \cdot \mathsf{M}(\mathbf{x}) \cdot \mathbf{S}$ can be simulated by either drawing a random matrix of rank k' or of rank k (by Lemma 7) ◁

3.3 $i\mathcal{O}$ Security

In contrast to one-time security, $i\mathcal{O}$ security holds even if the adversary is free to evaluate M on as many inputs $\mathbf{x}$ of its choosing. In particular, we ask that $\mathsf{M} = (\mathbf{A}, \mathbf{B}_1, \ldots, \mathbf{B}_n)$ be an $i\mathcal{O}$ of the input program f.

▶ **Definition 9.** ADP-Encode *is* $i\mathcal{O}$-secure *for* $\mathcal{F}_n$ *if for any two functions* $f_1, f_2 \in \mathcal{F}_n$ *such that* $f_1 \equiv f_2$ *and* $|\langle f_1 \rangle| = |\langle f_2 \rangle|$*, then*

$$\mathsf{M}_1 \approx_C \mathsf{M}_2$$

where $\mathsf{M}_1 \leftarrow \mathsf{ADP\text{-}Encode}(1^\lambda, \langle f_1 \rangle)$ *and* $\mathsf{M}_2 \leftarrow \mathsf{ADP\text{-}Encode}(1^\lambda, \langle f_2 \rangle)$.

4 Constructing ADPs

4.1 Obfuscating Simple Functionalities with ADPs

In this section we describe simple ADP-based obfuscations of point functions and conjunctions. These constructions have appeared before in the literature, but we recast them in the language of our ADP framework.

Point Functions

Consider the class of point functions $\{\mathcal{I}_\mathbf{y}\}_\mathbf{y}$ over $\{0,1\}^n$, where for any $\mathbf{x} \in \{0,1\}^n$, $\mathcal{I}_\mathbf{y}(\mathbf{x})$ outputs 1 if and only if $\mathbf{x} = \mathbf{y}$, and outputs 0 otherwise. There exists a simple ADP encoding scheme for point functions due to [9] where each ADP matrix is 1×1 (i.e. a scalar).

On input point $\mathbf{y}$, the ADP encoding of $\mathcal{I}_\mathbf{y}$ is generated as follows. Fix a superpolynomial-size modulus p, draw uniformly random $b_1, \ldots, b_n \leftarrow \mathbb{Z}_p$, and set

$$a = -\sum_{i \in [n] : y_i = 1} b_i.$$

The resulting scalar ADP is $\mathsf{M} = (a, b_1, \ldots, b_n)$. To evaluate M on input $\mathbf{x} \in \{0,1\}^n$, compute the scalar sum $\mathsf{M}(\mathbf{x}) = a + \sum_{i \in [n]} x_i b_i$ and output 1 if and only if the resulting scalar is 0. Correctness holds with high probability on any input since the modulus p is superpolynomial. Observe that this ADP construction fits into the more general framework of taking a determinant of an affine combination of matrices, since the determinant is simply the identity for 1×1 matrices.

If the accepting point $\mathbf{y}$ is chosen from a distribution with sufficient entropy, this ADP information-theoretically hides $\mathbf{y}$ (this follows immediately from the Leftover Hash Lemma).

Conjunctions

Point functions can be naturally extended to *conjunctions*, a more expressive class of functionalities that allow for "wildcard" positions (which we denote by $*$) indicating locations where the input bit can be either 0 or 1. Formally, a conjunction $\mathcal{C}_\mathsf{pat}$ is parameterized by a pattern string $\mathsf{pat} \in \{0, 1, *\}^n$. On input $\mathbf{x} \in \{0,1\}^n$, $\mathcal{C}_\mathsf{pat} = 1$ if and only if $\mathbf{x}$ matches pat at all indices i where $\mathsf{pat}_i \in \{0,1\}$. Bartusek et al. [8] give a simple ADP-based obfuscation for conjunctions, which we recall here.

For $\mathsf{pat} \in \{0,1,*\}^n$, the corresponding ADP will consist of square matrices of width n over $\mathbb{F}_p$ where p is a superpolynomial-size prime. We use the notation $\mathsf{colspan}(\mathbf{M})$ to denote the linear subspace spanned by the columns of $\mathbf{M}$.

1. Sample a uniformly random rank-$(n-1)$ matrix $\mathbf{L} \in \mathbb{F}_p^{n \times n}$.
2. For each index $i \in [n]$ where $\mathsf{pat}_i \in \{0,1\}$, sample a uniformly random rank-one matrix $\mathbf{B}_i \in \mathbb{F}_p^{n \times n}$.
3. For each index $i \in [n]$ where $\mathsf{pat}_i = *$, sample a uniformly random rank-one matrix $\mathbf{B}_i$ conditioned on $\mathsf{colspan}(\mathbf{B}_i) \subseteq \mathsf{colspan}(\mathbf{L})$. Since a rank-one $\mathbf{B}_i$ can be written as $\mathbf{u}_i \mathbf{v}_i^\top$, this is equivalent to sampling $\mathbf{v}_i$ uniformly at random from $\mathbb{F}_p^n$, sampling $\mathbf{w}_i$ uniformly at random from $\mathbb{F}_p^n$, and setting $\mathbf{u}_i := \mathbf{L}\mathbf{w}_i$.
4. Set $\mathbf{A} = \mathbf{L} - \sum_{i \in [n], \mathsf{pat}_i = 1} \mathbf{B}_i$, and output

$$\mathsf{M} = (\mathbf{A}, \mathbf{B}_1, \ldots, \mathbf{B}_n)$$

To evaluate the ADP M on input $\mathbf{x} \in \{0,1\}^n$, compute

$$\mathsf{M}(\mathbf{x}) = \mathbf{A} + \sum_{i \in [n] : x_i = 1} \mathbf{B}_i$$

and accept (output 1) if and only if $\det(\mathsf{M}(\mathbf{x})) = 0$.

Correctness follows since for any input $\mathbf{x}$ that agrees with pat on all 0/1 positions, $\mathsf{colspan}(\mathsf{M}(\mathbf{x})) \subseteq \mathsf{colspan}(\mathbf{L})$. Since $\mathbf{L}$ is rank-$(n-1)$, $\det(\mathsf{M}(\mathbf{x}))$ will be 0. On any input $\mathbf{x}$ that disagrees with pat on a 0/1 position, $\mathsf{M}(\mathbf{x})$ is the sum of a random rank-$(n-1)$ matrix and at least one independent random rank-one matrix, which yields a full-rank with overwhelming probability.

Bartusek et al. [8] prove that this construction statistically hides pat assuming that pat has sufficient entropy on its 0/1 entries. The entropy of pat is used to argue that $\mathbf{A} = \mathbf{L} - \sum_{i \in [n], \mathsf{pat}_i = 1} \mathbf{B}_i$ is statistically close to a uniformly random matrix. This means $\mathbf{L}$ is statistically hidden, and so for all $i \in [n]$, $\mathbf{B}_i$ is statistically close to uniformly random rank-one matrix. At this point, the distribution of $(\mathbf{A}, \mathbf{B}_1, \ldots, \mathbf{B}_n)$ is independent of pat.

4.2 ADPs for Log-depth Boolean Formulas

We now give a simple construction that directly converts any log-depth formula into a functionally equivalent ADP. In contrast to the point function and conjunction setting, we have been unable to prove that this ADP encoding achieves any meaningful notion of obfuscation security. Nevertheless, this encoding procedure will be useful for one of our candidate witness encryption constructions (see Section 5), as well as to develop general intuition for the ADP framework.

We describe the construction recursively, associating an ADP with each wire in the formula and demonstrating how to construct an output ADP given two input ADPs to either an OR gate or an AND gate.

Fix a superpolynomial-size prime p. For this ADP construction, evaluation is done by computing the determinant of $\mathsf{M}(\mathbf{x}) := \mathbf{A} + \sum_{i \in [n] : x_i = 1} \mathbf{B}_i$ over $\mathbb{F}_p$ and accepting if and only if the result is 0.

Throughout this construction, we maintain the invariant that (with overwhelming probability) on an accepting input $\mathsf{M}(\mathbf{x})$ will be rank-deficient by 1, and on a rejecting input $\mathsf{M}(\mathbf{x})$ will be full-rank.

- Positive Input Wire: An ADP for the (positive) input wire $f(x_1, \ldots, x_n) = x_j$ consists of $n+1$ scalars (i.e. a width-1 ADP) $(a, b_1, \ldots, b_n)$ set as follows. Sample a uniformly random $r \leftarrow \mathbb{F}_p$ and set $a = -r$ and $b_j = r$. For all $i \neq j$, set $b_i = 0$.

- Negated Input Wire: An ADP for the (negated) input wire $f(x_1, \ldots, x_n) = \neg x_j$ consists of $n+1$ scalars $(a, b_1, \ldots, b_n)$ set as follows. Sample a uniformly random $r \leftarrow \mathbb{F}_p$ and set $b_j = r$. Set $a = 0$ and $b_i = 0$ for all $i \neq j$.

For any input $\mathbf{x} \in \{0,1\}^n$, correctness of the input wire ADP constructions holds with high probability over the randomness of r.

The ADP M that computes the AND of two n-bit input ADPs $\mathsf{M}_1, \mathsf{M}_2$ with widths w_1, w_2 respectively is constructed as follows. Define $k \coloneqq k_1 + k_2$. Sample a uniformly random matrix $\mathbf{R} \leftarrow \mathbb{F}_q^{(k-1)\times k}$ and a uniformly random matrix $\mathbf{S} \leftarrow \mathbb{F}_q^{k\times(k-1)}$. Output

$$\mathsf{M}(\mathbf{x}) = \mathbf{R} \cdot \begin{pmatrix} \mathsf{M}_1(\mathbf{x}) & 0 \\ 0 & \mathsf{M}_2(\mathbf{x}) \end{pmatrix} \cdot \mathbf{S}.$$

We briefly sketch correctness of the AND construction.

- (Both input wires accept) If $\det(\mathsf{M}_1(\mathbf{x})) = 0$ and $\det(\mathsf{M}_2(\mathbf{x})) = 0$, then

$$\begin{pmatrix} \mathsf{M}_1(\mathbf{x}) & 0 \\ 0 & \mathsf{M}_2(\mathbf{x}) \end{pmatrix}$$

is a $k \times k$ matrix which is rank-deficient by 2. Left- and right-multiplying by random matrices $\mathbf{R} \leftarrow \mathbb{F}_q^{(k-1)\times k}, \mathbf{S} \leftarrow \mathbb{F}_q^{k\times(k-1)}$ yields $\mathsf{M}(\mathbf{x})$ which is rank deficient by 1 with overwhelming probability.

- (Some input wire rejects) If either of $\det(\mathsf{M}_1(\mathbf{x})) \neq 0$ or $\det(\mathsf{M}_2(\mathbf{x})) \neq 0$ holds, then

$$\begin{pmatrix} \mathsf{M}_1(\mathbf{x}) & 0 \\ 0 & \mathsf{M}_2(\mathbf{x}) \end{pmatrix}$$

is a $k \times k$ matrix which is rank-deficient by at most 1. Left- and right-multiplying by random matrices $\mathbf{R} \leftarrow \mathbb{F}_q^{(k-1)\times k}, \mathbf{S} \leftarrow \mathbb{F}_q^{k\times(k-1)}$ yields $\mathsf{M}(\mathbf{x})$ which is full-rank with overwhelming probability.

An ADP M which computes the OR of two n-bit input ADPs $\mathsf{M}_1, \mathsf{M}_2$ with widths w_1, w_2 respectively can be constructed as follows. Define $k \coloneqq k_1 + k_2$. Sample a uniformly random matrix $\mathbf{R} \leftarrow \mathbb{F}_q^{k\times k}$ and a uniformly random matrix $\mathbf{S} \leftarrow \mathbb{F}_q^{k\times k}$. Sample a uniformly random affine function $\mathbf{T} : \{0,1\}^n \to \mathbb{F}_q^{k_1 \times k_2}$. Output

$$\mathsf{M}(\mathbf{x}) = \mathbf{R} \cdot \begin{pmatrix} \mathsf{M}_1(\mathbf{x}) & \mathbf{T}(\mathbf{x}) \\ 0 & \mathsf{M}_2(\mathbf{x}) \end{pmatrix} \cdot \mathbf{S}.$$

We briefly sketch correctness of the OR construction.

- (Either input wire accepts) If $\det(\mathsf{M}_1(\mathbf{x})) = 0$ or $\det(\mathsf{M}_2(\mathbf{x})) = 0$, then

$$\begin{pmatrix} \mathsf{M}_1(\mathbf{x}) & \mathbf{T}(\mathbf{x}) \\ 0 & \mathsf{M}_2(\mathbf{x}) \end{pmatrix}$$

is a $k \times k$ matrix which is rank-deficient by 1; even if both $\det(\mathsf{M}_1(x)) = 0$ and $\det(\mathsf{M}_2(x)) = 0$ hold, the random $\mathbf{T}(\mathbf{x})$ will ensure the rank is only deficient by 1 with overwhelming probability. Left- and right-multiplying by random matrices $\mathbf{R} \leftarrow \mathbb{F}_q^{k\times k}, \mathbf{S} \leftarrow \mathbb{F}_q^{k\times k}$ yields $\mathsf{M}(\mathbf{x})$ which is still rank deficient by 1 with overwhelming probability.

- (Both input wires reject) If both $\det(\mathsf{M}_1(\mathbf{x})) \neq 0$ and $\det(\mathsf{M}_2(\mathbf{x})) \neq 0$ hold, then

$$\begin{pmatrix} \mathsf{M}_1(\mathbf{x}) & \mathbf{T}(\mathbf{x}) \\ 0 & \mathsf{M}_2(\mathbf{x}) \end{pmatrix}$$

is a full-rank $k \times k$ matrix. Left- and right-multiplying by random matrices $\mathbf{R} \leftarrow \mathbb{F}_q^{k\times k}, \mathbf{S} \leftarrow \mathbb{F}_q^{k\times k}$ still yields a full-rank $\mathsf{M}(\mathbf{x})$ with overwhelming probability.

4.3 Encoding Branching Programs as ADPs

In this section we describe a way to represent branching programs as ADPs. This representation is implicit in [28] (see also [4]). Different notions of branching programs (e.g., deterministic vs. non-deterministic vs. counting) correspond to different choices of the field and evaluation Eval.

We start by defining the most expressive notion and then specialize it to the weaker notions on which we will rely.

▶ **Definition 10** (Counting Branching Programs [28]). *A $\mathbb{Z}$-arithmetic branching program computing $f : \{0,1\}^n \to \mathbb{Z}$ is specified by a directed acyclic graph $G = (V, E)$ and a labeling $\phi(\cdot,\cdot)$ where each edge (u, v) is labeled with a literal $\phi(u, v) = x_i$ or $\phi(u, v) = \neg x_i$, or the constant 1. Two of the vertices are labeled s, t. Its size is $|V| - 1$. Any input x induces a subgraph G_x limited to edges consistent with x (i.e. edges that evaluate to 1 on x). An accepting path on input x is a directed $s - t$ path in G_x. A $\mathbb{Z}$-arithmetic branching program (ABP) computes the function $f : \{0,1\}^n \to \mathbb{Z}$ such that $f(x)$ is the number of accepting paths. It computes a boolean function $f : \{0,1\}^n \to \{0,1\}$ if the number of accepting paths is equal to $f(x) \in \{0,1\}$.*

▶ **Lemma 11** ([28, 4]). *Suppose there is a $\mathbb{Z}$-arithmetic branching program (ABP) of size ℓ computing a boolean function f. Suppose $L(x)$ satisfies the following*

- *$L(x)$ has -1 along the second diagonal (right below the main diagonal).*
- *$L(x)$ is 0 below the second diagonal.*
- *Each entry of $L(x)$ is a degree (at most) 1 polynomial in a single input variable x_i.*

More precisely, $L(x)$ is defined as follows. Fix a topological ordering of the vertices in V, and label the columns / rows (from left to right / top to bottom) according to this ordering of vertices. In particular we want s labeled 1 and t labeled ℓ. We first define a matrix $A(x)$ of dimension $\ell \times \ell$. For entry (i, j) of $A(x)$, write $\phi(i, j)$ if (i, j) is an edge in G and 0 otherwise. Note that $A(x)$ will be 0 on and below the main diagonal. Now consider $A(x) - I$, and delete its first column and last row to obtain the $(\ell - 1) \times (\ell - 1)$ dimensional matrix $L(x)$.

Then for all $x \in \{0,1\}^n$, we have $\det(L(x)) = f(x)$. (If f is boolean, we say $f(x)$ accepts if $\det(L(x))$ is non-zero.)

This immediately gives us an ADP for $\mathbb{Z}$-arithmetic branching programs. Namely, $\mathsf{M} = L$ and Eval is simply the identity function (if f is boolean). This ADP can also be implemented over a finite field $\mathbb{F}_p$ if the number of accepting paths is at most $p - 1$. Simply compute $f(x) = \det(L(x)) \mod p$.

Example

Suppose we have vertices in the order s, v_1, v_2, t. We can write a point function that accepts on $x = 010$ with an edge from s to v_1 labeled with $1 - x_1$, an edge from v_1 to v_2 labeled with x_2, and an edge from v_2 to t labeled with $1 - x_3$. So the resulting $A(x) - I$ and $L(x)$ matrices are

$$A(x) = \begin{pmatrix} -1 & 1 - x_1 & 0 & 0 \\ 0 & -1 & x_2 & 0 \\ 0 & 0 & -1 & 1 - x_3 \\ 0 & 0 & 0 & -1 \end{pmatrix},$$

$$L(x) = \begin{pmatrix} 1-x_1 & 0 & 0 \\ -1 & x_2 & 0 \\ 0 & -1 & 1-x_3 \end{pmatrix}.$$

Thus the resulting ADP is of the following form:

$$\mathbf{A} = \begin{pmatrix} 1 & 0 & 0 \\ -1 & 0 & 0 \\ 0 & -1 & 1 \end{pmatrix}, \mathbf{B}_1 = \begin{pmatrix} -1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix},$$

$$\mathbf{B}_2 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \mathbf{B}_3 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -1 \end{pmatrix}.$$

Then, as done in [4] and described above, the resulting program can be "re-randomized" by left- and right-multiplying by uniformly random matrices $\mathbf{R}, \mathbf{S}$ such that $\det(\mathbf{R}) = \det(\mathbf{S}) = 1$.

5 Witness Encryption

5.1 Definitions

In this section we will use ADPs to construct *witness encryption* [18]. Since our focus will not be on encoding *functions* as ADPs, we will ignore the Eval part of our ADP formalism. For the remainder of this section, a width-k ADP M over $\mathbb{F}_p$ will refer to a tuple of $n+1$ matrices $(\mathbf{A}, \mathbf{B}_1, \ldots, \mathbf{B}_n)$, where each matrix is in $\mathbb{F}_p^{k \times k}$. Since we will not have a dedicated Eval function, the evaluation of $\mathsf{M}(\cdot)$ on an input $\mathbf{x} \in \{0,1\}^n$ will refer to the matrix

$$\mathsf{M}(\mathbf{x}) \coloneqq \mathbf{A} + \sum_{i \in [n]} x_i \mathbf{B}_i.$$

Consider the NP-complete subset sum problem (or more generally, *vector* subset sum).

▶ **Definition 12.** *The* VECTOR-SUBSET-SUM *language consists of instances* $(\mathbf{H}, \boldsymbol{\ell})$ *where* $\mathbf{H} \in \mathbb{Z}^{d \times n}, \boldsymbol{\ell} \in \mathbb{Z}^d$*, such that there exists* $\mathbf{w} \in \{0,1\}^n$ *satisfying* $\mathbf{H} \cdot \mathbf{w} = \boldsymbol{\ell}$ *(over the integers).*

We refer to d as the *dimension* of the instance. When $d = 1$, we simply refer to this as SUBSET-SUM.

Vector Subset Sum Encoding

A dimension $d = 1$ instance $(\mathbf{h}, \ell)$ can be represented as an ADP of arbitrary width k as follows. Sample a uniformly random matrix $\mathbf{R} \leftarrow \mathbb{F}_p^{k \times k}$ and set $\mathbf{A} = -\ell\mathbf{R}$ and $\mathbf{B}_i = h_i\mathbf{R}$ for all $i \in [n]$. This ADP accepts on input $\mathbf{w} \in \{0,1\}^n$ if and only if

$$\det(\mathbf{A} + \sum_{i \in [n]} w_i \mathbf{B}_i) = \det((-\ell + \sum_{i \in [n]} w_i h_i)\mathbf{R}) = 0.$$

We can generalize this approach to any instance $(\mathbf{H}, \boldsymbol{\ell})$ of dimension $d \geq 1$ as follows. We define the procedure VSS.Encode, which takes an instance $(\mathbf{H}, \boldsymbol{\ell})$ of VECTOR-SUBSET-SUM instance and outputs a width-k ADP over $\mathbb{F}_p$.

- VSS.Encode$((\mathbf{H} \in \mathbb{Z}^{d \times n}, \boldsymbol{\ell} \in \mathbb{Z}^d), p, k)$: Interpret each entry of $\mathbf{H}$ and $\boldsymbol{\ell}$ as the corresponding element over $\mathbb{F}_p$; for correctness we will require $p > \max_{\mathbf{x} \in \{0,1\}^n} ||\mathbf{H} \cdot \mathbf{x}||_\infty$. For each $j \in [d]$, draw uniformly random $\mathbf{R}_j \leftarrow \mathbb{F}_p^{k \times k}$. Set $\mathbf{A} \coloneqq -\sum_{j \in [d]} \ell_j \mathbf{R}_j$ and $\mathbf{B}_i \coloneqq \sum_{j \in [d]} H_{j,i} \mathbf{R}_j$ for each $i \in [n]$. Output

  $$\mathsf{M}_{\mathbf{H},\boldsymbol{\ell}} = (\mathbf{A}, \mathbf{B}_1, \ldots, \mathbf{B}_n).$$

An Insecure Witness Encryption

Consider the following *insecure* construction of witness encryption for VECTOR-SUBSET-SUM. To "encrypt" a bit b with respect to an instance $(\mathbf{H}, \boldsymbol{\ell})$:

1. Fix a width k and a prime $p > \max_{\mathbf{x}\in\{0,1\}^n} ||\mathbf{H}\cdot\mathbf{x}||_\infty$ such that p is at least super-polynomial in the security parameter.
2. Sample
$$\mathsf{M}_{\mathbf{H},\boldsymbol{\ell}} \leftarrow \mathsf{VSS.Encode}((\mathbf{H},\boldsymbol{\ell}), p, k).$$
3. Sample uniformly random $\mathbf{S} \leftarrow \mathbb{F}_p^{k\times k}$ and output the ciphertext
$$\mathsf{M}_{\mathbf{H},\boldsymbol{\ell},b} \coloneqq \mathsf{M}_{\mathbf{H},\boldsymbol{\ell}} + (b\mathbf{S}, \mathbf{0}, \ldots, \mathbf{0}).$$

To decrypt a ciphertext of the form $(\mathbf{A}, \mathbf{B}_1, \ldots, \mathbf{B}_n)$ using a witness $\mathbf{w} \in \{0,1\}^n$, we compute

$$\det(\mathbf{A} + \sum_{i\in[n]} w_i\mathbf{B}_i).$$

If this determinant is a non-zero element in $\mathbb{F}_p$, then $b = 1$ and if the result is zero then $b = 0$.

Correctness holds since if $\mathbf{w} \in \{0,1\}^n$ is a witness for $(\mathbf{H}, \boldsymbol{\ell}) \in$ VECTOR-SUBSET-SUM, then $\mathbf{H}\cdot\mathbf{w} = \boldsymbol{\ell}$, so

$$\mathbf{A} + \sum_{i\in[n]} w_i\mathbf{B}_i = b\mathbf{S}.$$

Since $\mathbf{S}$ is a random matrix in $\mathbb{F}_p$, where p is super-polynomial in the security parameter, its determinant is nonzero with overwhelming probability.

However, this construction is insecure; given an instance $(\mathbf{H}, \boldsymbol{\ell}) \notin$ VECTOR-SUBSET-SUM, an adversary can potentially find $\mathbf{x} \in \mathbb{F}_p^n$ (where $\mathbf{x} \notin \{0,1\}^n$) such that $\mathbf{H}\cdot\mathbf{x} = \boldsymbol{\ell}$. The same correctness argument used for *valid* witnesses $\mathbf{w} \in \{0,1\}^n$ will also imply that for such an $\mathbf{x} \in \mathbb{F}_p^n$, the "encrypted" bit b can be recovered as

$$b = \mathsf{Eval}_{\neq 0}(\det(\mathsf{M}_{\mathbf{H},\boldsymbol{\ell},b}(\mathbf{x}))).$$

Preventing Invalid Evaluations

For this approach to have any hope for security, we will need to modify the ciphertext $\mathsf{M}_{\mathbf{H},\boldsymbol{\ell},b}$ so that $\det(\mathsf{M}_{\mathbf{H},\boldsymbol{\ell},b}(\mathbf{x}))$ does not leak the value of b on inputs $\mathbf{x} \notin \{0,1\}^n$.

Our approach is to add "noise" to $\mathsf{M}_{\mathbf{H},\boldsymbol{\ell},b}$ which will prevent the adversary from gaining information on non-binary inputs. This noise will take the form of another ADP $\mathsf{M}_{\mathsf{noise}}$ which will satisfy the following properties:

- (Zero on binary inputs) For all $\mathbf{x} \in \{0,1\}^n$, $\det(\mathsf{M}_{\mathsf{noise}}(\mathbf{x})) = 0$,
- (Non-zero on non-binary inputs): For all $\mathbf{x} \in \mathbb{F}_p^n \setminus \{0,1\}^n$, $\Pr[\det(\mathsf{M}_{\mathsf{noise}}(\mathbf{x})) = 0] = \mathsf{negl}(n)$,

where the probability is taken over the randomness used to generate $\mathsf{M}_{\mathsf{noise}}$. In the construction, we will simply add $\mathsf{M}_{\mathsf{noise}}$ to $\mathsf{M}_{\mathbf{H},\boldsymbol{\ell},b}$, so the "non-zero on non-binary inputs" property will intuitively block the attack described above. We set $\mathsf{M}_{\mathsf{noise}}$ to be the result of sampling from an "All-Accept" encoding scheme, defined as follows.

An All-Accept ADP encoding scheme AllAcc.Gen is a randomized procedure that takes an input length n and field $\mathbb{F}_p$, and outputs M of width k with the following properties:

- (Correctness) For all $\mathbf{x} \in \{0,1\}^n$,
$$\Pr[\det(\mathsf{M}(\mathbf{x})) = 0 : \mathsf{AllAcc.Gen}(n, \mathbb{F}_p) \to (\mathsf{M}, k)] = 1.$$
- (Rejection of Invalid Inputs) For all non-binary $\mathbf{x}$, i.e. $\mathbf{x} \in \mathbb{F}_p^n \setminus \{0,1\}^n$,
$$\Pr[\det(\mathsf{M}(\mathbf{x})) = 0 : \mathsf{AllAcc.Gen}(n, \mathbb{F}_p) \to (\mathsf{M}, k)] = \mathsf{negl}(n).$$

An All-Accept ADP from a Boolean Formula Encoding

A natural attempt to construct such an encoding scheme would be to use the formula encoding scheme described in Section 4.2 to encode any formula that accepts all boolean inputs $\mathbf{x} \in \{0,1\}^n$. Unfortunately, this does not in general guarantee rejection of invalid inputs. However, it turns out that encoding the simple boolean formula

$$f(x_1, \ldots, x_n) = (x_1 \vee \neg x_1) \wedge \cdots \wedge (x_n \vee \neg x_n)$$

via the procedure described above does guarantee rejection of invalid inputs. This sampling procedure is equivalent to the concrete procedure given here.

$\mathsf{AllAcc.Gen}^{\mathsf{FORM}}(n, \mathbb{F}_p)$:

- Draw uniformly at random 4 sets of n vectors of dimension $n+1$:

  $$\{\mathbf{u}_i\}_{i\in[n]}, \{\mathbf{v}_i\}_{i\in[n]}, \{\mathbf{s}_i\}_{i\in[n]}, \{\mathbf{t}_i\}_{i\in[n]},$$

  along with n^2 scalars $\{c_j^{(i)}\}_{i,j\in[n]}$ over $\mathbb{F}_p$.
- Let

  $$\mathbf{A} = \sum_{i\in[n]} \mathbf{u}_i \mathbf{v}_i^\top,$$
  $$\mathbf{B}_i = -\mathbf{u}_i \mathbf{v}_i^\top + \mathbf{s}_i \mathbf{t}_i^\top + \sum_{j\in[n]} c_i^{(j)} \mathbf{u}_j \mathbf{t}_j^\top,$$

  and output $\mathsf{M} = (\mathbf{A}, \mathbf{B}_1, \ldots, \mathbf{B}_n)$.

Correctness and security follow by fixing the field size $p = \omega(\mathsf{poly}(n))$ and rewriting an evaluation on input $\mathbf{x} \in \mathbb{F}_p^n$ as follows, setting $k_i \coloneqq \sum_{j\in[n]} x_j c_j^{(i)}$:

$$\begin{aligned}
\mathbf{A} + \sum_{i\in[n]} x_i \mathbf{B}_i &= \sum_{i\in[n]} \Big((1-x_i)\mathbf{u}_i \mathbf{v}_i^\top + x_i \mathbf{s}_i \mathbf{t}_i^\top + k_i \mathbf{u}_i \mathbf{t}_i^\top \Big) \\
&= \sum_{i\in[n]:x_i=0} \mathbf{u}_i \Big(\mathbf{v}_i^\top + k_i \mathbf{t}_i^\top \Big) + \sum_{i\in[n]:x_i=1} \Big(\mathbf{s}_i + k_i \mathbf{u}_i \Big) \mathbf{t}_i^\top \\
&\quad + \sum_{i\in[n]:x_i\notin\{0,1\}} \mathbf{u}_i \Big((1-x_i)\mathbf{v}_i^\top + k_i \mathbf{t}_i^\top \Big) + x_i \mathbf{s}_i \mathbf{t}_i^\top .
\end{aligned}$$

Thus when $\mathbf{x}$ is binary, $\mathbf{A} + \sum_{i\in[n]} x_i \mathbf{B}_i$ can be written as a sum of n rank one matrices (recall the dimension is $n+1$). Otherwise, it can be written as a sum of at least $n+1$ rank one matrices which do not share column or row spans (with overwhelming probability over the sampling randomness). Since the field size is $p = \omega(\mathsf{poly}(n))$, the resulting matrix will be full rank except with probability negligible in the input length n.

We now describe a slight generalization of the above sampling procedure, which will help with security of the eventual witness encryption candidate. We replace the vectors $\mathbf{u}_i, \mathbf{v}_i, \mathbf{s}_i, \mathbf{t}_i$ with matrices $\mathbf{U}_i, \mathbf{V}_i, \mathbf{S}_i, \mathbf{T}_i$ of width $q(n)$ (where $q(\cdot)$ is now a parameter for the encoding procedure). In the sampling procedure given above, $q(n)$ is implicitly set to always output 1. In general, the resulting dimension of the output program will be set to $k = nq(n) + 1$. This more general sampling procedure is used in the concrete candidate we give in Section 5.2.

An All-Accept ADP from "Nearly Skew Symmetric" Matrices

To state our next construction of an all-accept ADP, it will be helpful to define a "nearly-skew-symmetric" matrix.

▶ **Definition 13** (Nearly Skew Symmetric Matrices). *A matrix* $\mathbf{N} \in \mathbb{F}_p^{k \times k}$ *is nearly-skew-symmetric (*NSS*) if* $\mathbf{N}_{i,j} = -\mathbf{N}_{i,j}$ *for all* $i, j > 1$ *such that* $i \neq j$*, and* $\mathbf{N}_{i,1} + \mathbf{N}_{1,i} + \mathbf{N}_{i,i} = 0$ *for all* i*.*

Observe that any nearly-skew-symmetric matrix will have $\mathbf{N}_{1,1} = 0$. Furthermore, the bottom-right $(k-1) \times (k-1)$ supmatrix of any NSS matrix is skew-symmetric. Setting $n = k - 1$, note that it is easy to sample a random NSS by choosing uniformly random field elements $a_1, \ldots, a_n, d_1, \ldots, d_n$, as well as a uniformly random width-n skew-symmetric matrix B, and outputting

$$\begin{pmatrix} 0 & 0 & \cdots & 0 \\ 0 & & & \\ \vdots & & B & \\ 0 & & & \end{pmatrix} + \begin{pmatrix} 0 & a_1 & \cdots & a_n \\ -a_1 - d_1 & d_1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ -a_n - d_n & 0 & \cdots & d_n \end{pmatrix}.$$

The above formulation makes it clear that for any finite field $\mathbb{F}_p$ and matrix width k, the set of all NSS matrices lies in a known linear subspace. Let $\mathsf{NSS.Sample}(p, k)$ be the procedure that produces a uniformly random sample from this subspace, i.e. outputs a random nearly-skew-symmetric matrix $\mathbf{N}$ of width k over $\mathbb{F}_p$.

We define the following all-accept encoding procedure.

$\mathsf{AllAcc.Gen}^{\mathsf{NSS}}(n, \mathbb{F}_p)$:

- Let $k = n + 1$, and draw a uniformly random full rank matrix $\mathbf{T} \leftarrow \mathbb{F}_p^{k \times k}$.
- For each $i \in [k]$, draw $\mathbf{N}_i \leftarrow \mathsf{NSS.Sample}(k)$ and set $\mathbf{C}_i = \mathbf{T}^{-1} \cdot \mathbf{N}_i$.
- Let $\mathbf{A}$ be the $k \times k$ matrix obtained by concatenating the first column of each of the $\mathbf{C}_i$ matrices, and let $\mathbf{B}_i$ be the $k \times k$ matrix obtained by concatenating the $i+1$st columns of each of the $\mathbf{C}_i$ matrices. Output $\mathsf{M} = (\mathbf{A}, \mathbf{B}_1, \ldots, \mathbf{B}_n)$.

To show correctness, fix any $\mathbf{x} \in \{0,1\}^n$, and let $\mathbf{x}' = [1 \mid \mathbf{x}^\top]^\top$. Denote the jth column of a matrix $\mathbf{M}$ by $(\mathbf{M})_j$. Then for all $j \in [n+1]$,

$$\mathbf{T} \cdot (\mathbf{A} + \sum_{i \in [n]} x_i \mathbf{B}_i)_j = \mathbf{T} \cdot \mathbf{C}_j \cdot \mathbf{x}' = \mathbf{N}_j \cdot \mathbf{x}'.$$

Now we claim that for all $j, \mathbf{x}'^\top \cdot \mathbf{N}_j \cdot \mathbf{x}' = 0$, which implies that $\mathbf{x}'^\top \cdot \mathbf{T}$ is in the kernel of $\mathbf{A} + \sum_{i \in [n]} x_i \mathbf{B}_i$, and thus that $\det(\mathsf{M}(\mathbf{x})) = 0$. In fact, for any NSS matrix $\mathbf{N}$, and any vector $\mathbf{x}' = [1 \mid \mathbf{x}^\top]^\top$, where $\mathbf{x} \in \{0,1\}^n$, it holds that $\mathbf{x}'^\top \cdot \mathbf{N} \cdot \mathbf{x}' = 0$. To see this, treat each entry of $\mathbf{x}$ as a distinct formal variable, and expand out the expression as a quadratic polynomial over $x_1, \ldots, x_n$. Since $\mathbf{x}$ is binary, we know that $x_i^2 = x_i$. Thus, the coefficient on x_i is equal to $\mathbf{N}_{1,i} + \mathbf{N}_{i,1} + \mathbf{N}_{i,i} = 0$, and the coefficient on $x_i x_j$ for $i \neq j$ is equal to $\mathbf{N}_{i,j} + \mathbf{N}_{j,i} = 0$, which establishes the claim.

To argue security, we take n to be the security parameter, and fix the field size to be $p = \omega(\mathsf{poly}(n))$. Fix any $\mathbf{x} \notin \{0,1\}^n$ and again let $\mathbf{x}' = [1 \mid \mathbf{x}^\top]^\top$. We want to bound the probability that there exists some vector $\mathbf{v} \neq \mathbf{0}$ such that $\mathbf{v}^\top \cdot (\mathbf{A} + \sum_i x_i \mathbf{B}_i) = 0$. Setting $\mathbf{w}'^\top = \mathbf{v}^\top \cdot \mathbf{T}^{-1}$, we have that

$$\mathbf{v}^\top \cdot (\mathbf{A} + \sum_i x_i \mathbf{B}_i) = 0 \Leftrightarrow \mathbf{w}'^\top \cdot \mathbf{N}_i \cdot \mathbf{x}' = 0 \ \forall i \in [n+1].$$

We will union bound over all vectors $\mathbf{w'}^\top$ to show that there does not exist such a vector with high probability. Note that it suffices to union bound over all $\mathbf{w'}$ with first entry equal to 0 or 1, for a total of $2p^n$ vectors.

First consider the case where $\mathbf{w'}^\top = [0 \mid \mathbf{w}^\top]$ for some $\mathbf{w} \in \mathbb{F}_p^n$. Recall that each $\mathbf{N}_i$ is generated by choosing random values $a_1, \ldots, a_n, d_1, \ldots, d_n$ and a skew-symmetric $\mathbf{B}$ and arranging in a matrix as described above. Now treat these values as formal variables, and consider the linear polynomial over these variables induced by the expression $[0 \mid \mathbf{w}^\top] \cdot \mathbf{N}_i \cdot [1 \mid \mathbf{x}^\top]^\top$. We argue that this polynomial must not be identically zero. The coefficient on the variable a_i is exactly $-w_i$. Since $\mathbf{w'} \neq \mathbf{0}$, it must be the case that $\mathbf{w} \neq \mathbf{0}$, so there must be some i such that $w_i \neq 0$.

Now consider the case where $\mathbf{w'}^\top = [1 \mid \mathbf{w}^\top]$ for some $\mathbf{w} \in \mathbb{F}_p^n$. The coefficient on a_i is exactly $w_i(x_i - 1)$ and the coefficient on d_i is exactly $x_i(w_i - 1)$. But by assumption, there must be some i such that $x_i \notin \{0, 1\}$, implying that the coefficient on either a_i or d_i is non-zero. Thus, for each of the $2p^n$ vectors $\mathbf{w'}$ that we consider, the probability that $\mathbf{w'}^\top \cdot \mathbf{N}_i \cdot \mathbf{x'} = 0$ is at most $1/p$ over the randomness of sampling $\mathbf{N}_i$. This holds simultaneously for each $i \in [n+1]$ with probability at most $1/p^{n+1}$. So by a union bound, there exists a $\mathbf{w'}$ orthogonal to each $\mathbf{N}_i \cdot \mathbf{x'}$ with probability at most $2/p = \mathsf{negl}(n)$.

5.2 Candidate Witness Encryption Constructions

Generic Candidate

Given the above definitions, we can define a general paradigm for constructing witness encryption (encrypting a single bit b) for VECTOR-SUBSET-SUM. Instantiating the framework with any AllAcc.Gen and VSS.Encode procedures will result in a correct witness encryption.

- $\mathsf{WE.Enc}(b, (\mathbf{H}, \boldsymbol{\ell}))$: Let $d \times n$ be the dimension of $\mathbf{H}$.
  1. Choose a prime $p > \max_{\mathbf{x} \in \{0,1\}^n} ||\mathbf{H} \cdot \mathbf{x}||_\infty$. Additionally we require $p = \omega(\mathsf{poly}(n))$.
  2. Sample $(\mathsf{M_{AA}}, k) \leftarrow \mathsf{AllAcc.Gen}(n, \mathbb{F}_p)$.
  3. Sample $\mathsf{M}_{\mathbf{H},\boldsymbol{\ell}} \leftarrow \mathsf{VSS.Encode}((\mathbf{H}, \boldsymbol{\ell}), \mathbb{F}_p, k)$.
  4. Sample $\mathbf{S} \leftarrow \mathbb{F}_p^{k \times k}$.
  5. Output $\mathsf{M_{AA}} + \mathsf{M}_{\mathbf{H},\boldsymbol{\ell}} + (b\mathbf{S}, \mathbf{0}, \ldots, \mathbf{0})$.
- $\mathsf{WE.Dec}(\mathsf{M}, w)$: output $\mathsf{Eval}_{\neq 0}(\det(\mathsf{M}(w)))$.

Concrete Candidate

We now give a simple, self-contained instantiation of the general witness encryption framework described above.

This candidate will be a witness encryption for SUBSET-SUM, with instances $(\mathbf{h} \in \mathbb{Z}^n, \ell \in \mathbb{Z})$. Given an instance $(\mathbf{h} \in \mathbb{Z}^n, \ell \in \mathbb{Z})$, define the instance $(\mathbf{H} \in \mathbb{Z}^{(n+1) \times 2n}, \boldsymbol{\ell} \in \mathbb{Z}^{n+1})$ of VECTOR-SUBSET-SUM where

$$\mathbf{H} \coloneqq \begin{pmatrix} h_1 & h_2 & \cdots & h_n & 0 & 0 & \cdots & 0 \\ 1 & 0 & \cdots & 0 & 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & 0 & 0 & \cdots & 1 \end{pmatrix}, \boldsymbol{\ell} \coloneqq \begin{pmatrix} \ell \\ 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix}.$$

As discussed in Section 7, this transformation will help with security. In particular, an instance $((h_1, \ldots, h_n), \ell) \notin \mathsf{SUBSET\text{-}SUM}$ could potentially have many $h_i = 0$, so encoding without the above transformation will leak many $\mathbf{B}_i$ of the all-accept encoding in the clear.

The following instantiation combines the $\mathsf{VSS.Encode}$ scheme described above with the all-accept encoding $\mathsf{AllAcc.Gen}^{\mathsf{FORM}}$, with parameter $q(n) = n^\epsilon$ for some constant $\epsilon > 0$. As discussed in Section 7.1, setting $q(\cdot)$ to be a super-constant function of n will help defend against MQ-style attack strategies. Note that the encryption scheme below takes as input a one-dimensional $\mathsf{SUBSET\text{-}SUM}$ instance and first follows the above conversion to a $\mathsf{VECTOR\text{-}SUBSET\text{-}SUM}$ instance. Then, it encrypts under this $\mathsf{VECTOR\text{-}SUBSET\text{-}SUM}$ instance following the generic paradigm above.

- $\mathsf{WE.Enc}(b, ((h_1, \ldots, h_n), \ell))$. Choose $p > \max_{\mathbf{x}\in\{0,1\}^n} |\sum_i x_i h_i|$ such that $p = \omega(\mathsf{poly}(n))$, and fix the field $\mathbb{F}_p$.
  1. For each $i \in [2n]$, draw 4 uniformly random matrices
  $$\mathbf{U}_i, \mathbf{V}_i, \mathbf{S}_i, \mathbf{T}_i \leftarrow \mathbb{F}_p^{(2n^{1+\epsilon}+1)\times n^\epsilon}.$$
  2. For each $i, j \in [2n]$, draw a uniformly random scalar
  $$c_j^{(i)} \leftarrow \mathbb{F}_p.$$
  3. Draw $2n$ uniformly random matrices
  $$\mathbf{R}_0, \mathbf{R}_1, \ldots, \mathbf{R}_n, \mathbf{S} \leftarrow \mathbb{F}_p^{(2n^{1+\epsilon}+1)\times(2n^{1+\epsilon}+1)}.$$
  4. Define
  $$\mathbf{A} := \sum_{i\in[2n]} \mathbf{U}_i\mathbf{V}_i^\top - \ell\mathbf{R}_0 - \sum_{i\in[n]} \mathbf{R}_i + b\mathbf{S}.$$
  For $\ell \in [n]$, define
  $$\mathbf{B}_\ell := -\mathbf{U}_\ell\mathbf{V}_\ell^\top + \mathbf{V}_\ell\mathbf{T}_\ell^\top + \sum_{j\in[2n]} c_\ell^{(j)}\mathbf{U}_j\mathbf{T}_j^\top + h_\ell\mathbf{R}_0 + \mathbf{R}_\ell$$
  For $\ell \in \{n+1, \ldots, 2n\}$, define
  $$\mathbf{B}_\ell := -\mathbf{U}_\ell\mathbf{V}_\ell^\top + \mathbf{S}_\ell\mathbf{T}_\ell^\top + \sum_{j\in[2n]} c_\ell^{(j)}\mathbf{U}_j\mathbf{T}_j^\top + \mathbf{R}_\ell.$$
  5. Finally, output
  $$\mathsf{M} = (\mathbf{A}, \mathbf{B}_1, \ldots, \mathbf{B}_{2n}).$$
- $\mathsf{WE.Dec}(\mathsf{M}, w)$. Let $\widehat{w}$ be such that each $\widehat{w}_i = 1 - w_i$. Output $\mathsf{Eval}_{\neq 0}(\det(\mathsf{M}([w \mid \widehat{w}])))$.

6 Applications of ADP-Based Witness Encryption

6.1 Public-Key Encryption with Short Public Keys

In this section, we show how to use our witness encryption candidate to construct a public key encryption scheme with extremely short public keys and with fast key generation.

6.1.1 A Generic Framework

We recall from [19] the generic construction of public-key encryption from a witness encryption scheme WE and any pseudo-random generator $g : \{0,1\}^\lambda \to \{0,1\}^{n(\lambda)}$.

- $\mathsf{PKE.Gen}(1^\lambda)$. Sample a uniformly random $\mathsf{sk} \leftarrow \{0,1\}^\lambda$, and set $\mathsf{pk} = g(\mathsf{sk})$. Output $(\mathsf{pk}, \mathsf{sk})$.
- $\mathsf{PKE.Encrypt}(\mathsf{pk}, m \in \{0,1\})$. Define the boolean circuit $C[g, \mathsf{pk}](\cdot)$, which on input $\mathbf{x} \in \{0,1\}^\lambda$, outputs 1 if and only if $g(\mathbf{x}) = \mathsf{pk}$. Interpreting $C[g, \mathsf{pk}](\cdot)$ as an instance of CIRCUIT-SAT, apply a deterministic reduction from CIRCUIT-SAT to SUBSET-SUM and obtain an instance $(\mathbf{h}, \ell)$. Output $c \leftarrow \mathsf{WE.Enc}(1^\lambda, (\mathbf{h}, \ell), m)$.
- $\mathsf{PKE.Decrypt}(\mathsf{sk}, (\mathbf{h}, \ell), c)$. Run the reduction from CIRCUIT-SAT to SUBSET-SUM to transform the witness sk for $C[g, \mathsf{pk}](\cdot)$ to a subset-sum witness $\mathbf{w}$ for $(\mathbf{h}, \ell)$. Return $m \leftarrow \mathsf{WE.Dec}(c, \mathbf{w})$.

▶ Remark 14. The choice of g is fixed in the specification of the public-key encryption scheme and is not included in pk. Notice that 1^λ is an input to WE.Enc since there is no set-up/gen algorithm in witness encryption, but it is not an explicit input to PKE.Encrypt; however, it can be easily obtained by inspecting the length of pk.

The following claim captures the fact that security of this construction can be proved assuming either the standard security notion for witness encryption [19], or the stronger extractable security notion given by [23]; in the latter case the requirement on g can be weakened to one-wayness.

▷ Claim 15 (Security of Witness-Encryption-based PKE). The above public-key encryption scheme satisfies CPA-security if

- WE is secure under the standard [19] security notion for witness encryption, and $g : \{0,1\}^\lambda \to \{0,1\}^{n(\lambda)}$ is a pseudo-random generator with $n(\lambda) \geq 2\lambda$, OR
- WE is secure under the stronger [23] security notion for *extractable* witness encryption, and $g : \{0,1\}^\lambda \to \{0,1\}^{n(\lambda)}$ is a one-way function.

Proof. In the case where we use the standard [19] security notion for witness encryption, an encryption with respect to pk which is not generated as the output of g computationally hides the encrypted message. Therefore, an adversary that can distinguish encryptions of 0 and 1 also distinguishes between outputs of g and uniformly random strings, breaking the pseudorandomness of g.

If we rely on the [23] extractable security notion, then an adversary which can distinguish encryptions of 0 and 1 implies the existence of an extractor which outputs a witness that pk is in the image of g; this breaks the one-wayness of g. ◀

6.1.2 A Concrete Instantiation from Goldreich's PRG

For an arbitrary one-way function, the CIRCUIT-SAT to SUBSET-SUM reduction may be costly. Thus, we give a concrete instantiation of the above approach using Goldreich's PRG [22, 3] in place of g, along with a custom reduction to VECTOR-SUBSET-SUM. We first recall the construction of Goldreich's PRG.

▶ **Definition 16** (Goldreich's PRG with locality k). *Fix a boolean predicate $P : \{0,1\}^k \to \{0,1\}$, and a list $S_1, \ldots, S_m \in [n]^k$ of k-tuples of indices. Write S_i as $(i_1, \ldots, i_k)$. On input $\mathbf{x} \in \{0,1\}^n$, the jth bit of the output is set to $P(x_{i_1}, \ldots, x_{i_k})$.*

For this work, we will exclusively focus on the locality $k = 5$ setting with the TSA ("Tri-Sum-And") predicate, defined as

$$\mathrm{TSA}(x_1, x_2, x_3, x_4, x_5) = x_1 + x_2 + x_3 + x_4 x_5 \mod 2.$$

Hereinafter, "Goldreich's PRG" will refer to the instantiation with the TSA predicate. For a list $S = (S_1, \ldots, S_m)$ where each $S_i \in [n]^5$, let $g_S : \{0,1\}^n \to \{0,1\}^m$ be Goldreich's PRG instantiated with the TSA predicate, parameterized by index sets $S_1, \ldots, S_m$.

▶ **Definition 17.** *The* GOLDREICH *language consists of instances* $(n, \mathbf{y}, (S_1, \ldots, S_m))$ *such that each* $S_i \in [n]^5$*, and* $\mathbf{y}$ *is in the image of* g_S *where* $S = (S_1, \ldots, S_m)$*.*

▶ **Lemma 18** (GOLDREICH to VECTOR-SUBSET-SUM)**.** *There is a Karp reduction which takes* $(n, \mathbf{y}, (S_1, \ldots, S_m))$ *in* GOLDREICH *and outputs* $(\mathbf{H}, \boldsymbol{\ell})$ *in* VECTOR-SUBSET-SUM*, where* $\mathbf{H} \in \mathbb{Z}^{m \times (n+2m)}$ *and* $\boldsymbol{\ell} \in \mathbb{Z}^m$*.*

Proof. Given a list $S = (S_1, \ldots, S_m)$ with corresponding Goldreich PRG $g_S : \{0,1\}^n \to \{0,1\}^m$, along with a string $\mathbf{y} \in \{0,1\}^m$, we construct an instance $(\mathbf{H}, \boldsymbol{\ell})$ where $\mathbf{H} \in \mathbb{Z}^{m \times (n+2m)}$ and $\boldsymbol{\ell} \in \mathbb{Z}^m$ such that there exists $\mathbf{w} \in \{0,1\}^{n+2m}$ satisfying $\mathbf{H} \cdot \mathbf{w} = \boldsymbol{\ell}$ if and only there exists $\mathbf{x} \in \{0,1\}^n$ satisfying $G_S(\mathbf{x}) = \mathbf{y}$.

We define $n + 2m$ variables by allocating 1 variable for each input position $j \in [n]$, and 2 variables a_i, b_i for each output position $i \in [m]$. We generate m equations as follows. For each $i \in [m]$, write $S_i = (i_1, i_2, i_3, i_4, i_5)$ and consider the pair (S_i, y_i).

- If $y_i = 0$, the associated equation is

$$2x_{i_1} + 2x_{i_2} + 2x_{i_3} - x_{i_4} - x_{i_5} - 4a_i + b_i = 0.$$

- If $y_i = 1$, the associated equation is

$$2x_{i_1} + 2x_{i_2} + 2x_{i_3} + x_{i_4} + x_{i_5} - 4a_i - b_i = 2.$$

Since each equation is a linear equation over $n + 2m$ variables, writing the coefficients of each of the m equations as a matrix yields the matrix $\mathbf{H} \in \mathbb{Z}^{m \times (n+2m)}$. Correspondingly, $\boldsymbol{\ell} \in \mathbb{Z}^m$ is specified by the right-hand-side values of the m equations.

We can extend a pre-image $\mathbf{x} \in \{0,1\}^n$ into a vector $\mathbf{w} \in \{0,1\}^{n+2m}$ satisfying $\mathbf{H} \cdot \mathbf{w} = \boldsymbol{\ell}$ as follows. For each $i \in [m]$:

- If $y_i = 0$, set $a_i = 1$ if and only if $x_{i_1} + x_{i_2} + x_{i_3} \geq 2$.
- If $y_i = 1$, set $a_i = 1$ if and only if $x_{i_1} + x_{i_2} + x_{i_3} + x_{i_4} x_{i_5} \geq 3$.
- Regardless of y_i, set $b_i = 1$ if and only if $x_{i_4} + x_{i_5} = 1$.

It can be verified that if any constraint i is not satisfied, then there is no $\{0,1\}$ setting of the a_i, b_i variables that can result in a satisfying VECTOR-SUBSET-SUM witness. ◀

6.1.3 Optimizations and Parameter Size Estimates

In this section, we describe an extension of the above witness encryption candidate for one-bit messages that allows for encoding an h-bit message in a single ADP. We will make use of the NSS based all-accept, and will first generalize the $\mathsf{AllAcc.Gen}^{\mathsf{NSS}}$ algorithm to output wide rectangular matrices of dimension $k \times (k + r)$. We include the parameter r in the input to the sampling procedure, noting that the procedure given before implicitly set $r = 0$.

$\mathsf{AllAcc.Gen}^{\mathsf{NSS}}(n, r, \mathbb{F}_p)$:

- Let $k = n + 1$, and draw a uniformly random full rank matrix $\mathbf{T} \leftarrow \mathbb{F}_p^{k \times k}$.
- For each $i \in [k + r]$, draw $\mathbf{N}_i \leftarrow \mathsf{NSS.Sample}(k)$ and set $\mathbf{C}_i = \mathbf{T}^{-1} \cdot \mathbf{N}_i$.
- Let $\mathbf{A}$ be the $k \times (k + r)$ matrix obtained by concatenating the first column of each of the $\mathbf{C}_i$ matrices, and let $\mathbf{B}_i$ be the $k \times (k + r)$ matrix obtained by concatenating the $i + 1$st column of each of the $\mathbf{C}_i$ matrices. Output $\mathsf{M} = (\mathbf{A}, \mathbf{B}_1, \ldots, \mathbf{B}_n)$.

It is also straightforward to generalize the $\mathsf{VSS.Encode}$ procedure to take in an additional input r and produce an ADP with matrices of width $k \times (k + r)$. Simply draw the random $\mathbf{R}_i$ matrices over $\mathbb{F}_p^{k \times (k+r)}$.

The optimized witness encryption algorithm will additionally take as input a parameter λ which determines the correctness of the encryption, and the input message will now be $\mathbf{m} \in \{0, 1\}^h$. It operates as follows.

- $\mathsf{WE.Enc}(1^\lambda, \mathbf{m}, (\mathbf{H}, \boldsymbol{\ell}))$: Let $d \times n$ be the dimension of $\mathbf{H}$, and h be the bit length of $\mathbf{m}$.
  1. Choose a prime $p > \max_{\mathbf{x} \in \{0,1\}^n} ||\mathbf{H} \cdot \mathbf{x}||_\infty$.
  2. Set $t = \lceil h / \log(p) \rceil$ and $r = \lambda + t$.
  3. Sample $(\mathsf{M_{AA}}, k) \leftarrow \mathsf{AllAcc.Gen}^{\mathsf{NSS}}(n, r, \mathbb{F}_p)$.
  4. Sample $\mathsf{M}_{\mathbf{H},\boldsymbol{\ell}} \leftarrow \mathsf{VSS.Encode}((\mathbf{H}, \boldsymbol{\ell}), \mathbb{F}_p, k, r)$.
  5. Encode $\mathbf{m} \in \{0, 1\}^h$ as a vector $\hat{\mathbf{m}} \in \mathbb{F}_p^t$. Let $\mathbf{X}_{\hat{\mathbf{m}}}$ be the matrix of dimension $k \times (k+r)$ whose only non-zero entries consist of the vector $\hat{\mathbf{m}}$, placed in the last t positions of the first row.
  6. Output $\mathsf{M_{AA}} + \mathsf{M}_{\mathbf{H},\boldsymbol{\ell}} + \mathbf{X}_{\hat{\mathbf{m}}}$.
- $\mathsf{WE.Dec}(\mathsf{M}, \mathbf{w})$: Compute $\mathbf{A}_\mathbf{w} = \mathbf{A} + \sum_i w_i \mathbf{B}_i$. If the first $k + \lambda$ columns of $\mathbf{A}_\mathbf{w}$ do not form a matrix of rank exactly $k - 1$, abort. Otherwise, find a rank $k - 1$ submatrix $\mathbf{A}'_\mathbf{w}$ of dimension $k \times (k - 1)$ among the first $k + \lambda$ columns of $\mathbf{A}_\mathbf{w}$. Let $\mathbf{X}$ be the matrix of all zeros except for a formal variable x in the top right corner. Now, for each $i \in [t]$, let $\hat{m}_i$ be the solution to the linear equation $\det([\mathbf{A}'_\mathbf{w} \mid (\mathbf{A}_\mathbf{w})_{k+\lambda+i}] - \mathbf{X}) = 0$ over the variable x. Assemble the $\hat{m}_i$ into a vector $\hat{\mathbf{m}} \in \mathbb{F}_p^t$ and recover the message $\mathbf{m} \in \{0, 1\}^h$.

To argue correctness of decryption on input a valid witness $\mathbf{w}$, first note that since $\mathbf{A}'_\mathbf{w}$ has full rank $k - 1$, each equation $\det([\mathbf{A}'_\mathbf{w} \mid (\mathbf{A}_\mathbf{w})_{k+\lambda+i}] - \mathbf{X}) = 0$ will be linear with a non-zero coefficient on variable x. Hence, each element of $\hat{\mathbf{m}}$ can easily be recovered. It is left to argue that the decryption function aborts with negligible probability. Let $\overline{\mathbf{A}}_\mathbf{w}$ be the first $k + \lambda$ columns of $\mathbf{A}_\mathbf{w}$. We claim that the rank of $\overline{\mathbf{A}}_\mathbf{w}$ is $k - 1$, except with negligible probability. We know it is not full rank, since by the properties of the NSS all-accept scheme, the vector $[1 \mid \mathbf{w}] \cdot \mathbf{T}$ must be in its left kernel. Similar to the proof of security of the original NSS all accept, we can union bound over all other possible vectors, concluding that $\overline{\mathbf{A}}_\mathbf{w}$ is rank deficient by at least 2 with probability at most $1/p^\lambda$, which is negligible in λ, even for constant size fields.

Now we estimate the ciphertext size of the public key encryption scheme that results from combining the optimized witness encryption scheme with our reduction from $\mathsf{GOLDREICH}$ to $\mathsf{VECTOR\text{-}SUBSET\text{-}SUM}$. We will set $\lambda = 80$ to be the security and correctness parameter. Note that for any public key encryption scheme, it suffices to encrypt a security parameter number of bits and then appeal to key encapsulation to encrypt arbitrarily long messages. We set $p = 11$ to be the smallest prime satisfying $p > \max_{\mathbf{x} \in \{0,1\}^n} ||\mathbf{H} \cdot \mathbf{x}||_\infty$ for $\mathbf{H}$ output by the reduction from $\mathsf{GOLDREICH}$.

We will make use of Goldreich's PRG[3] with input length $n = 300$ and output length $m = 300$ [15]. Plugging in these parameters, the optimized witness encryption will output $n + 2m = 900$ matrices of dimension 900×1007 over the field of size 11. Assuming 4 bits per field element, this comes to a ciphertext of size about 400 MB. Again, the *public key* size will be extremely small: 300 bits. Perhaps even more impressively, the key generation can be done by a Boolean circuit with 300 AND gates and 900 XOR gates. This can be useful for fast distributed generation of keys via secure multiparty computation. We are not aware of any other (practically feasible) public-key encryption schemes whose key generation is nearly as efficient.

7 Cryptanalysis of the Witness Encryption Candidate

Recall that in our generic witness encryption framework of Section 5, we encrypt a bit b with respect to an instance $(\mathbf{h}, \ell)$ by sampling the following 3 ADPs.

- An instance-encoding ADP

$$\mathsf{M}_{\mathbf{h},\ell} = (\mathbf{A}^{(\mathbf{h},\ell)}, \mathbf{B}_1^{(\mathbf{h},\ell)}, \ldots, \mathbf{B}_n^{(\mathbf{h},\ell)}).$$

- A bit-encoding ADP

$$\mathsf{M}_b = (b\mathbf{S}, 0, \ldots, 0).$$

- An all-accept ADP

$$\mathsf{M}_{\mathsf{AA}} = (\mathbf{A}^{(\mathsf{AA})}, \mathbf{B}_1^{(\mathsf{AA})}, \ldots, \mathbf{B}_n^{(\mathsf{AA})}) \leftarrow \mathsf{AllAcc.Gen}(n, \mathbb{F}_p).$$

The ciphertext is the ADP

$$\mathsf{M}_{\mathbf{h},\ell} + \mathsf{M}_b + \mathsf{M}_{\mathsf{AA}} = (\mathbf{A}^{(\mathbf{h},\ell)} + b\mathbf{S} + \mathbf{A}^{(\mathsf{AA})}, \mathbf{B}_1^{(\mathbf{h},\ell)} + \mathbf{B}_1^{(\mathsf{AA})}, \ldots, \mathbf{B}_n^{(\mathbf{h},\ell)} + \mathbf{B}_n^{(\mathsf{AA})}).$$

A described in Section 5, $\mathsf{M}_{\mathbf{h},\ell} + \mathsf{M}_b$ on its own would be insufficient for security, since given any vector $\mathbf{w} \in \mathbb{F}_p^n$ such that $\mathbf{h} \cdot \mathbf{w} = \ell$, it is possible to recover b. For security, recovering b should only be possible when the vector $\mathbf{w}$ additionally satisfies $\mathbf{w} \in \{0,1\}^n$.

We therefore include the "all-accept" ADP M_{AA} such that for any $\mathbf{x} \notin \{0,1\}^n$, $\det(\mathsf{M}_{\mathsf{AA}}(\mathbf{x})) \neq 0$ with overwhelming probability, and for any $\mathbf{x} \in \{0,1\}^n$, $\det(\mathsf{M}_{\mathsf{AA}}(\mathbf{x})) = 0$. We note that both $\mathsf{M}_{\mathbf{h},\ell} + \mathsf{M}_b$ and M_{AA} on their own are "insecure" in the sense that we can show *randomness recovery* attacks on the procedures used to sample them. However, security relies on the idea that adding these ADPs to each other may block such attacks.

First, we show how to mount a randomness recovery attack on M_{AA}. On its own, this will not constitute an attack on the witness encryption ciphertext. However, we will show that this attack strategy can be extended to a full attack on a simplified version of our concrete witness encryption candidate, without the additional safeguards.

Indeed, to obtain secure witness encryption for VECTOR-SUBSET-SUM, we need semantic security of ciphertexts to hold for *any* instance $(\mathbf{h}, \ell)$ that is not in the language. A toy NO instance to consider is

$$\mathbf{h} = (0, \ldots, 0, 1), \ell = 2.$$

[3] This function is conjectured to be *one-way* when the output length is equal to the input length. Here we use it as a one-way function, which by Claim 15 suffices if our WE candidate is extractable. The latter assumption is only sufficient but not necessary; in fact, the security of this PKE candidate is a relatively clean and falsifiable assumption that can serve as a good target for cryptanalysis.

Intuitively, such an instances is particularly vulnerable since the resulting $\mathsf{M}_{\mathbf{h},\ell} + \mathsf{M}_b + \mathsf{M}_{\mathsf{AA}}$ will give out all but 2 of the matrices of M_{AA} in the clear.

Attacks on this instance motivate the concrete candidate given in Section 5, in which we first encode the instance $(\mathbf{h}, \ell)$ into a VECTOR-SUBSET-SUM instance $(\mathbf{H}, \boldsymbol{\ell})$ where $\mathbf{H}$ consists of many linearly independent rows.

7.1 Formula-Based All-Accept

We will consider two attacks on the $\mathsf{AllAcc.Gen}^{\mathsf{FORM}}$ sampling procedure, where the width of each $\mathbf{u}_i, \mathbf{v}_i, \mathbf{s}_i, \mathbf{t}_i$ is set to 1. The second attack in particular will motivate the need to consider a more general width parameter $q(\cdot)$.

7.1.1 Correlated Span Attack

We first show a polynomial-time algorithm that given $\mathsf{M} = (\mathbf{A}, \mathbf{B}_1, \ldots, \mathbf{B}_n) \leftarrow \mathsf{AllAcc.Gen}(n, \mathbb{F}_p)$, recovers all $4n$ vectors $\{\mathbf{u}_i, \mathbf{v}_i, \mathbf{s}_i, \mathbf{t}_i\}_{i\in[n]}$ up to scalings. The high-level intuition for the attack is that for any $\mathbf{x} \in \{0,1\}^n$, the matrix $\mathbf{A} + \sum_i x_i \mathbf{B}_i$ is rank-deficient. Moreover, if we look at the kernel of this matrix for many different choices of $\mathbf{x} \in \{0,1\}^n$, correlations between these kernels will allow us to recover the original randomness used to sample $(\mathbf{A}, \mathbf{B}_1, \ldots, \mathbf{B}_n)$. We refer to this as a *correlated span attack*.

More precisely, the algorithm proceeds in the following steps.

1. Observe that $\mathbf{A}$ is rank deficient by 1. Thus, we can find vectors $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ such that $\mathbf{A} \cdot \boldsymbol{\alpha} = \mathbf{0}$ and $\boldsymbol{\beta}^\top \cdot \mathbf{A} = \mathbf{0}$.
2. Observe that with high probability $\boldsymbol{\alpha} \perp \mathbf{v}_i$ for all $i \in [n]$. Similarly, $\boldsymbol{\beta} \perp \mathbf{u}_i$ for all $i \in [n]$.
3. Now to recover $\mathbf{t}_i$ up to a scalar, compute $\boldsymbol{\beta}^\top \cdot \mathbf{B}_i$. Since $\boldsymbol{\beta}$ is perpendicular to $\mathbf{u}_i$ for all $i \in [n]$, $\boldsymbol{\beta}^\top \cdot \mathbf{B}_i = (\boldsymbol{\beta}^\top \cdot \mathbf{s}_i) \cdot \mathbf{t}_i$.
4. Now find a vector $\boldsymbol{\gamma}$ that is perpendicular to $\mathbf{t}_i$ for $i \in [n]$. Then, one can compute $\mathbf{B}_i \cdot \boldsymbol{\gamma}$ to find a vector that is a scalar multiple of $\mathbf{u}_i$. Continuing this way, one can recover all vectors $\mathbf{u}_i, \mathbf{v}_i, \mathbf{s}_i, \mathbf{t}_i$ for $i \in [n]$ up to scalar multiples.

Note that this attack crucially relies on access to the rank deficient matrix $\mathbf{A} + \sum_i x_i \mathbf{B}_i$ for various choices of $x \in \{0,1\}^n$. In the witness encryption candidate, it appears difficult to recover such a matrix when the instance is not in the language SUBSET-SUM, which is crucial for security.

7.1.2 Linearization Attack

Next we discuss linearization-style attacks [29, 14]. At a high level, a linearization attack first models the problem to be solved as a system of non-linear equations. Then, it attempts to find certain structures within the equations that enable solving them with just linear algebra. Generally, one derives new variables and re-expresses the system as a larger system of linear equations. Note that in general, systems of quadratic equations are difficult to solve. However, certain systems may be vulnerable to this style of attack.

For concreteness, we give a particular witness encryption instance that we are interested in attacking. We describe the output of encrypting a bit b under the SUBSET-SUM NO instance $(\mathbf{h}, \ell) = ((0, \ldots, 0, 1), 2)$, without first applying the transformation to VECTOR-SUBSET-SUM instance $(\mathbf{H}, \boldsymbol{\ell})$.

- $\mathsf{WE.Enc}(b, ((0,\ldots,0,1),2))$. Fix a large enough field $\mathbb{F}_p$ and draw at random 4 sets of n vectors of dimension $n+1$: $\{\mathbf{u}_i\}_{i\in[n]}, \{\mathbf{v}_i\}_{i\in[n]}, \{\mathbf{s}_i\}_{i\in[n]}, \{\mathbf{t}_i\}_{i\in[n]}$, along with n^2 scalars $\{c_j^{(i)}\}_{i,j\in[n]}$ over $\mathbb{F}_p$. Also draw uniformly random matrices $\mathbf{R}, \mathbf{S} \in \mathbb{F}_p^{(n+1)\times(n+1)}$. Let

$$\mathbf{A} = \sum_{i\in[n]} \mathbf{u}_i\mathbf{v}_i^\top - 2\mathbf{R} + b\mathbf{S},$$
$$\mathbf{B}_1 = -\mathbf{u}_1\mathbf{v}_1^\top + \mathbf{s}_1\mathbf{t}_1^\top + \sum_{j\in[n]} c_1^{(j)}\mathbf{u}_j\mathbf{t}_j^\top,$$
$$\vdots$$
$$\mathbf{B}_{n-1} = -\mathbf{u}_{n-1}\mathbf{v}_{n-1}^\top + \mathbf{s}_{n-1}\mathbf{t}_{n-1}^\top + \sum_{j\in[n]} c_{n-1}^{(j)}\mathbf{u}_j\mathbf{t}_j^\top,$$
$$\mathbf{B}_n = -\mathbf{u}_n\mathbf{v}_n^\top + \mathbf{s}_n\mathbf{t}_n^\top + \sum_{j\in[n]} c_n^{(j)}\mathbf{u}_j\mathbf{t}_j^\top + \mathbf{R}.$$

  Output

$$\mathsf{M} = (\mathbf{A}, \mathbf{B}_1, \ldots, \mathbf{B}_n).$$

First, observe that $\mathbf{A}$ and $\mathbf{B}_n$ are masked by $\mathbf{R}$, so we will attempt to extract information only from $\mathbf{B}_1, ..., \mathbf{B}_{n-1}$. Our goal is to recover vectors $(\mathbf{u}_1, ..., \mathbf{u}_n, \mathbf{s}_1)$. In the end, we will be left with $\Omega(n^6)$ equations of degree 3 over the the $O(n^2)$ variables that constitute these vectors. As discussed at the end of the section, these asymptotics fall in the range where successful linearization attacks may be mounted. Thus, we view this as evidence that the above simplified scheme requires the safeguards described in Section 5.2.

The idea can now be described as follows.

1. Compute $\mathbf{M}_i = \mathbf{B}_1 \cdot \mathbf{B}_i^{-1}$ for $i \in \{2,\ldots,n-1\}$. Denote matrix $\mathbf{P} = [\mathbf{u}_1, ..., \mathbf{u}_n, \mathbf{s}_1]$ and $\mathbf{Q} = [\mathbf{t}_1, ..., \mathbf{t}_n, -\mathbf{v}_1]^\top$. In this notation, $\mathbf{B}_1$ can be written as $\mathbf{P}\cdot\mathbf{D}_1\cdot\mathbf{Q}$ where $\mathbf{D}_1 \in \mathbb{F}_p^{(n+1)\times(n+1)}$ is:

$$\mathbf{D}_1 = \begin{bmatrix} c_1^{(1)} & 0 & \ldots & 0 & 1 \\ 0 & c_1^{(2)} & \ldots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \ldots & c_1^{(n)} & 0 \\ 1 & 0 & \ldots & 0 & 0 \end{bmatrix}$$

2. Now for $i \in \{2,\ldots,n-1\}$, $\mathbf{B}_i$ can be expressed as $\mathbf{P}\cdot\mathbf{D}_i\cdot\mathbf{Q} + \mathsf{LowRank}_i$. Here, $\mathbf{D}_i$ is described below, and $\mathsf{LowRank}_i$ is a matrix with rank at most 2.

$$\mathbf{D}_i = \begin{bmatrix} c_i^{(1)} & 0 & \ldots & 0 & 0 \\ 0 & c_i^{(2)} & \ldots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \ldots & c_i^{(n)} & 0 \\ 0 & 0 & \ldots & 0 & 0 \end{bmatrix}$$

3. Calculate $\mathbf{D}'_i \coloneqq \mathbf{D}_1^{-1} \cdot \mathbf{D}_i$ as

$$\mathbf{D}'_i = \begin{bmatrix} 0 & 0 & \dots & 0 & 0 \\ 0 & (c_1^{(2)})^{-1} c_i^{(2)} & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & (c_1^{(n)})^{-1} c_i^{(n)} & 0 \\ 0 & 0 & \dots & 0 & 0 \end{bmatrix}$$

4. Now, define $\mathbf{M}_i \coloneqq \mathbf{B}_1^{-1} \cdot \mathbf{B}_i$. Note that the attacker can compute all $\mathbf{M}_i$ for $i \in \{2, \dots, n-1\}$ in the clear. Observe that

$$\mathbf{Q} \cdot \mathbf{M}_i = \mathbf{Q} \cdot \mathbf{B}_1^{-1} \cdot \mathbf{B}_i = \mathbf{Q} \cdot \mathbf{Q}^{-1} \cdot \mathbf{D}'_i \cdot \mathbf{Q} + \mathbf{Q} \cdot \mathsf{LowRank}_i = \mathbf{D}'_i \cdot \mathbf{Q} + \mathsf{LowRank}'_i,$$

   where $\mathsf{LowRank}'_i$ is a matrix of rank at most 2.

5. Defining $\mathbf{Q}'_i \coloneqq \mathbf{D}'_i \cdot \mathbf{Q}$, we have that $\mathbf{Q} \cdot \mathbf{M}_i - \mathbf{Q}'_i$ has rank at most 2. Then, for any i, letting $\mathbf{Q}$ and $\mathbf{Q}'_i$ be matrices of formal variables, the attacker can set up $O(n^6)$ degree 2 equations over $2n^2$ variables. These arise by setting the determinant of each of the $O(n^6)$ submatrices of size 3×3 in $\mathbf{Q} \cdot \mathbf{M}_i - \mathbf{Q}'_i$ equal to zero.

   We also have more equations that arise from the fact that $\mathbf{Q}'_i$ has the following structure:

$$\mathbf{Q}'_i = \begin{bmatrix} 0 & 0 & \dots & 0 \\ (c_1^{(2)})^{-1} c_i^{(2)} \mathbf{Q}_{2,1} & (c_1^{(2)})^{-1} c_i^{(2)} \mathbf{Q}_{2,2} & \dots & (c_1^{(2)})^{-1} c_i^{(2)} \mathbf{Q}_{2,n+1} \\ \vdots & \vdots & & \vdots \\ (c_1^{(n)})^{-1} c_i^{(n)} \mathbf{Q}_{n,1} & (c_1^{(n)})^{-1} c_i^{(n)} \mathbf{Q}_{n,2} & \dots & (c_1^{(n)})^{-1} c_i^{(n)} \mathbf{Q}_{n,n+1} \\ 0 & 0 & \dots & 0 \end{bmatrix}$$

   Thus, for every $i_1, i_2 \in [2, \dots, n-1]$ and $j_1, j_2 \in [n+1], \mathbf{Q}'_{i,j_1} \cdot \mathbf{Q}_{i,j_2} = \mathbf{Q}'_{i,j_2} \cdot \mathbf{Q}_{i,j_1}$. Similarly, one can set up new equations for every $i \in [n]$. We can further impose that $\sum_i \mathbf{Q}_{i,i} = 1$ to ensure non-triviality.

6. It is not immediately clear that there are enough linearly independent equations to mount an attack. As long as we have a unique solution, Kipnis and Shamir [29] suggest that (heuristically) it should be possible to solve for $\mathbf{Q}$ and $\mathbf{D}'_i$ (up to scaling factors).[4] This argument can be generalized to any constant degree d. For a degree 3 system with n^2 variables, we need $0.1 \cdot n^6$ equations. A rudimentary counting argument suggests that if $\mathbf{B}_i$ are random, such a solution should not exist. We leave a more refined analysis of this attack to future work.

This attack suggests that we should set parameters so that $\mathsf{LowRank}$ matrices have a super-constant rank, or alternatively, that we do not allow the attacker to see many $\mathbf{B}_i$ matrices in the clear. The construction in Section 5.2 incorporates safeguards that address both, generalizing the width of the $\mathbf{u}_i, \mathbf{v}_i, \mathbf{s}_i, \mathbf{t}_i$ vectors via the parameter $q(\cdot)$, and encoding SUBSET-SUM instances into VECTOR-SUBSET-SUM instances with many linearly independent vectors.

[4] More specifically, for a degree-2 system of equation over n variables, roughly $0.1 \cdot n^2$ linearly independent quadratic equations suffice to recover an over-determined solution in polynomial time [29, 35].

7.2 NSS-Based All-Accept

One potential advantage that the NSS-based construction has over the formula-based construction is in the amount of randomness used to generate a sample. For dimension n, the formula-based sampling procedure uses $4(n+1)n + n^2 = O(n^2)$ uniformly random field elements to produce $n+1$ matrices of dimension $(n+1) \times (n+1)$. On the other hand, the NSS-based sampling procedure generates a fresh NSS matrix for each $i \in [n]$, each of which requires about $n^2/2$ fresh random variables. Thus the sampling procedure overall uses $O(n^3)$ random field elements to produce $n+1$ matrices of dimension $(n+1) \times (n+1)$.

However, the NSS matrices are very structured, which leads to randomness recovery attacks in certain settings, even when the attacker is not given the entire all-accept sample. Consider a sample $(\mathbf{A}, \mathbf{B}_1, \ldots, \mathbf{B}_n) \leftarrow \mathsf{AllAcc.Gen}^{\mathsf{NSS}}(n, \mathbb{F}_p)$, recalling that the width k of each matrix is $n+1$. The goal will be to recover the random matrix $\mathbf{T}$ drawn during the sampling procedure. First assume that just $\mathbf{B}_1$ and $\mathbf{B}_2$ are given in the clear. This means we have the 2nd and 3rd column of each $\mathbf{C}_i$ matrix for $i \in [k]$, where $\mathbf{N}_i = \mathbf{T} \cdot \mathbf{C}_i$ and each $\mathbf{N}_i$ is NSS. The symmetric properties of each $\mathbf{N}_i$ immediately give us k linear equations over the $2k$ variables comprising the 2nd and 3rd row of $\mathbf{T}$. Generalizing, if we started with s matrices $\mathbf{B}_i$ in the clear, we could form $\binom{s}{2}$ linear equations over $2s$ variables of $\mathbf{T}$ and eventually the linear system will be over-determined.

We can use this structure to attack the $(\mathbf{h}, \ell) = ((1, 0, \ldots, 0), 2)$ instance of SUBSET-SUM as follows (note this is the same instance considered above but relabeled for convenience). Encryption of the bit 0 results in the set of matrices $\mathbf{A}' = \mathbf{A} + 2\mathbf{R}, \mathbf{B}'_1 = \mathbf{B}_1 + \mathbf{R}, \mathbf{B}_2, \ldots, \mathbf{B}_n$, where $(\mathbf{A}, \mathbf{B}_1, \ldots, \mathbf{B}_n) \leftarrow \mathsf{AllAcc.Gen}^{\mathsf{NSS}}(n, \mathbb{F}_p)$ and $\mathbf{R}$ is a uniformly random matrix. Using $\mathbf{B}_2, \ldots, \mathbf{B}_n$, recover all but the first two rows of $\mathbf{T}$ as described above, and call this matrix $\widetilde{\mathbf{T}} \in \mathbb{F}_p^{(k-2)\times k}$. We will be interested in solving for the first and second rows of $\mathbf{T}$, denoted $\mathbf{t}^{(1)}$ and $\mathbf{t}^{(2)}$, using $\widetilde{\mathbf{T}}$ and the matrices $\mathbf{A}'$ and $\mathbf{B}'_1$.

Note that $\widetilde{\mathbf{T}}$ and $\mathbf{B}_2, \ldots, \mathbf{B}_n$ reveal the bottom right $(k-2) \times (k-2)$ submatrix of each $\mathbf{N}_i$, and in particular the diagonal entries of each of these submatrices. Now let $\mathbf{d}_i$ be the column vector that consists of the final $k-2$ elements on the diagonal of $\mathbf{N}_i$, and arrange the k columns $\mathbf{d}_i$ into a matrix $\widetilde{\mathbf{N}}$ of dimension $(k-2)$ by k. Using the symmetries present in the $\mathbf{N}_i$, we see that

$$\widetilde{\mathbf{T}} \cdot \mathbf{A} = \begin{bmatrix} -\mathbf{t}^{(1)} \cdot \mathbf{B}_2 \\ \vdots \\ -\mathbf{t}^{(1)} \cdot \mathbf{B}_n \end{bmatrix} - \widetilde{\mathbf{N}} := \widetilde{\mathbf{T}}^{(1)} - \widetilde{\mathbf{N}} \quad \text{and} \quad \widetilde{\mathbf{T}} \cdot \mathbf{B}_1 = \begin{bmatrix} -\mathbf{t}^{(2)} \cdot \mathbf{B}_2 \\ \vdots \\ -\mathbf{t}^{(2)} \cdot \mathbf{B}_n \end{bmatrix} := \widetilde{\mathbf{T}}^{(2)},$$

and we can then compute

$$\widetilde{\mathbf{T}}(\mathbf{A}' + 2\mathbf{B}'_1) = \widetilde{\mathbf{T}}(\mathbf{A} - 2\mathbf{R} + 2\mathbf{B}_1 + 2\mathbf{R}) = \widetilde{\mathbf{T}}(\mathbf{A} + 2\mathbf{B}_1) = \widetilde{\mathbf{T}}^{(1)} - 2\widetilde{\mathbf{T}}^{(2)} - \widetilde{\mathbf{N}}.$$

This gives k^2 linear equations over the $2k$ variables comprising $\mathbf{t}^{(1)}$ and $\mathbf{t}^{(2)}$.

This attack again highlights the danger of giving the adversary many $\mathbf{B}_i$ matrices in the clear. This motivates the need for encoding the SUBSET-SUM instance $(\mathbf{h}, \ell)$ into a VECTOR-SUBSET-SUM instance $(\mathbf{H}, \boldsymbol{\ell})$ where $\mathbf{H}$ contains many linearly independent rows, as described in the concrete candidate above. Now an adversary attacking the $(\mathbf{h}, \ell) = ((1, 0, \ldots, 0), 2)$ instance can recover matrices $\mathbf{B}_1 - \mathbf{B}_{n/2+1}, \ldots, \mathbf{B}_{n/2} - \mathbf{B}_n$ in the clear, and about half the information in $\mathbf{T}$, but it is unclear how to use this limited information to mount a full randomness recovery or message distinguishing attack.

8 Candidate Obfuscation for Branching Programs

As discussed in Section 4.3, the ADP model is powerful enough to capture different types of branching programs. In this section, we describe a general paradigm for taking an arbitrary deterministic branching program BP and producing an ADP that is plausibly an *indistinguishability obfuscation* ($i\mathcal{O}$) of BP. Since polynomial-size deterministic branching programs are powerful enough to simulate the class NC^1 (by Barrington's theorem) or even log-space computations, our construction directly yields a candidate $i\mathcal{O}$ for NC^1 and log-space. As shown in [17], this suffices to obtain a candidate $i\mathcal{O}$ for all polynomial-size circuits.

8.1 Functionality-Preserving Transformations

In this section, we describe three generic transformations that can be applied to affine determinant programs satisfying specific conditions. Our candidate NC^1 obfuscation will be the result of applying these three transformations in sequence to an ADP encoding of [28, 4].

8.1.1 Transformation 1: Random Local Substitutions

Recall the ADP encoding of counting branching programs described in Section 4.3. Consider any such branching program, specified by a directed acyclic graph $G = (V, E)$, and fix a topological ordering on the vertices $v_1, \ldots, v_{|V|}$. Each pair of ordered vertices (v_j, v_k), for $j < k$, is labeled by a function x_i, $\neg x_i$, 1, or 0 (no edge). Given such a branching program, our first transformation will perform what we term a *random local substitution* for each vertex pair (v_j, v_k). We describe in Section 9.3 an attack strategy that motivates the need for random local substitutions.

Viewing the branching program as $\mathsf{ADP} = (\mathsf{M}, \mathsf{Eval})$, where $\mathsf{M} = (\mathbf{A}, \mathbf{B}_1, \ldots, \mathbf{B}_n)$, we see that each vertex pair (v_j, v_k), for $j < k$, defines a width-1 ADP. For concreteness, these matrices can be thought to be over some finite field $\mathbb{F}_p$ and Eval is just the identity function. In particular, each pair gives rise to the $(j, k-1)$th entry of each matrix $\mathbf{A}, \mathbf{B}_1, \ldots, \mathbf{B}_n$, which we denote by $a^{(j,k)}, b_1^{(j,k)}, \ldots, b_n^{(j,k)}$, as follows. If the label is the bit 0 or 1, then $a^{(j,k)}$ is equal to the bit, and all $b_i^{(j,k)} = 0$. If the label is x_i, then $a^{(j,k)} = 0, b_i^{(j,k)} = 1$, and $b_{i'}^{(j,k)} = 0$ for $i' \neq i$. If the label is $\neg x_i$, then $a^{(j,k)} = 1, b_i^{(j,k)} = -1$, and $b_{i'}^{(j,k)} = 0$ for $i' \neq i$.

We will inject entropy into the branching program by replacing each of these width-1 ADPs with a random width-2 ADP computing the same function. There are many possible such local substitution operations and generalizations, but we present here a particularly simple realization, which maintains the property that the resulting matrices have all entries in $\{-1, 0, 1\}$.

In the graph representation of the original branching program, the effect of replacing each each width-1 ADP with a width-2 ADP amounts to adding a vertex $v_{j,k}$ for each pair (v_j, v_k). Thus, if the width of the original ADP was ℓ, the width of the resulting ADP $\mathbf{A}', \mathbf{B}'_1, \ldots, \mathbf{B}'_n$ will be $\ell + \binom{\ell+1}{2}$. With this view it is easy to see what transformations are possible. For any pair (v_j, v_k), we consider the set of 2×2 submatrices of $\mathbf{A}', \mathbf{B}'_1, \ldots, \mathbf{B}'_n$ with rows indexed by $v_j, v_{j,k}$ and columns indexed by $v_{j,k}, v_k$. Denote this set of 2×2 matrices as $\mathbf{A}'^{(j,k)}, \mathbf{B}_1'^{(j,k)}, \ldots, \mathbf{B}_n'^{(j,k)}$. If the label of (v_j, v_k) was 0, we set all $\mathbf{B}_i'^{(j,k)} = \mathbf{0}$, and have the following possibilities for $\mathbf{A}'^{(j,k)}$:

$$\mathbf{A}_1^{(0)} = \begin{bmatrix} 0 & 0 \\ -1 & 0 \end{bmatrix}, \mathbf{A}_2^{(0)} = \begin{bmatrix} 1 & 0 \\ -1 & 0 \end{bmatrix}, \mathbf{A}_3^{(0)} = \begin{bmatrix} 0 & 0 \\ -1 & 1 \end{bmatrix}.$$

If the label of (v_j, v_k) was 1, we set all $\mathbf{B}_i^{(j,k)} = \mathbf{0}$, and have the following possibilities for $\mathbf{A}^{(j,k)}$:

$$\mathbf{A}_1^{(1)} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}, \mathbf{A}_2^{(1)} = \begin{bmatrix} 1 & 1 \\ -1 & 0 \end{bmatrix}, \mathbf{A}_3^{(1)} = \begin{bmatrix} 0 & 1 \\ -1 & 1 \end{bmatrix}, \mathbf{A}_4^{(1)} = \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix}.$$

If the label of (v_j, v_k) was x_i, we can set $\mathbf{A}'^{(j,k)}$ to be $\mathbf{A}_c^{(0)}$ for some $c \in \{1, 2, 3\}$, and $\mathbf{B}_i'^{(j,k)}$ to be $\mathbf{A}_d^{(1)} - \mathbf{A}_c^{(0)}$ for some $d \in \{1, 2, 3, 4\}$. This gives a total of 12 possible substitutions. Finally, if the label of (v_j, v_k) was $1 - x_i$, we again obtain 12 possible substitutions by fixing $\mathbf{A}'^{(j,k)}$ to be $\mathbf{A}_c^{(1)}$ for some $c \in \{1, 2, 3, 4\}$, and $\mathbf{B}_i'^{(j,k)}$ to be $\mathbf{A}_d^{(0)} - \mathbf{A}_c^{(1)}$, for some $d \in \{1, 2, 3\}$.

The operation we perform will, for each vertex pair, pick uniformly at random from the set of possibilities described above. For convenience, we denote this *random local substitution* operation by $\mathsf{ADP}' = \mathsf{RLS}(\mathsf{ADP})$.

We stress that our particular method of performing random local substitutions is only one potential candidate, and there are many possible transformations to explore. Our candidate was designed specifically to thwart attacks on iO described in Section 9.3, which take advantage of the fact that the structure of the underlying branching program is known.

8.1.2 Transformation 2: Small Even-Valued Noise

Our next transformation assumes the following about the input $\mathsf{ADP} = (\mathsf{M}, \mathsf{Eval})$.

- $\mathsf{M} = (\mathbf{A}, \mathbf{B}_1, \ldots, \mathbf{B}_n)$ is such that each entry of $\mathbf{A}, \mathbf{B}_1, \ldots, \mathbf{B}_n$ is in $\{-1, 0, 1\}$.
- $\mathsf{Eval} = \mathsf{Eval}_{\neq 0}$, and furthermore, on any input $\mathbf{x} \in \{0,1\}^n$, $\det(\mathbf{A} + \sum_i x_i \mathbf{B}_i) \in \{0, 1\}$.

Note that any ADP output by the [28, 4] encoding and then subjected to the random local substitution above satisfies these properties.

$\mathsf{AddNoise}(\mathsf{ADP})$:

- Let ℓ be the width of the matrices in ADP and n be the input length. Based on n, ℓ, fix a prime modulus p and error distribution χ as explained below. Consider each matrix $\mathbf{A}, \mathbf{B}_i$ as now being over $\mathbb{F}_p^{\ell \times \ell}$.
- Sample matrices $\mathbf{U}, \mathbf{V}$ randomly in $\mathbb{F}_p^{\ell \times \ell}$ such that $\det(\mathbf{U}) \cdot \det(\mathbf{V}) = 1$.
- Sample matrices $\mathsf{Err}_i \leftarrow \chi^{\ell \times \ell}$.
- Set $\mathbf{A}' = \mathbf{U} \cdot (\mathbf{A} + 2 \cdot \mathsf{Err}_0) \cdot \mathbf{V}$ and $\mathbf{B}'_j = \mathbf{U} \cdot (\mathbf{B}_j + 2 \cdot \mathsf{Err}_j) \cdot \mathbf{V}$ for all $j \in [n]$.
- Finally, in the resulting ADP, set $\mathsf{M} = (\mathbf{A}', \mathbf{B}'_1, ..., \mathbf{B}'_n)$ and set $\mathsf{Eval} = \mathsf{Eval}_{\text{parity}}$.

Parameters

We set χ to be the distribution that samples uniformly from the range $[-B(\ell), +B(\ell)]$, where $B(\ell) = \ell^{\omega(1)}$. We set p to be a prime modulus such that p is $\Theta((n \cdot B(\ell) \cdot \sqrt{\ell})^\ell)$. This can be done by setting the bit length of p as $\ell^{1+\epsilon}$ for any constant $\epsilon > 0$, assuming $\ell > n$.

Correctness

Let $\mathbf{x}$ be any input in $\{0,1\}^n$ and let $L(\mathbf{x}) \coloneqq \mathbf{A} + \sum_i x_i \mathbf{B}_i$ be the branching program that is being obfuscated. Observe that

$$\begin{aligned} \det(\mathsf{M}(\mathbf{x})) &= \det(\mathbf{U} \cdot (L(\mathbf{x}) + 2 \cdot \mathsf{Err}_0 + \sum_i x_i \cdot 2 \cdot \mathsf{Err}_i) \cdot \mathbf{V}) \\ &= \det(\mathbf{U}) \det(\mathbf{V}) \det(L(\mathbf{x}) + 2 \cdot \mathsf{Err}_0 + \sum_i x_i \cdot 2 \cdot \mathsf{Err}_i) \\ &= \det(L(\mathbf{x}) + 2(\mathsf{Err}_0 + \sum_i x_i \cdot \mathsf{Err}_i)). \end{aligned}$$

Correctness follows from the following observations.

- The value of $\det(L(\mathbf{x}) + 2(\mathsf{Err}_0 + \sum_i x_i \cdot \mathsf{Err}_i))$ over the integers is of the form $L(\mathbf{x}) + 2z$ for some $z \in \mathbb{Z}$.
- The number of monomials in the degree ℓ polynomial defined by the determinant is (loosely) bounded by ℓ^ℓ.
- Each entry in $L(\mathbf{x}) + 2(\mathsf{Err}_0 + \sum_i x_i \cdot \mathsf{Err}_i)$ is bounded by $n \cdot B(\ell) + 1$ in absolute value, so the determinant is smaller than $(n \cdot B(\ell)+1)^\ell \cdot \ell^\ell$ in absolute value. If $p > (n \cdot B(\ell)+1)^\ell \cdot \ell^\ell$, correctness holds.

One can also carry out a more refined calculation as follows. For a boolean input x, let $Q_x = L(\mathbf{x}) + 2(\mathsf{Err}_0 + \sum_i x_i \cdot \mathsf{Err}_i)$. With overwhelming probability over the choice of x, Q_x has entries bounded by $n^{1/2+\delta}B(\ell)$ in absolute value for any $\delta > 0$. This can be shown using a Chernoff bound. The signs of the entries of this matrix Q_x, are random and independent, so one can use the matrix Bernstein inequality (see for example, Theorem 6.6.1 [36]) to bound the maximum eigenvalue and hence the determinant (as determinant is the product of eigenvalues), to be $(n^{1/2+\delta} \cdot B(\ell) \cdot \ell^{1/2+\delta})^\ell$. This implies that p can be a $\Theta(\ell^{1+\epsilon})$ bit prime for any $\epsilon > 0$.

8.1.3 Transformation 3: Block-Diagonal Matrices

Ideally, the obfuscation of a circuit over n bits should not leak anything other than its input/output behavior on $\mathbf{x} \in \{0,1\}^n$. However, consider evaluating the ADP that results from the above transformation on a short but non-binary input $\mathbf{x}$. Due to the setting of parameters necessary for correctness, the determinant of the matrix $\mathbf{A} + \sum_i x_i \mathbf{B}_i$ will not be large enough to wrap around the modulus. Intuitively, the even-valued noise should ensure that the only useful information gained from this determinant is the evaluation of the circuit on input $\mathbf{x}$ mod 2. However, the fact that the determinant does not wrap around p is nevertheless worrisome, and we present a simple method to potentially block any attacks that might make use of short non-binary evaluations, such as the polynomial extension attacks described in Section 9.1.

The idea will be to post-process any $\mathsf{M} = (\mathbf{A}, \mathbf{B}_1, \ldots, \mathbf{B}_n)$ in a way that forces the determinant on non-binary inputs to be large and random. This can be accomplished using $2n$ random matrices $\{\mathbf{G}_i, \mathbf{H}_i\}_{i\in[n]}$ of determinant 1. We will append each $\mathbf{G}_i$ to $\mathbf{A}$ along the diagonal, and then append $\mathbf{H}_i - \mathbf{G}_i$ to $\mathbf{B}_i$ in the ith slot along the diagonal. After appending, we rerandomize as before with $\mathbf{U}$ and $\mathbf{V}$. The determinant on any binary input will be the product of the determinant of the original ADP times the product of the determinant of $\mathbf{G}_i$ or $\mathbf{H}_i$ for each i, which are both 1. On any non-binary input, some block diagonal will be a linear combination of $\mathbf{G}_i$ and $\mathbf{H}_i$ that results in a large and random determinant. The following transformation also takes as input a parameter d, which determines the size of the random $\mathbf{G}_i, \mathbf{H}_i$ matrices. In our obfuscation construction, it is reasonable to set $d = 2$.

$\mathsf{AddBlockDiagonals}(\mathsf{ADP}, d)$:

- Let ℓ be the width of the matrices in ADP and n be the input length.
- Sample $2n$ matrices $\{\mathbf{G}_i, \mathbf{H}_i\}_{i\in[n]}$ uniformly from $\mathbb{F}_p^{d\times d}$ conditioned on their determinant being equal to 1.
- Sample $\mathbf{U}, \mathbf{V}$ uniformly from $\mathbb{F}_p^{(\ell+nd)\times(\ell+nd)}$ conditioned on $\det(\mathbf{U}) \cdot \det(\mathbf{V}) = 1$.
- Set $\mathbf{A}' = \mathbf{U} \cdot \mathsf{diag}(\mathbf{A}, \mathbf{G}_1, \ldots, \mathbf{G}_n) \cdot \mathbf{V}$, and $\mathbf{B}'_i = \mathbf{U} \cdot \mathsf{diag}(\mathbf{B}_i, \mathbf{0}, \ldots, \mathbf{0}, \mathbf{H}_i - \mathbf{G}_i, \mathbf{0}, \ldots, \mathbf{0}) \cdot \mathbf{V}$, and return the resulting ADP, consisting of $\mathsf{M} = (\mathbf{A}', \mathbf{B}'_1, \ldots, \mathbf{B}'_n)$ and $\mathsf{Eval} = \mathsf{Eval}_{\text{parity}}$.

In summary, to obfuscate a deterministic (alternatively, $\mathbb{F}_2$ counting) branching program, we first convert it into an ADP as described in Section 4.3. We then output $\mathsf{AddBlockDiagonals}(\mathsf{AddNoise}(\mathsf{RLS}(\mathsf{ADP})), 2)$ as the final obfuscation.

8.2 Extensions of the Basic Construction

8.2.1 Obfuscating Affine $\mathbb{F}_2$ Counting Branching Programs

Motivated by the goal of improved concrete efficiency, we extend the branching program model to allow labeling of edges by any *affine function* over the input bits. This is captured by the following notion of Affine $\mathbb{F}_2$ Counting Branching Programs.

▶ **Definition 19** (Affine Counting Branching Programs [28]). *An Affine $\mathbb{F}_2$ Counting Branching Program computing $f : \{0,1\}^n \to \{0,1\}$ is specified by a directed acyclic graph $G = (V, E)$ and a labeling $\phi(\cdot,\cdot)$ where each edge $(u, v) \in E$ is labeled with an affine function $\phi(u, v) = f_{u,v}(\mathbf{x})$, and two special source and sink vertices are labeled s and t respectively. $f_{u,v}$ has the form*

$$f_{u,v}(\mathbf{x}) = \sum_{i \in S_{u,v}} x_i + c_{u,v} \mod 2,$$

where $c_{u,v} \in \mathbb{F}_2$. Its size is $|V| - 1$. Any input $\mathbf{x} \in \{0,1\}^n$ induces a sub-graph $G_{\mathbf{x}}$ limited to edges consistent with $\mathbf{x}$ (i.e. edges that evaluate to 1 on $\mathbf{x}$). An accepting path on input $\mathbf{x}$ is a directed $s - t$ path in $G_{\mathbf{x}}$. An Affine $\mathbb{F}_2$ Counting Branching Program computes the function $g : \{0,1\}^n \to \{0,1\}$ such that $g(\mathbf{x})$ is the number of accepting paths in $G_{\mathbf{x}}$ mod 2.

We now import the following theorem.

▶ **Lemma 20** ([28, 4]). *Suppose there is an Affine $\mathbb{F}_2$ Counting Branching Program of size ℓ computing a boolean function f. Suppose $L(\mathbf{x})$ satisfies the following*

- *$L(\mathbf{x})$ has -1 along the second diagonal (right below the main diagonal).*
- *$L(\mathbf{x})$ is 0 below the second diagonal.*
- *Each entry of $L(\mathbf{x})$ is a degree (at most) 1 polynomial in a single input variable x_i. More precisely, $L(\mathbf{x})$ is defined as follows. Fix a topological ordering of the vertices in V, and label the columns / rows (from left to right / top to bottom) according to this ordering of vertices. In particular we want s labeled 1 and t labeled ℓ. We first define a matrix $A(\mathbf{x})$ of dimension $\ell \times \ell$. For entry (i, j) entry of $A(\mathbf{x})$ write affine function $\phi(i, j)$ (written as a degree one polynomial over the reals) if (i, j) is an edge in G and 0 otherwise. Note that $A(\mathbf{x})$ will be 0 on and below the main diagonal. Now consider $A(\mathbf{x}) - I$, and delete its first column and last row to obtain the $(\ell - 1) \times (\ell - 1)$ dimensional matrix $L(\mathbf{x})$.*

Then for all $\mathbf{x} \in \{0,1\}^n$, we have $\det(L(\mathbf{x})) \mod 2 = f(\mathbf{x})$.

Observe that this class of branching programs contain $\mathbb{Z}$ branching programs, since the evaluation of a $\mathbb{Z}$ branching program is exactly its evaluation modulo 2. What we observe is that in this case, our obfuscation scheme described in Section 8 is already capable of obfuscating affine $\mathbb{F}_2$ counting branching programs. The idea is that we can apply transformation 2 given in Section 8 to the matrix $L(\mathbf{x})$ representing an Affine $\mathbb{F}_2$ Counting Branching Program as follows: Each edge in $L(\mathbf{x})$ is an affine function of the form $f(\mathbf{x}) = \sum_{i \in S} x_i + c \mod 2$. We construct a new matrix $L'(\mathbf{x})$ where the edge is replaced by $f'(\mathbf{x}) = \sum_{i \in S} x_i + c$, and the computation is now over integers. The range of this new affine function is $[0, n + 1]$. If the parameters are chosen appropriately, what we end up computing by the evaluation procedure is $\det(L'(\mathbf{x})) \bmod 2$. We set p so that this can be computed without a wrap-around. Note that $\det(L(\mathbf{x})) \mod 2 = \det(L'(\mathbf{x})) \bmod 2$ as the error that is added to the matrices is even. This ensures correctness.

In the next section, we refine the notion of random local substitution and propose another alternative safeguard that relies on the ideas developed in this section.

8.2.2 Using Embedded Affine $\mathbb{F}_2$ Branching Programs

In Section 8.1.1, we discussed the notion of random local substitutions, which is our safeguard for preventing attacks such as the one described in Section 9.3. However, that solution incurs a quadratic blow up in the size of the branching program. In this section, we propose an alternate safeguard. Let $\mathsf{BP} : \{0,1\}^n \to \{0,1\}$ be a $\mathbb{Z}$ branching program with ℓ vertices, including source s and target t. First, sample a $\mathsf{BP}' : \{0,1\}^n \to \{0,1\}$ as follows:

- Fix ℓ vertices, and select two vertices s', t' among these ℓ vertices to act as the source vertex and target vertex of BP'.
- Induce a random directed acyclic graph among these vertices. This is done by connecting all vertices $i \in [\ell]$ to vertices j where $j \geq i+1$. In this numbering, the starting vertex s' is the 1st vertex and t' is the ℓ^{th} vertex. For every edge u, v sample a random affine function $f_{u,v}(\mathbf{x})$ which is used to label the edges.

We now combine BP and BP' into the final transformed BP'' as follows. BP'' consists of $2\ell+2$ vertices: the ℓ vertices of BP, the ℓ vertices of BP', and a new source s'' and target t''. Form edges from s'' to s, from s'' to s', and from t to t'', all with label 1. Note that there will be no edge between t' and t''. Observe that BP'' computes the same function as BP'. Indeed, any additional paths that go via BP' are never connected to the target vertex t''. There can be either 0 or 1 path going via BP since we started with a deterministic $\mathbb{Z}$ branching program. This resulting new BP'' can now be input to transformations 2 and 3 outlined in Section 8.

9 Cryptanalysis and Parameter Estimation for the Branching Program Obfuscation Candidate

9.1 Polynomial Extension Attacks

As an instructive example, consider the ADP obfuscation of a point function $\mathcal{I}_v$, without the even-valued error. Recall that on input $\mathbf{x} \in \{0,1\}^n$, $\mathcal{I}_v$ outputs 1 if $\mathbf{x} = \mathbf{v}$ and 0 otherwise. A simple branching program computing $\mathcal{I}_v$ contains $n+1$ vertices, with a single edge from each vertex i to vertex $i+1$ that is either set to v_i or $1-v_i$. Thus, $L(\mathbf{x})$ can be written as the following matrix:

$$\begin{bmatrix} 1+2v_1x_1-v_1-x_1 & 0 & \dots & \dots & 0 \\ -1 & 1+2v_2x_2-v_2-x_2 & \dots & \dots & 0 \\ 0 & -1 & \ddots & \ddots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & -1 & 1+2v_nx_n-v_n-x_n \end{bmatrix}$$

Hence, for $\mathbf{x} \in \{0,1\}^n$, $\det(L(\mathbf{x}))$ is 1 if $\mathbf{x} = \mathbf{v}$ and 0 otherwise. Writing

$$L(\mathbf{x}) = L_0 + x_1L_1 + \ldots + x_nL_n,$$

we see that an attacker given

$$\mathsf{ADP} = (\mathbf{A} = \mathbf{U} \cdot L_0 \cdot \mathbf{V}, \mathbf{B}_1 = \mathbf{U} \cdot L_1 \cdot \mathbf{V}, ..., \mathbf{B}_n = \mathbf{U} \cdot L_n \cdot \mathbf{V}),$$

can evaluate

$$\det(\mathbf{A} + \sum_i x_i\mathbf{B}_i) = \det(\mathbf{U} \cdot L(\mathbf{x}) \cdot \mathbf{V}) = \det(L(\mathbf{x})).$$

Thus, the attacker has input-output access to the polynomial $\det(L(\mathbf{x}))$ over the field $\mathbb{F}_p$, and is not restricted to evaluation on binary inputs $\mathbf{x} \in \{0,1\}^n$. This leads to the following attack.

1. Observe that $\det(\mathbf{A} + \sum_i x_i \mathbf{B}_i)$ is the unique multilinear polynomial over $\mathbb{F}_p^n$ agreeing with $\prod_{i\in[n]}(1 + 2v_i x_i - v_i - x_i)$ on inputs from $\{0,1\}^n$. This implies

$$\det(\mathbf{A} + \sum_i x_i \mathbf{B}_i) = \prod_{i\in[n]} (1 + 2v_i x_i - v_i - x_i)$$

   over $\mathbb{F}_p^n$.
2. To recover v_1, compute

$$\det(\mathbf{A} + x_1 \cdot \mathbf{B}_1 + \sum_{i\geq 2} 2^{-1}\mathbf{B}_i) = (1 - 2^{-1})^{n-1}(1 + 2v_1 x_1 - v_1 - x_1).$$

   We can find $x_1 = 1 - v_1$ by equating this quantity to 0. This recovers the first bit of the point function, and the others can be computed in the same way.

The reason that this particular attack succeeds is that the polynomial extending the boolean function $\det(L(\mathbf{x}))$ is *multilinear*. Read-once branching programs give rise to such polynomials. More generally, Klivans and Shpilka [30] showed that *any* read-once, oblivious branching program can be learned efficiently. In their model, the attacker is given membership and equivalence query access to an unknown polynomial P computed by a read once branching program over a field $\mathbb{F}_p$. This means the attacker is able to evaluate the program on any input $\mathbf{x} \in \mathbb{F}_p^n$, as well as submit a hypothesis H and learn that either H is equal to P or receive a point $\mathbf{y}$ on which $H(\mathbf{y}) \neq P(\mathbf{y})$. In our setting, we just argued that the attacker has membership query access to the polynomial computed by the branching program, and equivalence queries can be simulated by evaluation on random points and appealing to Schwartz-Zippel.

For our construction to satisfy the security notion of indistinguishablity obfuscation, we need to be able to successfully obfuscate simple classes of functions computable by read once branching programs. Thus, we have to include some noise in the obfuscation procedure that destroys the read once nature of any such program. One can view the random even-valued error as adding an edge between every pair of vertices in the branching program, labeled with a random, small, even linear combination of the entire input vector. The resulting program is very far from read-once, as evaluating every edge requires reading the entire input. Thus, learning results that apply to restricted classes on branching programs will not apply. As an example, [30] state that their techniques would not apply to a polynomial as simple as $f(x_1, \ldots, x_n) = \prod_{i=2}^n (x_1 + x_i)$.

9.2 The Need for Super-polynomial Error and Modulus

Now we present a contrived example consisting of two ADPs which will motivate the need for super-polynomial noise for transformation 2. While these specific ADPs will not arise from the [28] transformation, we will set our error-size to err on the side of caution.

The attack works as follows. Suppose we have an ADP $\mathbf{A}, \mathbf{B}_1, \ldots, \mathbf{B}_n$ that is in one of the two following forms.

- In the first form, $\mathbf{A}, \mathbf{B}_1, \ldots, \mathbf{B}_n = \mathbf{0}$.
- In the second form, $\mathbf{A} = \mathbf{0}$, and $\mathbf{B}_i$ for $i \in [n]$ is the matrix

$$\begin{bmatrix} 1 & 1 & \ldots & 1 \\ 0 & 0 & \ldots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \ldots & 0 \end{bmatrix}.$$

Now consider applying the AddNoise operation to one of the ADPs, hoping that the choice of ADP is masked by this operation. Assume that the even-valued error added by the AddNoise operation is actually chosen in $\{-2, 2\}$, so that each entry of the matrices $\mathsf{Err}_0, \mathsf{Err}_1, \ldots, \mathsf{Err}_n$ is uniform in $\{-1, +1\}$. Observe that in both cases $\det(\mathbf{A} + \sum_i x_i \mathbf{B}_i) = 0$ for all $\mathbf{x} \in \{0,1\}^n$, so the parity of the determinant will not reveal which ADP was chosen. A relevant result of [33] states that the determinant of a matrix where the entries are chosen uniformly at random from $\{-1, +1\}$ concentrates in absolute value around $\sqrt{n!}$. This fact can be used to estimate the determinant of the matrix $\mathbf{A} + \sum_i \mathbf{B}_i$ in both cases, which is the ADP evaluated on input $x = [1, \ldots, 1]$.

Case 1: The value $\det(\mathbf{A} + \sum_i \mathbf{B_i})$ is exactly $\det(2(\mathsf{Err}_0 + \sum_i x_i \cdot \mathsf{Err}_i))$, whose entries are independently signed, with standard deviation about $\sqrt{n}$. Thus, the determinant will concentrate around $\sqrt{n!} \cdot (\sqrt{n})^n$ in absolute value.

Case 2: To estimate the value $\det(\mathbf{A} + \sum_i \mathbf{B_i})$ in this case, we apply a Chernoff bound and find that the elements in the top row are typically in the range $[n - \sqrt{n}\log^2 n, n + \sqrt{n}\log^2 n]$. If we divide the top row by $\sqrt{n}$, we are left with a matrix whose entries are distributed as the matrix in case 1, with the only difference being that the top row is positively signed. Heuristically, this will result in $\det(\mathbf{A} + \sum_i \mathbf{B_i})$ being a $\sqrt{n}$ factor larger.

Thus, the magnitude of the determinant on input $[1, \ldots, 1]$ will give a distinguisher. Note however that neither ADP considered here is a valid [28, 4] encoding of a branching program. We chose these particular ADPs for simplicity, to illustrate the need for super-polynomial error; similar attacks can be carried out on ADPs that result from encodings of real branching programs.

Setting the Modulus

Once we fix the noise bound $B(\ell)$, we can choose p in a more refined manner as follows. Let $L = (L_0, ..., L_n)$ be the branching program to obfuscate. We would like to add noise to the matrices of L to form an ADP $(\mathbf{A}, \mathbf{B}_1, \ldots, \mathbf{B}_n)$, in such a way that for all $y_0, y_1, ..., y_n \in \{0,1\}$,

$$\left(\det(y_0\mathbf{A} + \sum_i y_i \mathbf{B}_i) \mod p\right) \mod 2 = \det(y_0 L_0 + \sum_i y_i L_i) \mod 2.$$

Let $t = \mathsf{poly}(\ell)$ be a parameter, and set $p = \Theta(t^\ell \cdot B(\ell)^\ell \cdot \sqrt{\ell!} \cdot \ell)$. Then we observe the following:

- If $t \geq n^2$, then with overwhelming probability we can correctly evaluate any boolean combination $y_0, y_1, ..., y_n \in \{0,1\}^{n+1}$. To see this, first note that the entries of $y_0\mathbf{A} + \sum_i y_i \mathbf{B}$ are randomly signed, and typically lie in $[-2 \cdot \sqrt{n+1}\log^2 n \cdot B(\ell), +2 \cdot \sqrt{n+1}\log^2 n \cdot B(\ell)]$ due to a Chernoff bound. Then, we can appeal to Theorem 1.1 from [33], stating that the determinant of any matrix $\mathbf{M}$ of dimension ℓ, with entries chosen uniformly at randomly from $\{-1, +1\}$, satisfies

$$\sqrt{\ell!} \cdot e^{-c\sqrt{\ell \log \ell}} \leq |\det(\mathbf{M})| \leq \sqrt{\ell!} \cdot \ell,$$

with overwhelming probability. Here $c > 0$ is any constant. In this case, the determinant will thus heuristically satisfy

$$|\det(y \cdot \mathbf{A} + \sum_i y_i \cdot \mathbf{B}_i)| \leq (2 \cdot B(\ell) \cdot \sqrt{n+1} \log^2 n)^\ell \sqrt{\ell!} \cdot \ell \leq p.$$

- Now, consider more general (non-binary) linear combinations $(y_0, y_1, \ldots, y_n)$. If the ℓ_1 norm of the combination satisfies $|y_0| + \sum_i |y_i| > t^3$, then it holds that

$$|\det(y_0\mathbf{A} + \sum_i y_i\mathbf{B}_i)| \geq p.$$

This overflow will hopefully help with security against attacks that evaluate the ADP on non-binary inputs. We can again observe heuristically from theorem 1.1 in [33]. If the weight is greater than t^3, then typically we expect each entries of the matrix to be randomly signed and having values larger than $t^{1.4} \cdot B(\ell)$. Thus with high probability,

$$|\det(y_0\mathbf{A} + \sum_i y_i\mathbf{B}_i)| \geq (t^{1.4} \cdot B(\ell))^\ell \sqrt{\ell!} \cdot e^{-c\sqrt{\ell \log \ell}} \geq p.$$

We set t to be n^2 to allow correct evaluation of boolean inputs. The above describes that setting the modulus appropriately can protect against attacks that involves evaluating programs on inputs of large (polynomial) weight. If the weight of the combination is bounded by some polynomial t, we claim the following.

▷ Claim 21. Fix any ADP $\mathbf{A}, \mathbf{B}_1, ..., \mathbf{B}_n$ of width ℓ with entries in $\{-1, 0, 1\}$. For any $x = (x_0, ..., x_n) \in \mathbb{Z}^{n+1}$, Let $L(\mathbf{x}) = x_0\mathbf{A} + \sum_i x_i\mathbf{B}_i$, and let $\mathsf{ADP}' = \mathsf{AddNoise}(\mathsf{ADP})$. Let $\mathsf{Err}_0, \mathsf{Err}_1, \ldots, \mathsf{Err}_n$ be the matrices drawn during the real $\mathsf{AddNoise}$ procedure (with a super-polynomial bound), and for $x \in Z^{n+1}$, let $\mathsf{Err}(x) := x_0 \cdot 2\mathsf{Err}_0 + \sum_i x_i \cdot 2\mathsf{Err}_i$. Then for any $x \in \mathbb{Z}^{n+1}$ where $\|x\|_1 < t$,

$$\Pr\left[\frac{|\det(x_0\mathbf{A}' + \sum_i x_i\mathbf{B}'_i)| - |\det(L(\mathbf{x})) \mod 2 + \det(\mathsf{Err}(x))|}{|\det(\mathsf{Err}(x))|} \leq \mathsf{negl}(\ell)\right] \geq 1 - \mathsf{negl}(\ell).$$

where the probability is taken over the randomness of generating $\mathsf{Err}_0, \mathsf{Err}_1, \ldots, \mathsf{Err}_n$.

This also shows that if the error is chosen from a super-polynomially large error distribution, then a distinguishing attack which only uses the magnitude of $\det(\mathsf{M}(\mathbf{x}))$ will not apply.

Proof. This follows from a result of Ipsen and Rehman [25] who showed that for any non-singular matrices $\mathbf{M}_1, \mathbf{M_2} \in \mathbb{R}^{\ell \times \ell}$,

$$|\det(\mathbf{M}_1 + \mathbf{M}_2) - \det(\mathbf{M}_1)|/|\det(\mathbf{M_1})| \leq \cdot \left(\kappa \cdot \frac{\|\mathbf{M}_2\|_2}{\|\mathbf{M}_1\|_2} + 1\right)^\ell - 1,$$

were $\|\mathbf{M}\|_2$ is the largest singular value of the matrix $\mathbf{M}$ and $\kappa := \|\mathbf{M}_1\|_2 \cdot \|\mathbf{M}_1^{-1}\|_2$ (κ is sometimes referred to as the condition number of $\mathbf{M}_1$). For a random matrix of size $\ell \times \ell$, the condition number is expected to be around $O(\ell)$ [12]. Now, we can substitute $\mathbf{M}_1$ as $\mathsf{Err}(x)$, $\mathbf{M}_2$ as $L(\mathbf{x})$. Observe that the maximum singular value of $\mathsf{Err}(x)$ will be at least $\Omega(B(\ell)\sqrt{\ell})$ (see Excercise 14, [34]). Since branching programs have small entries, their singular values are also bounded by $\|x\|_1 \cdot O(\ell) = O(t \cdot \ell)$. Thus, the right hand side of the previous equation is

$$\left(\kappa \cdot \frac{\|L(\mathbf{x})\|_2}{\|\mathsf{Err}(x)\|_2} + 1\right)^\ell - 1 = \left(O(\ell) \cdot \frac{O(t \cdot \ell)}{\Omega(B(\ell) \cdot \sqrt{\ell})} + 1\right)^\ell - 1.$$
$$\approx O(\ell^2 \cdot t \cdot B(\ell)^{-1})$$

Since $B(\ell)$ is super-polynomial in ℓ, this concludes the argument. Although above we talked about learning for any $\mathbf{y} \in \mathbb{Z}^{n+1}$ determinants of the form $\det(y_0\mathbf{A} + \sum_i y_i\mathbf{B}_i)$, however, in $\mathbb{Z}$ branching programs, whenever $y = 0 \mod 2$, then $\det(y_0 L_0 + \sum_i y_i L_i) = 0 \mod 2$ and thus such combinations should not reveal anything useful. Heuristically, Claim 21 suggests that the magnitude of $\det(y_0\mathbf{A} + \sum_i y_i\mathbf{B}_i)$ should only reveal information about $\det((y_0 \mod 2)L_0 + \sum_i (y_i \mod 2)L_i)$. ◀

9.3 The Need for Random Local Substitutions

Consider any fixed branching program $L(\mathbf{x}) = \mathbf{A} + \sum_i x_i\mathbf{B}_i$ of width ℓ. Suppose we obfuscate $L(\mathbf{x})$ but skip the random local substitution step. Let $\mathsf{Err}^{(0)}, \mathsf{Err}^{(1)}, \ldots, \mathsf{Err}^{(n)}$ be the $n+1$ error matrices drawn by the $\mathsf{AddNoise}$ transformation. Write each $\mathsf{Err}^{(i)}_{j,k} = 2e^{(i)}_{j,k}$. We can recover the parity of each $e^{(i)}_{j,k}$ as follows. Let $\mathbf{A}', \mathbf{B}'_1, \ldots, \mathbf{B}'_n$ be the ADP after obfuscation. For any input $\mathbf{x}$,

$$\det(\mathbf{A}' + \sum_i x_i\mathbf{B}'_i) \equiv \det(L(\mathbf{x})) + \sum_{j,k}(\sum_i x_i 2e^{(i)}_{j,k}) \det(L(\mathbf{x})_{(j,k)}) \bmod 4,$$

where $L(\mathbf{x})_{(j,k)}$ is the (j,k) minor of $L(\mathbf{x})$. Note that everything is known except the parities of $e^{(i)}_{j,k}$, so this gives a linear equation mod 2 over these $(n+1)\ell^2$ parities. We have an exponential number of equations of this form, and can eventually obtain $(n+1)\ell^2$ linearly independent equations.

This readily gives an attack on $i\mathcal{O}$. To distinguish between two known and functionally equivalent branching programs L_1, L_2, simply run the above attack assuming the underlying BP is L_1 and see if there exists a solution or not. Now, the random local substitution is meant to randomize the underlying branching program $L(\mathbf{x})$, so an attacker does not know all of the $L(\mathbf{x})_{(j,k)}$ values. Without these, the attacker cannot set up and solve the above system of linear equations.

10 Applications and Future Work

In future work, we aim to explore applications of optimized variants of our candidates. We give some of the more promising directions below. Since our focus here is on concrete efficiency, we make a heuristic leap of faith of treating our $i\mathcal{O}$ candidates as *ideal* obfuscation schemes for the purpose of these applications.

Concretely Efficient Obfuscation for Circuits

If the safeguards described are secure, then in order to obfuscate a branching program of size ℓ, we need about $O(\ell^{4+\epsilon})$ bits for any $\epsilon > 0$. In concrete terms, if the size of the branching program exceeds 2^{13} vertices, the size is already around 100 terabytes. This motivates the study of efficient bootstrapping mechanisms and simple "obfuscation complete" families of branching programs. In particular, we would like to understand if the SNARG-based bootstrapping approach from [11] or the PRF-based approach from [24, 2] are practically feasible for our candidates.

Obfuscating PRFs

Efficiently obfuscating PRFs and simple computations that employ them is a highly desirable goal for both theoretical and practical applications. For instance, the bootstrapping theorems of [24, 2] reduce the obfuscation of a circuit to multiple obfuscations of simple functions that each use a constant number of PRF calls. This and other applications motivate PRF

candidates that have small *affine* branching programs, which can be efficiently handled by our construction. One such candidate could be based on the work of [10]. They propose a weak PRF[5] candidate based on Learning With Rounding (LWR) modulo a constant-size composite, which can be computed by a linear-size deterministic branching program. They also suggest a heuristic for converting a weak PRF into a strong one by evaluating the weak PRF on a suitable encoding of the input; for this particular weak PRF candidate, a linear encoding over a prime that does not divide the LWR modulus seems like a natural choice. For instance, one could use a linear code over $\mathbb{F}_2$ and LWR modulo 15 or linear code over $\mathbb{F}_3$ and LWR modulo 10. This approach yields (strong) PRF candidates that can be evaluated by a linear-size *affine* branching program.

Optimally-Succinct Non-Interactive Arguments

The ability to efficiently obfuscate a PRF would give several compelling applications. One example, due to Sahai and Waters [32], is a succinct non-interactive argument (SNARG) with proof length that is the best one could hope for; namely, an s-bit proof suffices to obtain (roughly) 2^{-s} soundness error.

To give slightly more detail, we briefly recall the obfuscation-based SNARG of [32]. Given a PRF and relation circuit $R_{\mathcal{L}}$ for language $\mathcal{L}$, the crs consists of the obfuscations of two programs $C_{\mathcal{P}}$ and $C_{\mathcal{V}}$. $C_{\mathcal{P}}$ takes as input an instance witness pair (x, w) and outputs $\mathsf{PRF}_k(x)$ (where k is a hard-coded PRF key) if and only if $\mathcal{R}_{\mathcal{L}}(x, w) = 1$. $C_{\mathcal{V}}$, which has the same k hard-coded, takes as input (x, π) and outputs 1 if and only if $\mathsf{PRF}_k(x) = \pi$. A prover wishing to prove that $x \in \mathcal{L}$ simply evaluates $\mathsf{Obf}(C_{\mathcal{P}})$ on (x, w) to obtain $\pi = \mathsf{PRF}_k(x)$. The verifier on input x and proof π runs $\mathsf{Obf}(C_{\mathcal{V}})$ on (x, π) and accepts if the output is 1.

If the obfuscation is an *ideal* obfuscation, then forging a proof on $x \notin \mathcal{L}$ requires predicting $\mathsf{PRF}(k, x)$. If the PRF is exponentially-secure, then the SNARG soundness error is (within polynomial factors of) 2^{-s}.

References

1 Prabhanjan Ananth, Aayush Jain, Huijia Lin, Christian Matt, and Amit Sahai. Indistinguishability Obfuscation Without Multilinear Maps: New Paradigms via Low Degree Weak Pseudorandomness and Security Amplification. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2019, Part III*, LNCS, pages 284–332. Springer, Heidelberg, August 2019. doi:10.1007/978-3-030-26954-8_10.

2 Benny Applebaum. Bootstrapping Obfuscators via Fast Pseudorandom Functions. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part II*, volume 8874 of *LNCS*, pages 162–172. Springer, Heidelberg, December 2014. doi:10.1007/978-3-662-45608-8_9.

3 Benny Applebaum. Cryptographic Hardness of Random Local Functions - Survey. *Computational Complexity*, 25(3):667–722, 2016.

4 Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography in NC^0. In *45th FOCS*, pages 166–175. IEEE Computer Society Press, October 2004. doi:10.1109/FOCS.2004.20.

5 Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography with Constant Input Locality. In Alfred Menezes, editor, *CRYPTO 2007*, volume 4622 of *LNCS*, pages 92–110. Springer, Heidelberg, August 2007. doi:10.1007/978-3-540-74143-5_6.

6 Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (Im)possibility of Obfuscating Programs. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 1–18. Springer, Heidelberg, August 2001. doi:10.1007/3-540-44647-8_1.

[5] A weak PRF is only guaranteed to be indistinguishable from a random function given its evaluation on *uniformly random* points (as opposed to on adversarially chosen points, as in the usual definition of a PRF).

7 David A. Mix Barrington. Bounded-Width Polynomial-Size Branching Programs Recognize Exactly Those Languages in NC^1. In *18th ACM STOC*, pages 1–5. ACM Press, May 1986. doi:10.1145/12130.12131.

8 James Bartusek, Tancrède Lepoint, Fermi Ma, and Mark Zhandry. New Techniques for Obfuscating Conjunctions. In Vincent Rijmen and Yuval Ishai, editors, *EUROCRYPT 2019, Part III*, LNCS, pages 636–666. Springer, Heidelberg, May 2019. doi:10.1007/978-3-030-17659-4_22.

9 Allison Bishop, Lucas Kowalczyk, Tal Malkin, Valerio Pastro, Mariana Raykova, and Kevin Shi. A Simple Obfuscation Scheme for Pattern-Matching with Wildcards. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part III*, volume 10993 of *LNCS*, pages 731–752. Springer, Heidelberg, August 2018. doi:10.1007/978-3-319-96878-0_25.

10 Dan Boneh, Yuval Ishai, Alain Passelègue, Amit Sahai, and David J. Wu. Exploring Crypto Dark Matter: New Simple PRF Candidates and Their Applications. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018, Part II*, volume 11240 of *LNCS*, pages 699–729. Springer, Heidelberg, November 2018. doi:10.1007/978-3-030-03810-6_25.

11 Dan Boneh, Yuval Ishai, Amit Sahai, and David J. Wu. Lattice-Based SNARGs and Their Application to More Efficient Obfuscation. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part III*, volume 10212 of *LNCS*, pages 247–277. Springer, Heidelberg, April/May 2017. doi:10.1007/978-3-319-56617-7_9.

12 Zizhong Chen and Jack J. Dongarra. Condition Numbers of Gaussian Random Matrices. *SIAM J. Matrix Analysis Applications*, 27(3):603–620, 2005.

13 Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. Practical Multilinear Maps over the Integers. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 476–493. Springer, Heidelberg, August 2013. doi:10.1007/978-3-642-40041-4_26.

14 Nicolas Courtois and Josef Pieprzyk. Cryptanalysis of Block Ciphers with Overdefined Systems of Equations. In Yuliang Zheng, editor, *ASIACRYPT 2002*, volume 2501 of *LNCS*, pages 267–287. Springer, Heidelberg, December 2002. doi:10.1007/3-540-36178-2_17.

15 Geoffroy Couteau, Aurélien Dupin, Pierrick Méaux, Mélissa Rossi, and Yann Rotella. On the Concrete Security of Goldreich's Pseudorandom Generator. In Thomas Peyrin and Steven Galbraith, editors, *ASIACRYPT 2018, Part II*, volume 11273 of *LNCS*, pages 96–124. Springer, Heidelberg, December 2018. doi:10.1007/978-3-030-03329-3_4.

16 Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate Multilinear Maps from Ideal Lattices. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 1–17. Springer, Heidelberg, May 2013. doi:10.1007/978-3-642-38348-9_1.

17 Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate Indistinguishability Obfuscation and Functional Encryption for all Circuits. In *54th FOCS*, pages 40–49. IEEE Computer Society Press, October 2013. doi:10.1109/FOCS.2013.13.

18 Sanjam Garg, Craig Gentry, Shai Halevi, Amit Sahai, and Brent Waters. Attribute-Based Encryption for Circuits from Multilinear Maps. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 479–499. Springer, Heidelberg, August 2013. doi:10.1007/978-3-642-40084-1_27.

19 Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 467–476. ACM Press, June 2013. doi:10.1145/2488608.2488667.

20 Craig Gentry, Sergey Gorbunov, and Shai Halevi. Graph-Induced Multilinear Maps from Lattices. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 498–527. Springer, Heidelberg, March 2015. doi:10.1007/978-3-662-46497-7_20.

21 Craig Gentry, Charanjit S. Jutla, and Daniel Kane. Obfuscation Using Tensor Products. Cryptology ePrint Archive, Report 2018/756, 2018. URL: https://eprint.iacr.org/2018/756.

22 Oded Goldreich. Candidate One-Way Functions Based on Expander Graphs. Cryptology ePrint Archive, Report 2000/063, 2000. URL: http://eprint.iacr.org/2000/063.

23 Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. How to Run Turing Machines on Encrypted Data. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 536–553. Springer, Heidelberg, August 2013. doi:10.1007/978-3-642-40084-1_30.

24 Vipul Goyal, Yuval Ishai, Amit Sahai, Ramarathnam Venkatesan, and Akshay Wadia. Founding Cryptography on Tamper-Proof Hardware Tokens. In Daniele Micciancio, editor, *TCC 2010*, volume 5978 of *LNCS*, pages 308–326. Springer, Heidelberg, February 2010. doi:10.1007/978-3-642-11799-2_19.

25 Ilse C. F. Ipsen and Rizwana Rehman. Perturbation Bounds for Determinants and Characteristic Polynomials. *SIAM J. Matrix Analysis Applications*, 30(2):762–776, 2008.

26 Yuval Ishai and Eyal Kushilevitz. Private simultaneous messages protocols with applications. In *Proceedings of the Fifth Israeli Symposium on Theory of Computing and Systems*, pages 174–183. IEEE, 1997.

27 Yuval Ishai and Eyal Kushilevitz. Randomizing Polynomials: A New Representation with Applications to Round-Efficient Secure Computation. In *41st FOCS*, pages 294–304. IEEE Computer Society Press, November 2000. doi:10.1109/SFCS.2000.892118.

28 Yuval Ishai and Eyal Kushilevitz. Perfect Constant-Round Secure Computation via Perfect Randomizing Polynomials. In Peter Widmayer, Francisco Triguero Ruiz, Rafael Morales Bueno, Matthew Hennessy, Stephan Eidenbenz, and Ricardo Conejo, editors, *ICALP 2002*, volume 2380 of *LNCS*, pages 244–256. Springer, Heidelberg, July 2002. doi:10.1007/3-540-45465-9_22.

29 Aviad Kipnis and Adi Shamir. Cryptanalysis of the HFE Public Key Cryptosystem by Relinearization. In Michael J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 19–30. Springer, Heidelberg, August 1999. doi:10.1007/3-540-48405-1_2.

30 Adam Klivans and Amir Shpilka. Learning Arithmetic Circuits via Partial Derivatives. In *Computational Learning Theory and Kernel Machines, 16th Annual Conference on Computational Learning Theory and 7th Kernel Workshop, COLT/Kernel 2003, Washington, DC, USA, August 24-27, 2003, Proceedings*, volume 2777, pages 463–476, January 2003. doi:10.1007/978-3-540-45167-9_34.

31 Amit Sahai. New Roads to Cryptopia: A research agenda. New Roads to Cryptopia Workshop at CRYPTO 2019, 2019.

32 Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In David B. Shmoys, editor, *46th ACM STOC*, pages 475–484. ACM Press, May/June 2014. doi:10.1145/2591796.2591825.

33 Terence Tao and Van Vu. On Random &Plusmn;1 Matrices: Singularity and Determinant. *Random Struct. Algorithms*, 28(1):1–23, January 2006. doi:10.1002/rsa.v28:1.

34 Terrence Tao. Operator Norm of a Random Matrix. Wordpress Bog. URL: https://terrytao.wordpress.com/2010/01/09/254a-notes-3-the-operator-norm-of-a-random-matrix/#utail-wigner.

35 Enrico Thomae and Christopher Wolf. Solving Underdetermined Systems of Multivariate Quadratic Equations Revisited. In *Public Key Cryptography - PKC 2012 - 15th International Conference on Practice and Theory in Public Key Cryptography, Darmstadt, Germany, May 21-23, 2012. Proceedings*, pages 156–171, 2012.

36 Joel A. Tropp. An Introduction to Matrix Concentration Inequalities. *Foundations and Trends in Machine Learning*, 8(1-2):1–230, 2015.

37 Joe Zimmerman. How to Obfuscate Programs Directly. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 439–467. Springer, Heidelberg, April 2015. doi:10.1007/978-3-662-46803-6_15.

OV Graphs Are (Probably) Hard Instances

Josh Alman
Harvard University, Cambridge, MA, USA
jalman@seas.harvard.edu

Virginia Vassilevska Williams
MIT, Cambridge, MA, USA
virgi@mit.edu

Abstract

A graph G on n nodes is an Orthogonal Vectors (OV) graph of dimension d if there are vectors $v_1, \ldots, v_n \in \{0,1\}^d$ such that nodes i and j are adjacent in G if and only if $\langle v_i, v_j \rangle = 0$ over $\mathbb{Z}$. In this paper, we study a number of basic graph algorithm problems, except where one is given as input the vectors defining an OV graph instead of a general graph. We show that for each of the following problems, an algorithm solving it faster on such OV graphs G of dimension only $d = O(\log n)$ than in the general case would refute a plausible conjecture about the time required to solve sparse MAX-k-SAT instances:

- Determining whether G contains a triangle.
- More generally, determining whether G contains a directed k-cycle for any $k \geq 3$.
- Computing the square of the adjacency matrix of G over $\mathbb{Z}$ or $\mathbb{F}_2$.
- Maintaining the shortest distance between two fixed nodes of G, or whether G has a perfect matching, when G is a dynamically updating OV graph.

We also prove some complementary results about OV graphs. We show that any problem which is NP-hard on constant-degree graphs is also NP-hard on OV graphs of dimension $O(\log n)$, and we give two problems which can be solved faster on OV graphs than in general: Maximum Clique, and Online Matrix-Vector Multiplication.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis; Theory of computation → Dynamic graph algorithms

Keywords and phrases Orthogonal Vectors, Fine-Grained Reductions, Cycle Finding

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.83

Funding *Josh Alman*: Supported by a Michael O. Rabin Postdoctoral Fellowship.
Virginia Vassilevska Williams: Partially supported by an NSF Career Award, a Sloan Fellowship, NSF Grants CCF-1417238, CCF-1528078 and CCF-1514339, and BSF Grant BSF:2012338.

Acknowledgements The authors would like to thank the anonymous reviewers for their comments on an earlier version.

1 Introduction

Two of the most studied conjectures in fine-grained complexity are the Strong Exponential Time Hypothesis (SETH), and the Orthogonal Vectors Conjecture (OVC). SETH was introduced by Impagliazzo, Paturi and Zane [17] regarding the complexity of k-SAT:

▶ **Hypothesis 1** (Strong Exponential Time Hypothesis). *For every $\varepsilon > 0$, there is an integer $k \geq 3$ such that k-SAT on n variables cannot be solved in $O(2^{(1-\varepsilon)n})$ (randomized) time.*

OVC concerns the Orthogonal Vectors (OV) problem: Given as input a set $A \subseteq \{0,1\}^d$ of $|A| = n$ vectors, determine whether there are $a, b \in A$ such that $\langle a, b \rangle = 0$ (all inner products in this paper, including this one, are taken over $\mathbb{Z}$ unless stated otherwise).

▶ **Hypothesis 2** (Orthogonal Vectors Conjecture). *For every $\varepsilon > 0$, there is a $c > 0$ such that OV in dimension $d = c \log n$ cannot be solved in $O(n^{2-\varepsilon})$ (randomized) time.*

Williams [28] showed that SETH implies OVC. Most of the known fine-grained implications of SETH use this result and are actually proved assuming OVC instead; see [27] for a survey of the many applications of OVC to graph algorithms, string algorithms, nearest neighbors problems, and more.

In this paper, we study a mathematical object inspired by OVC which we call an OV graph: a graph G on n nodes is an OV graph of dimension d if there are n vectors $v_1, \ldots, v_n \in \{0,1\}^d$ such that nodes i and j are adjacent in G if and only if $\langle v_i, v_j \rangle = 0$. Given as input the vectors $v_1, \ldots, v_n \in \{0,1\}^d$ defining G, there are a number of natural algorithmic questions one might ask, including:

- OV: does G have any edges?
- OVΔ: does G contain any triangles?
- OV-DIR-k-CYCLE: given a partition of the nodes of G into k parts, is there a k-cycle containing one node from each part?

Detecting triangles and more generally, k-cycles are among the most basic algorithmic questions one can ask about graphs. OV graphs of low dimension $d \ll n$ make up a small fraction of all graphs: there are only $2^{O(nd)}$ such graphs on n nodes, compared to $2^{\Theta(n^2)}$ total graphs on n nodes. However, in this paper we will show that solving these problems on OV graphs of dimension only $d = O(\log n)$ may be just as hard as solving them in general graphs. Somewhat analogously to how Williams showed that faster algorithms for OV in dimension $O(\log n)$ would lead to breakthroughs in solving k-SAT, we will show that faster algorithms for OVΔ or OV-DIR-k-CYCLE on OV graphs of dimension $O(\log n)$ would lead to breakthroughs in solving MAX-k-SAT.

MAX-k-SAT

In the MAX-k-SAT problem for an integer $k \geq 2$, given as input a k-CNF formula ϕ, the goal is to determine the maximum number of clauses of ϕ which can be satisfied by a single assignment.

MAX-k-SAT on n variables and m clauses can be solved in $O(2^n m)$ time by exhaustive search. Williams [28, 29] showed that MAX-2-SAT has a much faster, $2^{\omega n/3}\text{poly}(n)$ time algorithm, where $\omega < 2.373$ is the exponent of matrix multiplication [26, 13]. This running time for MAX-2-SAT has remained unchallenged for over 15 years. It is an interesting open problem whether a faster algorithm exists.

Williams' techniques for MAX-2-SAT do not carry over to MAX-k-SAT for $k \geq 3$ (see [19] for a discussion), and there is no known $O((2-\varepsilon)^n)$ time algorithm for MAX-k-SAT for any $k \geq 3$ and $\varepsilon > 0$.

Unlike with k-SAT [16], there is no known sparsification lemma for MAX-k-SAT, so that in principle MAX-k-SAT on formulas with $O(n)$ clauses might be easier than the general case of MAX-k-SAT that might have n^k clauses. This has led researchers to investigate the complexity of such sparse instances of MAX-k-SAT (e.g. [11, 9, 3]).

The fastest known algorithms for MAX-k-SAT on n variables and cn clauses for constant c run in time $2^{n(1-1/O(\log^2 c))}\text{poly}(n)$ when $k \leq 4$, or in time $2^{n(1-1/O(c^{1/3}))}\text{poly}(n)$ when $k > 4$ [3]. Unfortunately, as c grows, these running times go to 2^n, the brute force running time. Thus the following hypothesis is fully consistent with the state-of-the art of MAX-k-SAT algorithms:

▶ **Hypothesis 3** (Sparse MAX-3-SAT Hypothesis). *For every $\varepsilon > 0$, there exists a $c > 0$ so that n variable MAX-3-SAT on cn clauses cannot be solved in time $O(2^{n(1-\varepsilon)})$.*

The hypothesis above strengthens an earlier hypothesis that MAX-3-SAT requires $2^{n(1-o(1))}$ time (see e.g. [27, 19]); it would be equivalent to that hypothesis if a sparsification lemma for MAX-3-SAT can be proven. We will base some of our hardness results on our strengthened hypothesis. We will also use an analogous hypothesis for Sparse MAX-2-SAT:

▶ **Hypothesis 4** (Sparse MAX-2-SAT Hypothesis). *For every $\varepsilon > 0$, there exists a $c > 0$ so that n variable MAX-2-SAT on cn clauses cannot be solved in time $O(2^{n(\omega/3-\varepsilon)})$.*

1.1 Our Results

Triangle Finding and Matrix Multiplication

The best known algorithm for finding a triangle in a graph on n nodes runs in time $n^{\omega+o(1)}$, where $\omega \leq 2.373$ is the matrix multiplication exponent [26, 13]. Our first result is that if there is a faster algorithm that finds triangles in *OV graphs*, then Hypothesis 4 would be violated and sparse MAX-2-SAT would have faster algorithms.

▶ **Theorem 5.** *Suppose OVΔ in OV graphs with n nodes and dimension $O(\log n)$ can be solved in time $n^{\omega-\varepsilon+o(1)}$ for some constant $\varepsilon > 0$. Then, for any constants $a, \delta > 0$, MAX-2-SAT on n variables and $a \cdot n$ clauses can be solved in time $2^{(\omega/3-\varepsilon/3+\delta)n}$.*

The best known algorithm for triangle finding in a general graph G works by reducing to the *Boolean matrix multiplication* of two copies of the adjacency matrix of G. In Boolean matrix multiplication, given as input two matrices $A, B \in \{0,1\}^{n\times n}$, the goal is to compute their product over the (AND,OR) semiring, i.e. the matrix $C \in \{0,1\}^{n\times n}$ given by $C[i,j] = \bigvee_{k=1}^{n} A[i,k] \wedge B[k,j]$. This can be solved in time $n^{\omega+o(1)}$ using a simple reduction to matrix multiplication over either $\mathbb{F}_2$ or $\mathbb{Z}$.

Similarly, OVΔ in OV graphs with n nodes and dimension d has a simple linear-time reduction to Boolean matrix multiplication of $n \times n$ matrices which are the adjacency matrices of *OV* graphs of dimension d. In other words, these are matrices $A \in \{0,1\}^{n\times n}$ for which there are vectors $v_1, \ldots, v_n \in \{0,1\}^d$ such that $A[i,j] = 1$ if and only if $\langle v_i, v_j \rangle = 0$. If such matrices for $d = O(\log n)$ can be multiplied in $n^{\omega-\varepsilon+o(1)}$ time for some constant $\varepsilon > 0$, it would lead to a corresponding speedup for MAX-2-SAT on n variables and $O(n)$ clauses as well. In other words, this small set of only $O(2^{nd}) = O(2^{n \log n})$ matrices with such efficient descriptions may be just as difficult to multiply as arbitrary $\{0,1\}$ matrices (of which there are $2^{\Theta(n^2)}$).

Consequences of the n^ω hardness of OVΔ for dynamic algorithms

Because of its simplicity, triangle detection has been reduced to many other problems. For instance, Abboud and Vassilevska Williams [1] present several clean reductions from triangle detection to a variety of dynamic problems. We next show that, as a consequence of the hardness of OVΔ, many of the triangle-based lower bounds in [1] also hold for OV graphs with $O(\log n)$ dimension.

Typically, in dynamic graph algorithms one supports edge insertions and deletions. In OV graphs, however, the edge relation is captured by the labels on the vertices of the graph. Thus, in dynamic OV graphs, we instead support vertex label updates as above. One could also consider an alternate model where the updates may only change one bit of a vertex label, but update times in these two models differ by at most a fairly negligible $O(d)$ factor.

Although changing a vertex label can change all the incident edges to the vertex, our lower bounds apply even when each update changes at most a *single* edge. Hence the lower bounds would also apply in the standard dynamic graph algorithms model if one maintains the OV graph as a graph, rather than as a set of vectors.

Assuming that OVΔ requires $n^{\omega-o(1)}$ time (as follows from Hypothesis 4), we obtain the following conditional lower bounds:

- Dynamic s-t OV Shortest Paths, maintaining the distance between two fixed vertices s and t in an n node OV graph with dimension $O(\log n)$ under vertex relabel updates (change the vector representing a node) requires either $n^{\omega-o(1)}$ preprocessing time, or $n^{\omega-1-o(1)}$ amortized update time; the lower bound holds even if the updates insert or delete at most a single edge. The same lower bounds hold for $5/3-\varepsilon$-approximating the s-t distance when arbitrary label updates are allowed, even when the preprocessing time can be arbitrary.
- Dynamic bipartite perfect matching in OV graphs on n nodes and dimension $O(\log n)$ under vertex relabel updates requires either $n^{\omega-o(1)}$ preprocessing time, or $n^{\omega-1-o(1)}$ amortized update time. The lower bound holds even if the updates insert or delete at most a single edge. If the updates can be arbitrary, the lower bound holds for arbitrary preprocessing time.

In the full version of the paper we present more such reductions. Because of the structure of our reductions, the lower bounds we prove in which the relabelings change only a single edge also hold for incremental algorithms (where edges are only inserted), but the lower bound is only for worst case update time.

Directed k-Cycle

The fastest known algorithm for finding a k-cycle for any constant $k \geq 3$ in a general directed n-node graph runs in $n^{\omega+o(1)}$ time via color-coding and matrix multiplication [6]. Our next result is that under Hypothesis 4, this running time is essentially tight even in OV graphs:

▶ **Theorem 6.** *Let $k \geq 3$ be any constant. Suppose that k-Cycle in directed OV graphs with n nodes and dimension $O(\log n)$ can be solved in time $n^{\omega-\varepsilon+o(1)}$ for some constant $\varepsilon > 0$. Then, for any constants $a, \delta > 0$, MAX-2-SAT on n variables and $a \cdot n$ clauses can be solved in time $2^{(\omega/3-\varepsilon/3+\delta)n}$.*

As our MAX-3-SAT Hypothesis 3 is potentially more believable than our MAX-2-SAT Hypothesis 4, we further investigate what it implies for k-Cycle detection. We show that unless Hypothesis 3 fails, there is no constant k for which k-cycle in an n node OV graph with dimension $O(\log n)$ can be found in $O(n^{3/2-\varepsilon})$ time for any $\varepsilon > 0$. The statement of our directed k-cycle results under Hypothesis 3 are strongest for $k = 4$:

▶ **Theorem 7.** *Suppose that OV-DIR-4CYCLE in OV graphs with n nodes and dimension $O(\log n)$ can be solved in time $n^{2-\varepsilon+o(1)}$ for some constant $\varepsilon > 0$. Then, for any constants $a, \delta > 0$, MAX-3-SAT on n variables and $a \cdot n$ clauses can be solved in time $2^{(1-\varepsilon/2+\delta)n}$, and Hypothesis 3 fails.*

If $\omega = 2$, the above would give an essentially tight lower bound under a better hypothesis. While the lower bounds we obtain under Hypothesis 3 are not nearly as strong as the tight results we get under Hypothesis 4, they are conditioned on a slightly more believable hypothesis. Moreover, the techniques seem to be slightly more flexible, so that there might be hope that a similar result might hold for k-cycle in *undirected* graphs. When k is odd,

our hardness results already hold for k-cycle in undirected OV graphs, but when k is even, k-cycle in general undirected graphs can be solved in $O(n^2)$ time regardless of k [30]. Hence our $n^{\omega-o(1)}$ lower bound from Hypothesis 4 may not extend to undirected graphs if $\omega > 2$. A strengthening of our MAX-3-SAT-based reductions to undirected graphs might still be possible, and would be significant as it would be tight, at least for 4-cycles.

Constant Degree Graphs have low OV dimension

Next, we study the complexity of various NP-hard graph problems when they are restricted to graphs on n nodes with OV dimension $O(\log n)$. We begin with problems which are known to be NP-hard on constant-degree graphs, including Hamiltonian Path and Minimum Vertex Cover. It follows from prior work [5] that constant-degree graphs have OV dimension $O(\log n)$. However, the proof of this is nonconstructive, and uses the probabilistic method. We nonetheless show how to derandomize this proof, giving a deterministic polynomial-time algorithm for finding a representation of a constant-degree graph as an OV graph of dimension $O(\log n)$. We hence show:

▶ **Theorem 8.** *Any problem which is NP-hard on graphs of constant maximum degree is also NP-hard on OV graphs of dimension $O(\log n)$.*

We also show that at least one NP-hard problem, the Max Clique problem, seems to become easier on OV graphs of dimension $d \ll n$, by reducing to a set packing problem:

▶ **Theorem 9.** *Given as input vectors $V = \{v_1, \ldots, v_n\} \subseteq \{0,1\}^d$ defining a dimension d OV graph G_V, a maximum size clique in G_V can be found in time $2^d \cdot n^{O(1)}$.*

Online Matrix-Vector Multiplication

Finally, we study the Online Matrix-Vector Multiplication (OMV) problem: preprocess a matrix $M \in \mathbb{F}_2^{n \times n}$ so that, given as input a query vector $v \in \mathbb{F}_2^n$, one can quickly return the product $M \cdot v$. The best known algorithms in general answer queries in time $n^{2-o(1)}$ [18]. We show that faster algorithms are possible when the matrix M is the adjacency matrix of a graph with OV dimension $c \log n$:

▶ **Theorem 10.** *For $c > 0$, we can preprocess vectors $u_1, \ldots, u_n, v_1, \ldots, v_n \in \{0,1\}^{c \log n}$, which define a matrix $M \in \mathbb{F}_2^{n \times n}$ as $M[i,j] = 1$ if and only if $\langle u_i, v_j \rangle = 0$, in preprocessing time $\tilde{O}(n^2)$ with high probability, such that given as input a vector $v \in \mathbb{F}_2^n$, we can compute the product $M \cdot v$ in time $n^{2-1/O(\log c)}$.*

Data structures for OMV have many applications. For instance, by multiplying the adjacency matrix of a graph G by an indicator vector for a subset $S \subseteq \{1, 2, \ldots, n\}$ of the nodes of G, one gets the neighborhood of S. Similar to [18, Corollary 1.1], our Theorem 10 thus yields a data structure with $\tilde{O}(n^2)$ preprocessing time for OV graphs G of dimension $c \cdot \log n$ which, given as a query a subset S of the nodes of G, can answer whether S is independent, dominating, or a vertex cover, in time $n^{2-1/O(\log c)}$.

Our data structure for Theorem 10 works by expressing the matrix M as the sum of a matrix of rank $n^{0.1}$ and a sparse matrix with $n^{2-1/O(\log c)}$ nonzero entries. It does this by combining the algorithm for OV by Abboud et al. [2], which makes use of the "polynomial method in algorithm design", together with known techniques for converting polynomial method constructions into "matrix rigidity" upper bounds [4].

1.2 Other Related Work

OV graphs have been studied in a number of other settings in mathematics and computer science.

The Edge Clique Cover Number (ECCN) of a graph G is the minimum number of cliques needed to cover the edges of G. We can see that a graph G is an OV graph of dimension d if and only if the ECCN of the *complement* of G is d. A line of work in the combinatorics community has shown that a number of simple classes of graphs have low ECCN; see e.g. [12, 5, 15, 24, 23].

The complexity of determining the ECCN of an input graph has also been studied. The problem is known to be NP-hard in general, but Gramm et al. [14] gave a parameterized algorithm running in time $2^{2^{O(k)}} \cdot \text{poly}(n)$ on an n node graph with ECCN k. Such a doubly-exponential dependence on k may be optimal: Cygan et al. [10] showed that a $2^{2^{o(k)}} \cdot \text{poly}(n)$ time algorithm would refute the Exponential Time Hypothesis (a weak version of SETH).

Representations of graphs by the orthogonality relations of vectors have also been studied in other areas. For instance, Lovász [20] describes "orthonormal representations" where each node must be assigned a unit vector in Euclidean space, and the vectors corresponding to nonadjacent vertices must be orthogonal. In this language, if G is a graph with low OV dimension, then the complement of G has low $\{0,1\}$-faithful orthogonality dimension; see e.g. [22, 21].

2 Preliminaries

2.1 Notation

For a positive integer d, write 0^d for the all-zeroes vector of dimension d. Write $||$ to denote concatenation of vectors.

Define $AND : \{0,1\}^d \times \{0,1\}^d \to \{0,1\}^d$, the bit-wise AND function by, for $x, y \in \{0,1\}^d$ and $\ell \in \{1, \ldots, d\}$, $AND(x,y)[\ell] = x[\ell] \cdot y[\ell]$.

2.2 OV Graphs

▶ **Definition 11** (OV Graph). *For positive integers n, d and vectors $V = \{v_1, \ldots, v_n\} \subseteq \{0,1\}^d$, the OV graph G_V is the graph on n nodes where nodes i and j are adjacent if and only if $\langle v_i, v_j \rangle = 0$, where the inner product is over $\mathbb{Z}$. The OV dimension of a graph G is the smallest nonnegative integer d such that G can be written as the OV graph of vectors in $\{0,1\}^d$.*

▶ **Definition 12** (Generalized Inner Product). *For any positive integers n, d and vectors $v_1, \ldots, v_n \in \mathbb{Z}^d$, we define the generalized inner product*

$$IP(\{v_1, \ldots, v_n\}) = \langle v_1, \ldots, v_n \rangle = \sum_{\ell=1}^{d} v_1[\ell] \cdot v_2[\ell] \cdots v_n[\ell].$$

The sum (as well as all inner products in this paper) is taken over $\mathbb{Z}$.

▶ **Definition 13** (k-Uniform OV Hypergraph). *For positive integers n, d and vectors $V = \{v_1, \ldots, v_n\} \subseteq \{0,1\}^d$, the k-uniform OV hypergraph $G_{V,k}$ is the k-uniform hypergraph on n nodes where, for distinct nodes $i_1, i_2, \ldots, i_k$, the hyperedge $(i_1, \ldots, i_k)$ is in $G_{V,k}$ if and only if $\langle v_{i_1}, v_{i_2}, \ldots, v_{i_k} \rangle = 0$.*

▶ **Definition 14** (OV dimension of a matrix). *The OV dimension of a matrix $M \in \{0,1\}^{n \times n}$ is the smallest nonnegative integer d such that there are $2n$ vectors $u_1, \ldots, u_n, v_1, \ldots, v_n \in \{0,1\}^d$ with the property that, for any $i, j \in \{1, 2, \ldots, n\}$, we have $M[i,j] = 1$ if and only if $\langle u_i, v_j \rangle = 0$. Equivalently, the OV dimension of M is the rank of $\bar{M}$ (the all 1s matrix minus M) over the OR, AND (Boolean) semiring.*

▶ Remark 15. The adjacency matrix of an OV graph of dimension d is a matrix of OV dimension d.

2.3 Algorithmic Problems

▶ **Definition 16.** *We define some problems. For positive integers n, d:*

- *OV$_{n,d}$: Given $A, B \subseteq \{0,1\}^d$ with $|A| = |B| = n$, determine whether there is an $(a,b) \in A \times B$ with $\langle a, b \rangle = 0$.*
- *EXACT-IP$_{n,d}$: Given $A, B \subseteq \{0,1\}^d$ with $|A| = |B| = n$, and an integer $0 \leq m \leq d$, determine whether there is an $(a,b) \in A \times B$ with $\langle a, b \rangle = m$.*
- *OV$\Delta_{n,d}$: Given $V \subseteq \{0,1\}^d$ with $|V| = n$, determine whether there are distinct $a, b, c \in V$ with $\langle a, b \rangle = \langle b, c \rangle = \langle c, a \rangle = 0$.*
- *EXACT-IP$\Delta_{n,d}$: Given $A, B, C \subseteq \{0,1\}^d$ with $|A| = |B| = |C| = n$, and three integers $0 \leq m_{AB}, m_{BC}, m_{CA} \leq d$, determine whether there is a $(a,b,c) \in A \times B \times C$ with $\langle a, b \rangle = m_{AB}$, $\langle b, c \rangle = m_{BC}$, and $\langle c, a \rangle = m_{CA}$.*
- *OV-HYPERGRAPH$_{n,d,\ell,k}$ (for $\ell > k \geq 2$): Given $V \subseteq \{0,1\}^d$ with $|V| = n$, determine whether there is a $T \subseteq V$ of size $|T| = \ell$ such that for all $S \subseteq T$ of size $|S| = k$ we have IP$(S) = 0$.*
- *DIRECTED-CYCLE$_{n,d,k}$: Given $V_1, \ldots, V_k \subseteq \{0,1\}^d$ with $|V_1| = \cdots = |V_k| = n$, determine whether there is a $v_i \in V_i$ for each $i \in \{1, 2, \ldots, k\}$ such that $\langle v_i, v_{i+1} \rangle = 0$ for all such i (with $v_{k+1} = v_1$).*
- *MAX-k-SAT$_{n,m}$: Given a k-CNF formula ϕ on n variables and m clauses, determine the maximum number of clauses of ϕ which can be satisfied by an assignment.*

3 Reduction from MAX-2SAT to Finding Triangles in OV Graphs

▶ **Lemma 17.** *There is a polynomial-time reduction from MAX-2-SAT$_{n,m}$ to $O(m^3)$ instances of EXACT-IP$\Delta_{3 \cdot 2^{\lceil n/3 \rceil}, m}$.*

Proof. Partition the input variables into three groups X, Y, Z of $n/3$ variables each. Let S be the set of clauses of ϕ. Partition S into six sets $S_X, S_Y, S_Z, S'_X, S'_Y, S'_Z$ such that:

- S'_X contains the clauses which consist only of literals of variables from X, and S'_Y and S'_Z are defined similarly.
- $S_X \subseteq S \setminus (S'_X \cup S'_Y \cup S'_Z)$ contains the clauses which contain one literal from Y and one from Z, and S_Y and S_Z are defined similarly.

For each of the $O(m^3)$ choices of $0 \leq m_X, m_Y, m_Z \leq m$, we will determine whether it is possible to find an assignment to ϕ which satisfies all but m_X clauses of $S'_Y \cup S_Z$, all but m_Y clauses of $S'_Z \cup S_X$, and all but m_Z clauses of $S'_X \cup S_Y$. From this we can compute the desired answer.

For any assignment $\alpha : X \to \{0,1\}^{n/3}$, define the vector $v_\alpha \in \{0,1\}^m$, whose entries are indexed by clauses c in S, by

$$v_\alpha[c] = \begin{cases} 0 & \text{if } c \in S'_X \cup S_Y \cup S_Z \text{ and } \alpha \text{ satisfies a literal in } c, \\ 0 & \text{if } c \in S'_Z \cup S_X, \\ 1 & \text{otherwise.} \end{cases}$$

Similarly, for any assignment $\beta : Y \to \{0,1\}$, define $v_\beta \in \{0,1\}^m$ by

$$v_\beta[c] = \begin{cases} 0 & \text{if } c \in S_X \cup S'_Y \cup S_Z \text{ and } \alpha \text{ satisfies a literal in } c, \\ 0 & \text{if } c \in S'_X \cup S_Y, \\ 1 & \text{otherwise.} \end{cases}$$

and for any assignment $\gamma : Z \to \{0,1\}$, define $v_\gamma \in \{0,1\}^m$ by

$$v_\gamma[c] = \begin{cases} 0 & \text{if } c \in S_X \cup S_Y \cup S'_Z \text{ and } \alpha \text{ satisfies a literal in } c, \\ 0 & \text{if } c \in S'_Y \cup S_Z, \\ 1 & \text{otherwise.} \end{cases}$$

Note that for any assignment s to all the variables, letting $\alpha = s|_X$, $\beta = s|_Y$, and $\gamma = s|_Z$, we have that $\langle v_\alpha, v_\beta \rangle$ counts the number of clauses in $S'_Y \cup S_Z$ which are unsatisfied by s, $\langle v_\beta, v_\gamma \rangle$ counts the number of clauses in $S'_Z \cup S_X$ which are unsatisfied by s, and $\langle v_\gamma, v_\alpha \rangle$ counts the number of clauses in $S'_X \cup S_Y$ which are unsatisfied by s. In other words, our goal is to determine whether there is an $\alpha : X \to \{0,1\}$, $\beta : Y \to \{0,1\}$, and $\gamma : Z \to \{0,1\}$ with $\langle v_\alpha, v_\beta \rangle = m_X$, $\langle v_\beta, v_\gamma \rangle = m_Y$, and $\langle v_\gamma, v_\alpha \rangle = m_Z$. This is exactly an instance of EXACT-IP$\Delta_{3 \cdot 2^{\lceil n/3 \rceil}, m}$, as desired. ◀

▶ **Theorem 18** (Reduction from EXACT-IP to OV implicit in [8, Proof of Lemma 4.2])**.** *For any positive integer n, and any $c, \varepsilon > 0$, set $s := n^{\varepsilon \log(c/\varepsilon)}$ and $d := O(2^{c/\varepsilon} \log n)$. There is a pair of maps*

$$r_1, r_2 : \{0,1\}^{c \log n} \times \{1, 2, \ldots, s\} \times \{0, 1, \ldots, c \log n\} \to \{0,1\}^d$$

which can be computed in deterministic time $O(s \cdot 2^{c/\varepsilon} \log n)$ such that for any $x, y \in \{0,1\}^{c \log n}$, and any $m \in \{0, 1, \ldots, c \log n\}$, we have $\langle x, y \rangle = m$ if and only if there is an $i \in \{1, 2, \ldots, s\}$ such that $\langle r_1(x, i, m), r_2(y, i, m) \rangle = 0$ (over $\mathbb{Z}$).

▶ **Lemma 19.** *For every $c, \varepsilon > 0$, there is a reduction from an EXACT-IP$\Delta_{n, c \log n}$ instance to $n^{3\varepsilon \log(c/\varepsilon)}$ many instances of OV$\Delta_{n, O(2^{c/\varepsilon} \log n)}$. The EXACT-IP$\Delta$ instance is a yes instance if and only if at least one of the OVΔ instances is a yes instance. The reduction takes time $2^{O(\varepsilon \log(c/\varepsilon) \log n + c/\varepsilon)}$.*

Proof. In our EXACT-IP$\Delta_{n, c \log n}$ instance, we are given $A, B, C \subseteq \{0,1\}^{c \log n}$ with $|A| = |B| = |C| = n$ and three nonnegative integers m_{AB}, m_{BC}, m_{CA}, and our goal is to determine whether there is a $(a, b, c) \in A \times B \times C$ such that $\langle a, b \rangle = m_{AB}$, $\langle b, c \rangle = m_{BC}$, and $\langle c, a \rangle = m_{CA}$.

Set $s := n^{\varepsilon \log(c/\varepsilon)}$ and $d := O(2^{c/\varepsilon} \log n)$, and let r_1, r_2 be the maps from Theorem 18. Then, our goal is equivalently to determine whether there is a choice of $(a, b, c) \in A \times B \times C$ and $i_{AB}, i_{BC}, i_{CA} \in \{1, 2, \ldots, s\}$ such that $\langle r_1(a, i_{AB}, m_{AB}), r_2(b, i_{AB}, m_{AB}) \rangle = \langle r_1(b, i_{BC}, m_{BC}), r_2(c, i_{BC}, m_{BC}) \rangle = \langle r_1(c, i_{CA}, m_{CA}), r_2(a, i_{CA}, m_{CA}) \rangle = 0$.

For each fixed choice of $i_{AB}, i_{BC}, i_{CA} \in \{1, 2, \ldots, s\}$, this is very nearly an instance of $\mathsf{OV}\Delta_{n,d}$, which would complete the proof. To convert it into an instance of $\mathsf{OV}\Delta_{n,3d}$, simply map:

$$a \to r_1(a, i_{AB}, m_{AB})||0^d||r_2(a, i_{CA}, m_{CA}),$$

$$b \to r_2(b, i_{AB}, m_{AB})||r_1(b, i_{BC}, m_{BC})||0^d,$$

$$c \to 0^d||r_2(c, i_{BC}, m_{BC})||r_1(c, i_{CA}, m_{CA}),$$

for each $a \in A$, $b \in B$, $c \in C$, where $||$ denotes concatenation of vectors, and 0^d is the all-0s vector of length d. ◀

▶ **Theorem 20.** *Suppose that $\mathit{OV}\Delta_{N,c\log N}$ can be solved in (deterministic) time $N^{\tau+o(1)}$ for some constant $\tau \geq 0$ and all constants $c > 0$. Then, for any constant $a > 0$, MAX-2-$\mathit{SAT}_{n,an}$ can be solved in (deterministic) time $O(2^{(\tau/3+\delta)n})$ for every $\delta > 0$.*

Proof. Let $\varepsilon > 0$ be a small constant to be set later. Lemma 17 gives a reduction from $\mathsf{MAX\text{-}2\text{-}SAT}_{n,an}$ to $O(n^3)$ instances of $\mathsf{EXACT\text{-}IP}\Delta_{O(2^{n/3}),an}$. Lemma 19 then reduces each of those to $2^{O(n\varepsilon\log(a/\varepsilon))}$ instances of $\mathsf{OV}\Delta_{O(2^{n/3}),O(n\cdot 2^{3a/\varepsilon})}$. The total time for computing all these reductions is $2^{O(n\varepsilon\log(a/\varepsilon)+a/\varepsilon)}$.

For any $\delta > 0$, we can pick a sufficiently small $\varepsilon > 0$ so that the combination is a reduction from $\mathsf{MAX\text{-}2\text{-}SAT}_{n,an}$ to $2^{\delta n/2}$ instances of $\mathsf{OV}\Delta_{O(2^{n/3}),O(n)}$, which can be computed in $2^{\delta/2}$ time. Each of those instances can be solved in time $2^{\tau n/3+o(n)}$ using the given algorithm with $N = O(2^{n/3})$, as desired. ◀

Finally, we can extend Theorem 20 from showing hardness just for $\mathsf{OV}\Delta_{n,O(\log n)}$ to showing hardness for $\mathsf{DIRECTED\text{-}CYCLE}_{n,O(\log n),k}$ for any integer $k \geq 3$ (the two problems coincide when $k = 3$) via a simple reduction:

▶ **Lemma 21.** *For any integer $k \geq 3$, there is a linear-time reduction from $\mathit{OV}\Delta_{n,O(\log n)}$ to $\mathit{DIRECTED\text{-}CYCLE}_{n,O(\log n),k}$.*

Proof. $\mathsf{OV}\Delta_{n,O(\log n)}$ reduces to $\mathsf{DIRECTED\text{-}CYCLE}_{n,O(\log n),3}$ by simply repeating the given input set three times (unless it contains the all-zeroes vector, in which case we include it in only one of the three sets). We next show how to reduce from $\mathsf{DIRECTED\text{-}CYCLE}_{n,O(\log n),k-1}$ to $\mathsf{DIRECTED\text{-}CYCLE}_{n,O(\log n),k}$ for any $k \geq 4$, which will complete the proof.

We are given as input $k-1$ sets $V_1, \ldots, V_{k-1} \subseteq \{0,1\}^d$ for $d = O(\log n)$ and $|V_i| = n$ for all $i \in \{1, 2, \ldots, k-1\}$. For each i write $V_i = \{v_{i,j}\}_{j\in\{1,\ldots,n\}}$.

Next, for a sufficiently large $d' = O(\log n)$, pick $2n$ vectors $\alpha_1, \ldots, \alpha_n, \beta_1, \ldots, \beta_n \in \{0,1\}^{d'}$ such that, for $i, j \in \{1, 2, \ldots, n\}$ we have

$$\langle \alpha_i, \beta_j \rangle = \begin{cases} 0 & \text{if } i = j \\ \geq 1 & \text{otherwise.} \end{cases}$$

This can be done, for instance, by picking each α_i to be a different vector with exactly half its entries 1, and picking β_i to be $\bar{\alpha_i}$ (i.e. the all 1s vector minus α_i). We thus can pick the smallest $d' = O(\log n)$ such that $\binom{d'}{d'/2} \geq n$.

We will now pick sets $V'_1, \ldots, V'_k \subseteq \{0,1\}^{2d+d'}$ for our $\mathsf{DIRECTED\text{-}CYCLE}_{n,O(\log n),k}$ instance. We will pick $V'_i = \{v'_{i,j}\}_{j\in\{1,2,\ldots,n\}}$, where the $v'_{i,j}$ vectors are defined as follows: If $i \in \{1, 2, \ldots, k-2\}$ then set $v'_{i,j} = v_{i,j}||v_{i,j}||0^{d'}$. Then, set $v'_{k-1,j} = 0^d||v_{k-1,j}||\alpha_j$ and $v'_{k,j} = v_{k-1,j}||0^d||\beta_j$.

We can see with these choices that the vectors $(v_{1,j_1}, v_{2,j_2}, \ldots, v_{k-1,j_{k-1}})$ formed a $k-1$ cycle in the original instance of DIRECTED-CYCLE$_{n,O(\log n),k-1}$ if and only if the k vectors $(v'_{1,j_1}, v'_{2,j_2}, \ldots, v'_{k-1,j_{k-1}}, v'_{k,j_{k-1}})$ form a k cycle in the new instance we created of DIRECTED-CYCLE$_{n,O(\log n),k}$. Moreover, $\langle v'_{k-1,j}, v'_{k,j'}\rangle = 0$ if and only if $j = j'$, so these are the only possible k cycles in the new instance, as desired. ◀

▶ **Corollary 22.** *Suppose for any $k \geq 3$ that DIRECTED-CYCLE$_{n,O(\log n),k}$ can be solved in (deterministic) time $N^{\tau+o(1)}$ for some constant $\tau \geq 0$. Then, for any constant $a > 0$, MAX-2-SAT$_{n,an}$ can be solved in (deterministic) time $O(2^{(\tau/3+\delta)n})$ for every $\delta > 0$.*

4 Hypercliques in OV Hypergraphs and Directed Cycles

A straightforward generalization of the above argument shows:

▶ **Theorem 23.** *Suppose, for integers $\ell > k \geq 2$, that OV-HYPERGRAPH$_{N,O(\log N),\ell,k}$ can be solved in (deterministic) time $N^{\tau+o(1)}$ for some constant $\tau \geq 0$. Then, for any constant $a > 0$, MAX-k-SAT$_{n,an}$ can be solved in (deterministic) time $O(2^{(\tau/\ell+\delta)n})$ for every $\delta > 0$.*

Following the reduction from finding hypercliques in hypergraphs to finding directed cycles of [19], we can also reduce to finding directed cycles in OV graphs, with some care to ensure that the OV dimension does not increase too much:

▶ **Theorem 24.** *Let $k \geq 4$ be a constant integer. There is a $\tilde{O}(d \cdot n^{k-\lceil k/3\rceil})$ time deterministic reduction from OV-HYPERGRAPH$_{n,d,k,3}$ to DIRECTED-CYCLE$_{n^{k-\lceil k/3\rceil},d+O(\log n),k}$.*

The proof of the theorem is a bit technical, so we first present the special case for $k = 4$ and then highlight some changes.

▶ **Theorem 25** ($k = 4$ case of Theorem 24)**.** *There is a $O(d \cdot n^2 \log n)$ time deterministic reduction from OV-HYPERGRAPH$_{n,d,4,3}$ to DIRECTED-CYCLE$_{\binom{n}{2},d+O(\log n),4}$.*

Proof. In OV-HYPERGRAPH$_{n,d,4,3}$, we are given as input a size-n set $V = \{v_1, \ldots, v_n\} \subseteq \{0,1\}^d$, and we want to determine whether there is a $T \subseteq V$ of size $|T| = 4$ such that, for all $S \subseteq T$ of size $|S| = 3$, we have $\mathsf{IP}(S) = 0$.

We first give some definitions. Define $AND : \{0,1\}^d \times \{0,1\}^d \to \{0,1\}^d$ by, for $x, y \in \{0,1\}^d$ and $\ell \in \{1, \ldots, d\}$, $AND(x,y)[\ell] = x[\ell] \cdot y[\ell]$. Next, similar to Lemma 21, for a sufficiently large $d' = O(\log n)$, pick $2n$ vectors $\alpha_1, \ldots, \alpha_n, \beta_1, \ldots, \beta_n \in \{0,1\}^{d'}$ such that, for $i, j \in \{1, 2, \ldots, n\}$ we have

$$\langle \alpha_i, \beta_j \rangle = \begin{cases} 0 & \text{if } i = j \\ \geq 1 & \text{otherwise.} \end{cases}$$

As before, we can pick the smallest $d' = O(\log n)$ such that $\binom{d'}{d'/2} \geq n$. Finally, define $\chi : \{1,2,3,4\}^2 \times \{1,2,\ldots,n\}^2 \to \{0,1\}^{d'}$ by

$$\chi(p,q,i,j) = \begin{cases} \alpha_j & \text{if } p = q \\ \beta_i & \text{if } p = q+1 \pmod 4 \\ 0^{d'} & \text{otherwise.} \end{cases}$$

Our reduction constructs sets $V_1, V_2, V_3, V_4 \subseteq \{0,1\}^{d+4d'}$, each of size $\binom{n}{2}$, as follows. For each $p \in \{1,2,3\}$, and $1 \leq i < j \leq n$, put into set V_p the vector

$$v_{i,j,p} := AND(v_i, v_j)||\chi(p,1,i,j)||\chi(p,2,i,j)||\chi(p,3,i,j)||\chi(p,4,i,j),$$

where $||$ denotes vector concatenation. Finally, for each $1 \leq i < j \leq n$, put into V_4 the vector

$$v_{j,i,4} := AND(v_i, v_j)||\chi(4,1,j,i)||\chi(4,2,j,i)||\chi(4,3,j,i)||\chi(4,4,j,i).$$

We picked the function χ so that, for $a, b, c, d \in \{1, 2, \ldots, n\}$,

- for $p \in \{1, 2, 3\}$, we have $\langle v_{a,b,p}, v_{c,d,p+1}\rangle = 0$ if and only if $a < b = c < d$ and

$$\langle AND(v_{i_a}, v_{i_b}), AND(v_{i_b}, v_{i_d})\rangle = \langle v_{i_a}, v_{i_b}, v_{i_d}\rangle = 0.$$

- For $p = 4$, we have $\langle v_{a,b,4}, v_{c,d,1}\rangle = 0$ if and only if $a > b = c < d$ and $\langle v_{i_a}, v_{i_b}, v_{i_d}\rangle = 0$.

In other words, there are $a, b, c, d \in \{1, 2, \ldots, n\}$ such that $\langle v_{a,b,1}, v_{b,c,2}\rangle = \langle v_{b,c,2}, v_{c,d,3}\rangle = \langle v_{c,d,3}, v_{d,a,4}\rangle = \langle v_{d,a,4}, v_{a,b,1}\rangle = 0$ if and only if $1 \leq a < b < c < d \leq n$ and, for every $S \subseteq \{v_{i_a}, v_{i_b}, v_{i_c}, v_{i_d}\}$ of size $|S| = 3$ we have $\mathsf{IP}(S) = 3$. In other words, the sets V_1, V_2, V_3, V_4 form the desired instance of $\mathsf{DIRECTED\text{-}CYCLE}_{\binom{n}{2}, d+4d', 4}$. ◀

Now we highlight the main changes to the above proof needed to obtain Theorem 24.

Sketch of the proof of Theorem 24. We start with $\mathsf{OV\text{-}HYPERGRAPH}_{n,d,k,3}$ where we are given as input a size-n set $V = \{v_1, \ldots, v_n\} \subseteq \{0,1\}^d$, and we want to determine whether there is a $T \subseteq V$ of size $|T| = k$ such that, for all $S \subseteq T$ of size $|S| = 3$, we have $\mathsf{IP}(S) = 0$.

Let $\gamma = k - \lceil k/3 \rceil$.

The reduction from k-hyperclique in 3-uniform hypergraphs to k-hypercycle in 3-uniform hypergraphs from [19] carries over to our case and we see (details in the full version) that $\mathsf{OV\text{-}HYPERGRAPH}_{n,d,k,3}$ reduces to the following OV Hypercycle problem: given k sets of vectors $W_1, \ldots, W_k$ where $W_i \subseteq \{0,1\}^d$ with $|Q_i| = n$, are there $a_1 \in W_1, \ldots, a_k \in Q_k$ so that for every $j \in \{1, \ldots, k\}$, we have that $\langle a_j, a_{j+1}, \ldots, a_{j+\gamma}\rangle = 0$, where indices are taken mod k.

We now take this hypercycle problem with generalized inner product over $(\gamma+1)$-tuples and reduce it to k-cycle in a graph with roughly n^γ vertices extending our construction for 4-cycle.

We extend the definition of AND to take the bitwise AND of a γ-tuple of d length bit vectors. We then select for sufficiently high d', $2n^{\gamma-1}$ vectors $\alpha_1, \ldots, \alpha_{n^{\gamma-1}}, \beta_1, \ldots, \beta_{n^{\gamma-1}} \in \{0,1\}^{d'}$ such that, for $i, j \in \{1, 2, \ldots, n\}^{\gamma-1}$ we have

$$\langle \alpha_i, \beta_j\rangle = \begin{cases} 0 & \text{if } i = j \\ \geq 1 & \text{otherwise.} \end{cases}$$

This can be done as before by picking each α_i to be a different vector with exactly half its entries 1, and picking β_i to be $\bar{\alpha_i}$. Then we get $d' = O(\log n)$ by picking the smallest d' with $\binom{d'}{d'/2} \geq n^{\gamma-1}$. We define $\chi : \{1, 2, \ldots, k\}^2 \times \{1, 2, \ldots, n\}^\gamma \to \{0,1\}^{d'}$ by the below, where $i_1, \ldots, i_\gamma \in \{1, \ldots, n\}$:

$$\chi(p, q, i_1, \ldots, i_\gamma) = \begin{cases} \alpha_{i_2, \ldots, i_\gamma} & \text{if } p = q \\ \beta_{i_1, \ldots, i_{\gamma-1}} & \text{if } p = q + 1 \pmod k \\ 0^{d'} & \text{otherwise.} \end{cases}$$

Our reduction constructs sets $V_1, V_2, \ldots, V_k \subseteq \{0,1\}^{d+kd'}$, each of size roughly n^γ, as follows. For each $p \in \{1, 2, \ldots, k\}$, and each γ-tuple $i_1, \ldots, i_\gamma \in \{1, \ldots, n\}$ put into set V_p the vector

$$v_{i_1,\ldots,i_\gamma,p} := AND(v^p_{i_1}, \ldots, v^p_{i_\gamma}) || \chi(p, 1, v^p_{i_1}, \ldots, v^p_{i_\gamma}) || \chi(p, 2, v^p_{i_1}, \ldots, v^p_{i_\gamma}) || \ldots || \chi(p, k, v^p_{i_1}, \ldots, v^p_{i_\gamma}),$$

where $||$ denotes vector concatenation and v^p_j is the jth node of W_p from the hypercycle instance.

We picked the function χ so that,

$$\langle v_{i_1,\ldots,i_\gamma,p}, v_{j_1,\ldots,j_\gamma,p+1} \rangle = 0$$

if and only if $(v^p_{i_2}, \ldots, v^p_{i_\gamma}) = (v^{p+1}_{i_1}, \ldots, v^{p+1}_{i_{\gamma-1}})$ and

$$\langle AND(v^p_{i_1}, \ldots, v^p_{i_\gamma}), (v^{p+1}_{j_1}, \ldots, v^{p+1}_{j_\gamma}) \rangle = \langle v^p_1, v^p_2, \ldots, v^p_\gamma, v^{p+1}_\gamma \rangle = 0.$$

In other words, there is an OV k-cycle if and only if there was an OV k-hypercycle. ◀

We obtain the following corollary:

▶ **Corollary 26.** *Suppose that there is some $\varepsilon > 0$ so that DIRECTED-CYCLE$_{n,O(\log n),k}$ can be solved in*

- *$O(n^{3/2-\varepsilon})$ time for some $k \geq 6$ divisible by 3, or*
- *$O(n^{3/2+1/(2\ell)-\varepsilon})$ time for some $k = 3\ell + 1 \geq 4$, or*
- *$O(n^{3/2+1/(4\ell+2)-\varepsilon})$ time for some $k = 3\ell + 2 \geq 5$.*

Then, for any constant $a > 0$, MAX-3-SAT$_{n,an}$ can be solved in (deterministic) time $O(2^{(1-\varepsilon/4+\delta)n})$ for every $\delta > 0$.

In particular, if DIRECTED-CYCLE$_{n,O(\log n),4}$ can be solved in time $n^{2-\varepsilon}$ time for some $\varepsilon > 0$, then MAX-3-SAT$_{n,an}$ has a faster algorithm.

5 Consequences for dynamic problems

In this section we will give two reductions that show that under Hypothesis 4, dynamic graph problems are hard even in OV graphs. We will need two tools to prove our theorems.

The first is from the previous section. It says that for any n we can construct $2n$ Boolean vectors of length $O(\log n)$, $\alpha_1, \ldots, \alpha_n, \beta_1, \ldots, \beta_n$ so that $\langle \alpha_i, \beta_j \rangle = 0$ if and only if $i = j$.

The second tool is a way to make an OV graph "layered".

▷ Claim 27. For any $\ell \geq 2$, there exist ℓ nonzero Boolean vectors $a_1, \ldots, a_\ell$ of length $2 + (\ell - 2)(\ell - 1)/2$ so that for each i, j, $\langle a_i, a_j \rangle = 0$ if and only if $j = i + 1$ or $i = j + 1$.

Proof. We proceed by induction. For $\ell = 2$, the vectors are $[1, 0]$ and $[0, 1]$. Suppose the vectors for ℓ are $a_1, \ldots, a_\ell$ and have length L, then for every $j < \ell$, replace a_j with a_j concatenated with the length $\ell - 1$ bit vector that is all 0 except in position j in which it is 1. Replace a_ℓ with a_ℓ concatenated with the length $\ell - 1$ all 0s vector. Finally set $a_{\ell+1}$ to be the length L all 0s vector concatenated with the length $\ell - 1$ all 1s vector.

After the vector replacement, $\langle a_j, a_{\ell+1} \rangle > 0$ for all $j < \ell$, and $\langle a_i, a_j \rangle$ is the same as before the replacement for $i, j \leq \ell$, and $\langle a_\ell, a_{\ell+1} \rangle = 0$. The length of the vectors goes up by $\ell - 1$. If we assume inductively that $L = 2 + (\ell - 2)(\ell - 1)/2$, adding $\ell - 1$, we complete the proof as $(\ell - 2)(\ell - 1)/2 + (\ell - 1) = (\ell - 1)\ell/2$. ◁

▶ Remark 28. In fact, we will see in Theorem 32 below that the bound $2+(\ell-2)(\ell-1)/2$ in Claim 27 can be replaced by only $O(\log \ell)$. However, such an improvement is unimportant for the results in this section, in which we will always pick ℓ to be a constant.

First let us consider the dynamic *s-t* OV Shortest Paths which asks to maintain the distance between two fixed vertices s and t in an n node OV graph with dimension $O(\log n)$ under vertex relabel updates (change the vector representing a node).

▶ **Theorem 29.** *Under Hypothesis 4, dynamic s-t OV Shortest Paths with vertex label updates that change at most one edge at a time requires either $n^{\omega-o(1)}$ preprocessing time, or $n^{\omega-1-o(1)}$ amortized update time. If arbitrary vertex label updates are supported, then $5/3-\varepsilon$-approximating the s-t distance requires $n^{\omega-1-o(1)}$ amortized update time even when starting with an empty graph (all vectors are all 1s).*

Proof. Suppose we are given an instance of OVΔ with n vectors V of dimension $d = O(\log n)$.

We will start with the proof for vertex label updates that change at most a single edge.

Let us create 4 sets of vectors V_1, V_2, V_3, V_4, using the vectors $a_1, a_2, \ldots, a_6$ from Claim 27: for each $i \in \{1, \ldots, 4\}$ and for every $v \in V$, put in V_j the vector v^j that concatenates v with a_{j+1}. In addition, create two new vectors, s and t, where s is the d-length all 1s vector concatenated with a_1 and t is the d-length all 1s vector concatenated with a_6.

By construction, s and t are non-orthogonal with all other vectors, and a vector $v^j \in V_j$ and a vector $u^k \in V_k$ for $j \leq k$ are orthogonal if and only if $k = j+1$ and u and v are orthogonal in the original OVΔ instance.

We will use the length $d' = O(\log n)$ vectors $\alpha_1, \ldots, \alpha_n, \beta_1, \ldots, \beta_n$ from our first tool. For every vector $v^j \in V^j$ for $j \in \{2, 3\}$, we concatenate the length d' all 0s vector to its end. For a vector $v^j_q \in V^j$ for $j \in \{1, 4\}$ (where v_q is the qth vector of V) we concatenate β_q to the end of it. This does not change the orthogonality of the vectors. We also concatenate the length d' all zeros vector to the ends of both s and t.

This completes the preprocessing stage.

Now, the updates are as follows. We have n stages, one for each vector $v_q \in V$. In the stage for v_q, we replace the last d' bits of s and t with α_q and the first d bits with the all 0 vector. This has the effect of inserting the edges (s, v^1_q) and (v^4_q, t) and leaving all other edges unchanged.

Then, the distance between s and t in the new OV graph is 5 if and only if there are vectors $a, b \in V$ such that $\langle v_q, a\rangle = \langle a, b\rangle = \langle b, v_q\rangle = 0$. At the end of the stage for v_q, we undo the relabeling of s and t and we move on to the next vector in V. This effectively deletes the two edges that were inserted.

The number of stages is n, and so if the preprocessing time is $O(n^{\omega-\varepsilon})$ for $\varepsilon > 0$, then the total time of all n update stages needs to be $n^{\omega-o(1)}$ under our Hypothesis, and hence the amortized time per update is $n^{\omega-1-o(1)}$.

Now we present a simpler proof where the updates are arbitrary relabelings. In the previous proof since the edges in the initial graph would be too expensive to insert one by one using single edge updates, we needed preprocessing. Here there will be no need for preprocessing since we can simply label all vertices as needed one by one.

Create just two sets of vectors V_1, V_2, using the vectors a_1, a_2, a_3, a_4 from Claim 27: for each $i \in \{1, 2\}$ and for every $v \in V$, put in V_j the vector v^j that concatenates v with a_{j+1}. In addition, create two new vectors, s and t, where s is the d-length all 1s vector concatenated with a_1 and t is the d-length all 1s vector concatenated with a_4. Then we will have n stages one for each vector $v_q \in V$ in which we will change s and t as follows. In the stage for v_q, we replace the first d bits of s and t with v_q. Now, s will only be orthogonal to the vectors in V_1

corresponding to original vectors that are orthogonal to v_q and t will be orthogonal to the vectors in V_2 corresponding to original vectors that are orthogonal to v_q. Thus the distance between s and t is 3 if there is a triangle in the OV graph, and it is at least 5 otherwise. ◀

The second dynamic problem we will prove hardness for is dynamic bipartite perfect matching in OV graphs: given an OV graph maintain whether it has a perfect matching, under vertex relabel updates.

▶ **Theorem 30.** *Dynamic bipartite perfect matching in OV graphs on n nodes and dimension $O(\log n)$ under vertex relabel updates that change only a single edge at a time requires either $n^{\omega-o(1)}$ preprocessing time, or $n^{\omega-1-o(1)}$ amortized update time. If arbitrary vertex relabelings are supported, the same lower bound holds but with arbitrary preprocessing.*

Proof. The proof is similar to the proof of the previous theorem.

We start with an instance of OVΔ, and make it layered with 10 layers, $s, V^{1,1}, V^{1,2}, V^{2,1}, V^{2,2}, V^{3,1}, V^{3,2}, V^{4,1}, V^{4,2}, t$. Here s and t are vertices and the rest contain vectors corresponding to the vectors in V. The layering is accomplished using Claim 27 as in the previous theorem. This adds a constant number of coordinates to the vectors.

For a particular j, the edges between $V^{j,1}$ and $V^{j,2}$ are just a matching between the vectors $v_q^{j,1}$ and $v_q^{j,2}$ corresponding to a vector $v_q \in V$. This is accomplished using the vectors α_i and β_i from our first tool.

The edges between $V^{j,2}$ and $V^{j+1,1}$ are between vectors $v_q^{j,2}$ and $v_r^{j+1,1}$ that correspond to $v_q, v_r \in V$ that are orthogonal.

The above two tasks are accomplished as follows. Let $L = O(1)$ be the length of the vectors a_j for $j \in \{1, \ldots, 12\}$ from Claim 27 . Let $d = O(\log n)$ be the length of the vectors from the original OVΔ instance and let $d' = O(\log n)$ be the length of the vectors $\alpha_1, \ldots, \alpha_n, \beta_1, \ldots, \beta_n$ from our first tool. Let 0_t denote the length-t all 0s vector. Each of the vectors $v_q^{j,b}$ for $j \in \{1,2,3,4\}, b \in \{1,2\}$ is of length $L + 2d' + 2d$ as follows. If j is odd, a vector $v_q^{j,1}$ is the concatenation of a_{2j+2}, $0_{d'}$, α_q, v_q and 0_d, and a vector $v_q^{j,2}$ is the concatenation of a_{2j+3}, $0_{d'}$, β_q, 0_d and v_q. On the other hand, if j is even, a vector $v_q^{j,1}$ is the concatenation of a_{2j+2}, α_q,$0_{d'}$, 0_d and v_q, and a vector $v_q^{j,2}$ is the concatenation of a_{2j+3}, β_q, $0_{d'}$, v_q and 0_d.

We make s and t orthogonal to each other but not to anything else. This can be done by letting s be a_1 followed by $0_{2d'+2d}$ and t be a_2 followed by $0_{2d'+2d}$. Since all other vectors start with a_k for $k \geq 4$, s and t are not orthogonal to any of the other vectors but are orthogonal to each other.

This completes the preprocessing stage. At this point there is a perfect matching consisting of (s,t) and the perfect matchings between $V^{j,1}$ and $V^{j,2}$ for $j \in \{1,2,3,4\}$.

There is a stage for each $v_q \in V$. In the stage for v_q, we make s be in layer 1 and only orthogonal to $v_q^{1,1}$, and t be in layer 10 and only orthogonal to $v_q^{4,2}$. This effectively deletes the edge (s,t) and inserts edges $(s, v_q^{1,1})$ and $(v_q^{4,2}, t)$. (These updates are undone at the end of the stage.)

Since $v_q^{1,1}$ is the concatenation of $a_4, \alpha_q, 0_{d'}, 0_d, v_q$ and $v_q^{4,2}$ is the concatenation of $a_{11}, \beta_q, 0_{d'}, v_q, 0_d$, we only need to set s to be the concatenation of $a_3, \beta_q, 0_{d'}, 0_d, 0_d$ and t to be the concatenation of $a_{12}, \alpha_q, 0_{d'}, 0_d, 0_d$.

Now the only way that for there to be a perfect matching is if there are $v_a, v_b \in V$ so that v_q, v_a, v_b form a triangle in the original OV graph. The number of updates is $O(n)$, and hence we get the same conditional lower bound as in the previous theorem. ◀

6 Problems still NP-hard on OV graphs

We begin by recalling Alon's original proof that graphs with low maximum degree can be written as OV graphs of low dimension:

▶ **Lemma 31** ([5, Lemma 3.2]). *For any graph G with n nodes and maximum degree d, there are n vectors in $\{0,1\}^{O(d^2 \log n)}$ whose OV graph is G.*

Proof. (We give here the original proof of [5].) Let $k = \lceil 2e^2(d+1)^2 \ln n \rceil$. We begin by picking n vectors $v_1, \ldots, v_n \in \{0,1\}^k$ at random: for each $i \in \{1,2,\ldots,n\}$ and each $\ell \in \{1,2,\ldots,k\}$ we set $v_i[\ell] = 1$ independently with probability $1/(d+1)$, and $v_i[\ell] = 0$ otherwise. Next, we perform a "correction": for each $i,j \in \{1,2,\ldots,n\}$ which are adjacent nodes in G, and each $\ell \in \{1,2,\ldots,k\}$ such that $v_i[\ell] = v_j[\ell] = 1$, we set both $v_i[\ell]$ and $v_j[\ell]$ to 0. After this correction, every edge in G is also an edge in the OV graph for $V = \{v_1, \ldots, v_n\}$.

Now, consider any $i,j \in \{1,2,\ldots,n\}$ which are not adjacent in G. They are also not adjacent in the OV graph of V so long as there is an $\ell \in \{1,2,\ldots,k\}$ such that (1) $v_i[\ell] = v_j[\ell] = 1$ before the correction phase, and (2) for every $i' \in \{1,2,\ldots,n\} \setminus \{i,j\}$ which is adjacent to i or j (there are at most $2d$ such i's), $v_{i'}[\ell] = 0$ before the correction phase. For a fixed $\ell \in \{1,2,\ldots,k\}$, this happens with probability

$$\frac{1}{(d+1)^2}\left(1 - \frac{1}{d+1}\right)^{2d} \geq \frac{1}{e^2(d+1)^2}.$$

Thus, for a fixed pair $i,j \in \{1,2,\ldots,n\}$ which are not adjacent in G, the probability that they are adjacent in the OV graph of V is at most

$$\left(1 - \frac{1}{e^2(d+1)^2}\right)^k \leq e^{-k/e^2(d+1)^2} \leq \frac{1}{n^2}.$$

It follows that the expected number of $i,j \in \{1,2,\ldots,n\}$ which are not adjacent in G but are adjacent in the OV graph of V is at most $\binom{n}{2} \cdot n^{-2} < 1/2$. Hence, by the probabilistic method, there is a choice of randomness for which there are no such i,j, and hence the OV graph of V is exactly our graph G, as desired. ◀

▶ **Theorem 32.** *Given a graph G with n nodes and maximum degree d, there is a deterministic algorithm running in time $n^{O(d^2)}$ to find n vectors in $\{0,1\}^{O(d^2 \log n)}$ whose OV graph is G.*

Proof. We will derandomize the construction from Lemma 31. We first note that, for a fixed ℓ, when picking $v_i[\ell]$ for all i before the correction phase, it is sufficient to pick them $(2d+2)$-wise independently, rather than fully independently as we did in the original proof. Indeed, in the proof that any $i,j \in \{1,2,\ldots,n\}$ which are not adjacent in G are also not adjacent in the OV graph of V, we only needed to consider $v_{i'}[\ell]$ for $(2d+2)$ choices of i'. For a fixed ℓ, we can thus draw the required $v_i[\ell]$ before the correction phase for all i, $(2d+2)$-wise independently, to be 1 with probability $1/(d+1)$, and 0 otherwise, using $O(d \log n)$ random bits via standard constructions.

Next, rather than independently sample such a vector for each ℓ, we use the standard derandomization trick of using a random walk on a constant-degree expander graph (see e.g. [25, Section 4.2]). We saw above that for a given $i,j \in \{1,2,\ldots,n\}$ which are not adjacent in G, the required vector entries for a particular ℓ can be drawn with $O(d \log n)$ random bits, and will cause them to be not adjacent in the OV graph of V with probability $1/O(d^2)$. Hence, using an expander random walk, we can draw the vector entries for $t = O(d^2 \log n)$ different ℓs using $O(t + d \log n) = O(d^2 \log n)$ random bits. As before, there is a choice of randomness for which the resulting OV graph is G. We can iterate over all $2^{O(d^2 \log n)} = n^{O(d^2)}$ choices of randomness to find such an OV graph as desired. ◀

This immediately gives a polynomial-time reduction from algorithmic problems on sparse graphs to algorithmic problems on OV graphs of dimension $O(\log n)$. For instance:

▶ **Corollary 33.** *Any problem which is NP-hard on graphs of constant maximum degree is also NP-hard on OV graphs of dimension $O(\log n)$.*

Finally, we note that some NP-hard graph problems do become easier on OV graphs:

▶ **Theorem 34.** *Given as input vectors $V = \{v_1, \ldots, v_n\} \subseteq \{0,1\}^d$ defining a dimension d OV graph G_V, a maximum size clique in G_V can be found in time $2^d \cdot n^{O(1)}$.*

Proof. A clique in G_V of size s corresponds to a subset $S \subseteq V$ of size $|S| = s$ such that, for each $i \in \{1, 2, \ldots, d\}$, there is at most one $v \in V$ such that $v[i] = 1$. Equivalently, if we view each $v \in V$ as a subset of $U := \{1, 2, \ldots, d\}$ (i.e. $i \in v$ if and only if $v[i] = 1$), then $S \subseteq V$ is a clique if and only if it is a collection of disjoint subsets of U. Finding the maximum size clique is hence the set packing problem in a universe of size d, which can be solved in $2^d \cdot n^{O(1)}$ time [7]. ◀

7 OV Matrices and Online Matrix-Vector Multiplication

▶ **Theorem 35.** *For any real numbers a, c with $c > 2a > 0$, and any matrix $M \in \mathbb{F}_2^{n \times n}$ of OV dimension $c \cdot \log n$, we can write $M = L \cdot R + S$ where $L, R^T \in \mathbb{F}_2^{n \times r}$ for $r = n^{a \log(c/a) + o(1)}$, and $S \in \mathbb{F}_2^{n \times n}$ has at most $O(n^{2-a})$ entries equal to 1. Moreover, given the vectors $u_1, \ldots, u_n, v_1, \ldots, v_n \in \{0,1\}^{c \log n}$ such that $M[i,j] = 1$ if and only if $\langle u_i, v_j \rangle = 0$ (over $\mathbb{Z}$), we can compute L, R and S in randomized time $\tilde{O}(n^2)$ with high probability.*

Proof. We use the polynomial method similar to [2, Theorem 1.1]. Let $d = c \log n$. For every subset $S \subseteq \{1, 2, \ldots, d\}$, define the polynomial $p_S : \mathbb{F}_2^d \to \mathbb{F}_2$ by $p_S(x) = \sum_{\ell \in S} x[\ell]$. Notice that, when $x = 0^d$, then $p_S(x) = 0$ for all S, and when $x \neq 0^d$, then $p_S(x) = 1$ for half of all S.

Let $m = a \log n$. We begin by picking m independent, uniformly random subsets $S_1, \ldots, S_m \subseteq \{1, 2, \ldots, d\}$. We then compute the set E of all pairs $(i,j) \in \{1, \ldots, n\}^2$ such that $AND(u_i, v_j) \neq 0^d$ but $p_{S_t}(AND(u_i, v_j)) = 0$ for all $t \in \{1, \ldots, m\}$. This can be computed in $O(n^2 md) = \tilde{O}(n^2)$ time. From the above discussion, a given (i,j) will be in E with probability $2^{-m} = n^{-a}$, and so the expected size of E is n^{2-a}. Hence, by Markov's inequality, $|E| \leq 100 \cdot n^{2-a}$ with probability at least 0.99; if this is not the case, repeat independently until it is.

Now, consider the polynomial $p : \mathbb{F}_2^d \to \mathbb{F}_2$ given by, for $z \in \mathbb{F}_2^d$, $p(z) = 1 + \prod_{t=1}^m (1 + p_{S_t}(z))$. Since p is over $\mathbb{F}_2$, with $x^2 = x$ for all $x \in \mathbb{F}_2$, we can assume that p is *multilinear* by reducing any exponent larger than 1 to 1. Hence, we can expand p into a sum of multilinear monomials, and the number r of monomials will be at most[1]

$$r \leq \sum_{g=0}^{m} \binom{d}{g} \leq O\left(\frac{d}{m}\right)^m = O\left(\frac{c \log n}{a \log n}\right)^{a \log n} \leq n^{a \log(c/a) + o(1)}.$$

[1] Here we use the fact, from Stirling's inequality, that for any $1 \leq k \leq n$ we have $\binom{n}{k} \leq \frac{n^k}{k!} \leq \left(\frac{e \cdot n}{k}\right)^k \leq O(n/k)^k$.

In particular, since for $x, y \in \mathbb{F}_2^d$ the vector $AND(x, y)$ consists of a single monomial in terms of x and y in each entry, i.e. $AND(x, y)[\ell] = x[\ell] \cdot y[\ell]$, we can write

$$p(AND(x, y)) = \sum_{w=1}^{r} \prod_{\ell \in T_w} x[\ell] \cdot y[\ell]$$

for some subsets $T_1, \ldots, T_r \subseteq \{1, 2, \ldots, d\}$. Define the map $V : \mathbb{F}_2^d \to \mathbb{F}_2^r$ by $V(x)[w] = \prod_{\ell \in T_w} x[\ell]$. Hence, $p(AND(x, y)) = \langle V(x), V(y) \rangle$ (where the inner product is over $\mathbb{F}_2$).

Finally, we can define our matrices L, R, S as follows: $L \in \mathbb{F}_2^{n \times r}$ is the matrix whose ith row is $V(u_i)$, $R \in \mathbb{F}_2^{r \times n}$ is the matrix whose jth column is $V(v_j)$, and $S \in \mathbb{F}_2^{n \times n}$ is the matrix whose entry $S[i, j]$ is 1 if and only if $(i, j) \in E$. Hence, we can see that $(L \cdot R)[i, j] = p(AND(u_i, v_j))$, and we have defined E so that $S[i, j] = 1$ if and only if $p(AND(u_i, v_j)) \neq M[i, j]$. ◀

▶ **Corollary 36.** *For $c > 0$, given vectors $u_1, \ldots, u_n, v_1, \ldots, v_n \in \{0,1\}^{c \log n}$ which define a matrix $M \in \mathbb{F}_2^{n \times n}$ of OV dimension $c \log n$, we can compute matrices $L \in \mathbb{F}_2^{n \times n^{0.1}}$, $R \in \mathbb{F}_2^{n^{0.1} \times n}$ and $S \in \mathbb{F}_2^{n \times n}$ in time $\tilde{O}(n^2)$ with high probability such that S has at most $n^{2-1/O(\log c)}$ nonzero entries and $M = L \cdot R + S$.*

Proof. Apply Theorem 35 with $a = 1/O(\log c)$ so that $a \log(c/a) < 0.1$. ◀

▶ **Corollary 37.** *For $c > 0$, we can preprocess vectors $u_1, \ldots, u_n, v_1, \ldots, v_n \in \{0,1\}^{c \log n}$ which define a matrix $M \in \mathbb{F}_2^{n \times n}$ of OV dimension $c \log n$, in preprocessing time $\tilde{O}(n^2)$ with high probability, such that given as input a vector $v \in \mathbb{F}_2^n$, we can compute the product $M \cdot v$ in time $n^{2-1/O(\log c)}$.*

Proof. In preprocessing we compute the matrices L, R, S from Corollary 36. Then, on input v, we compute $L \cdot (R \cdot v)$ in $O(n^{1.2})$ time and $S \cdot v$ in $n^{2-1/O(\log c)}$ time, and sum the results. ◀

References

1. Amir Abboud and Virginia Vassilevska Williams. Popular Conjectures Imply Strong Lower Bounds for Dynamic Problems. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 434–443, 2014.
2. Amir Abboud, Ryan Williams, and Huacheng Yu. More applications of the polynomial method to algorithm design. In *Proceedings of the twenty-sixth annual ACM-SIAM symposium on Discrete algorithms*, pages 218–230. Society for Industrial and Applied Mathematics, 2015.
3. Josh Alman, Timothy M Chan, and Ryan Williams. Polynomial representations of threshold functions and algorithmic applications. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 467–476. IEEE, 2016.
4. Josh Alman and Ryan Williams. Probabilistic rank and matrix rigidity. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 641–652. ACM, 2017.
5. Noga Alon. Covering graphs by the minimum number of equivalence relations. *Combinatorica*, 6(3):201–206, 1986.
6. Noga Alon, Raphael Yuster, and Uri Zwick. Color-Coding. *J. ACM*, 42(4):844–856, 1995.
7. Andreas Björklund, Thore Husfeldt, and Mikko Koivisto. Set partitioning via inclusion-exclusion. *SIAM Journal on Computing*, 39(2):546–563, 2009.
8. Lijie Chen and Ryan Williams. An equivalence class for orthogonal vectors. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 21–40. SIAM, 2019.

9 Ruiwen Chen and Rahul Santhanam. Improved Algorithms for Sparse MAX-SAT and MAX-k-CSP. In *Theory and Applications of Satisfiability Testing - SAT 2015 - 18th International Conference, Austin, TX, USA, September 24-27, 2015, Proceedings*, pages 33–45, 2015.

10 Marek Cygan, Marcin Pilipczuk, and Michał Pilipczuk. Known algorithms for edge clique cover are probably optimal. *SIAM Journal on Computing*, 45(1):67–83, 2016.

11 Evgeny Dantsin and Alexander Wolpert. MAX-SAT for formulas with constant clause density can be solved faster than in $o(s^2)$ time. In *Theory and Applications of Satisfiability Testing - SAT 2006, 9th International Conference, Seattle, WA, USA, August 12-15, 2006, Proceedings*, pages 266–276, 2006.

12 Paul Erdös, Adolph W Goodman, and Louis Pósa. The representation of a graph by set intersections. *Canadian Journal of Mathematics*, 18:106–112, 1966.

13 François Le Gall. Powers of tensors and fast matrix multiplication. In *International Symposium on Symbolic and Algebraic Computation, ISSAC 2014, Kobe, Japan, July 23-25, 2014*, pages 296–303, 2014.

14 Jens Gramm, Jiong Guo, Falk Hüffner, and Rolf Niedermeier. Data reduction and exact algorithms for clique cover. *Journal of Experimental Algorithmics (JEA)*, 13:2, 2009.

15 András Gyárfás. A simple lower bound on edge coverings by cliques. *Discrete Mathematics*, 85(1):103–104, 1990.

16 Russell Impagliazzo and Ramamohan Paturi. On the Complexity of k-SAT. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001.

17 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001.

18 Kasper Green Larsen and Ryan Williams. Faster online matrix-vector multiplication. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2182–2189. Society for Industrial and Applied Mathematics, 2017.

19 Andrea Lincoln, Virginia Vassilevska Williams, and Ryan Williams. Tight hardness for shortest cycles and paths in sparse graphs. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1236–1252. SIAM, 2018.

20 László Lovász. On the Shannon capacity of a graph. *IEEE Transactions on Information theory*, 25(1):1–7, 1979.

21 László Lovász. *Graphs and geometry*. American Mathematical Society, 2019.

22 László Lovász, Michael Saks, and Alexander Schrijver. Orthogonal representations and connectivity of graphs. *Linear Algebra and its applications*, 114:439–454, 1989.

23 TS Michael and Thomas Quint. Sphericity, cubicity, and edge clique covers of graphs. *Discrete Applied Mathematics*, 154(8):1309–1313, 2006.

24 Sylvia D Monson, Norman J Pullman, and Rolf Rees. A survey of clique and biclique coverings and factorizations of (0, 1)-matrices. *Bull. Inst. Combin. Appl*, 14:17–86, 1995.

25 Salil P Vadhan et al. Pseudorandomness. *Foundations and Trends® in Theoretical Computer Science*, 7(1–3):1–336, 2012.

26 Virginia Vassilevska Williams. Multiplying matrices faster than Coppersmith-Winograd. In *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 - 22, 2012*, pages 887–898, 2012.

27 Virginia Vassilevska Williams. On some fine-grained questions in algorithms and complexity. In *Proceedings of the ICM*, 2018.

28 Ryan Williams. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theoretical Computer Science*, 348(2-3):357–365, 2005.

29 Ryan Williams. *Algorithms and resource requirements for fundamental problems*. PhD thesis, Ph. D. Thesis, Carnegie Mellon University, CMU-CS-07-147, 2007.

30 Raphael Yuster and Uri Zwick. Finding Even Cycles Even Faster. *SIAM J. Discrete Math.*, 10(2):209–222, 1997.

Finding Skewed Subcubes Under a Distribution

Parikshit Gopalan
VMware Research, Palo Alto, CA, USA
pgopalan@vmware.com

Roie Levin[1]
Carnegie Mellon University, Pittsburgh, PA, USA
roiel@cs.cmu.edu

Udi Wieder
VMware Research, Palo Alto, CA, USA
uwieder@vmware.com

Abstract

Say that we are given samples from a distribution ψ over an n-dimensional space. We expect or desire ψ to behave like a product distribution (or a k-wise independent distribution over its marginals for small k). We propose the problem of enumerating/list-decoding all large subcubes where the distribution ψ deviates markedly from what we expect; we refer to such subcubes as skewed subcubes. Skewed subcubes are certificates of dependencies between small subsets of variables in ψ. We motivate this problem by showing that it arises naturally in the context of algorithmic fairness and anomaly detection.

In this work we focus on the special but important case where the space is the Boolean hypercube, and the expected marginals are uniform. We show that the obvious definition of skewed subcubes can lead to intractable list sizes, and propose a better definition of a minimal skewed subcube, which are subcubes whose skew cannot be attributed to a larger subcube that contains it. Our main technical contribution is a list-size bound for this definition and an algorithm to efficiently find all such subcubes. Both the bound and the algorithm rely on Fourier-analytic techniques, especially the powerful hypercontractive inequality.

On the lower bounds side, we show that finding skewed subcubes is as hard as the sparse noisy parity problem, and hence our algorithms cannot be improved on substantially without a breakthrough on this problem which is believed to be intractable. Motivated by this, we study alternate models allowing query access to ψ where finding skewed subcubes might be easier.

2012 ACM Subject Classification Mathematics of computing → Probabilistic algorithms

Keywords and phrases Fourier Analysis, Anomaly Detection, Algorithmic Fairness, Probability, Unsupervised Learning

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.84

Related Version A full version of the paper is available at `https://arxiv.org/abs/1911.07378`.

1 Introduction

Assume that we observe samples from a distribution ψ over points in n-dimensional space $\mathcal{D}^n$. Our prior belief is that each attribute has a marginal distribution μ_i and that the various attributes are nearly independent (or at least k-wise independent for small k), hence ψ is close to the product distribution $\mu = \prod_i \mu_i$. Our goal is to find significant deviations between our hypothesis μ and the observed distribution ψ, manifested as significant dependencies between small sets of variables. The distribution μ might represent either a prior model for

[1] Work done while an intern at VMware Research.

ψ, or it might be represent a target distribution that we wish ψ to be close to. This problem arises naturally in several machine learning applications as we detail in Section 1.1, but first we formulate the problem with more detail.

To formulate a precise statement, we first define the notion of subcubes. Assume that $\mathcal{D}$ is ordered and bounded, the two canonical examples are $\mathcal{D}^n = \{0,1\}^n$ and $\mathcal{D}^n = [0,1]^n$. Let $K \subseteq [n]$ be a set of k coordinates. For $x \in \mathcal{D}^n$, x_K denotes the projection of x onto coordinates in K. For each $j \in K$, let $I_j \subsetneq \mathcal{D}$ be an interval in $\mathcal{D}$. We call the set of points $C = \{x \in \mathcal{D}^n : x_K \in \prod_{j \in K} I_j\}$ a subcube of codimension k. We have

$$\mu(C) := \Pr_{\mathbf{x} \sim \mu}[\mathbf{x} \in C] = \prod_{j \in K} \Pr_{\mathbf{x}_j \sim \mu_j}[\mathbf{x}_j \in I_j] = \prod_{j \in K} \mu_j(I_j).$$

If we similarly define $\psi(C) := \Pr_{\mathbf{x} \sim \psi}[\mathbf{x} \in C]$, then our goal is to find subcubes such that $|\mu(C) - \psi(C)| \geq \gamma$. Motivated by our applications, we add two more desiderata to our problem formulation (that will be justified shortly): we restrict to large subcubes, and we want algorithms that enumerate all subcubes that satisfy our conditions.

One way to restrict to large subcubes is to only consider subcubes with $\mu(C) \geq \eta$ for some $\eta \in [0,1]$. Alternately, we could bound the codimension by k. The advantage of the latter is that we only need that μ is k-wise independent for the equality $\mu(C) = \prod_{j \in K} \mu_j(I_j)$ to hold. In the discrete case $\mathcal{D}^n = \{0,1\}^n$, the two notions coincide since $\mu(C) = 2^{-k}$ for subcubes of codimension k.

Rather then phrasing this as an optimization question where the goal is to find the subcube that maximizes the deviation γ, our goal will be to come up with a *list-decoding* style algorithm that enumerates over all subcubes of codimension k such that $|\mu(C) - \psi(C)| \geq \gamma$.

In addition to being a natural algorithmic question in its own right, this problem comes up in recent work in machine learning, on anomaly detection and fairness.

1.1 Motivation

Fairness in Machine Learning

Assume there is a base population P of individuals, each described by n attributes. We naturally view P as inducing a distribution μ on the attribute space $\mathcal{D}^n$. Suppose that small subsets of the attributes are nearly independent, so that μ is close to being k-wise independent for some k which is small compared to n. We are given a distribution ψ over this population. Our goal is to discover significant biases in the distribution that are not present in the original population P. For instance the population P might be the set of students that apply to a university, and ψ might represent the set of successful applicants. Or P might be the training data for a machine learning algorithm while ψ represents the misclassified inputs. The latter setting has received a fair amount of attention in the context of algorithmic bias and fairness in Machine learning, where the most commonly studied notion is that of intersectionality bias [4]: we are interested in biases where we restrict the values of some small subset of attributes, which are typically discrete. See for instance a recent study showing that facial recognition software has higher error rates for women of color [3]. Our motivation for considering subcubes is that it captures intersectionality in the discrete setting.

Enumerating over all subcubes is more appropriate than optimization in this setting since not all intersectionalities might be equally important. The fact that college applications submitted during certain days of the week are less likely to be accepted might not be as significant as the fact that certain zipcodes are less likely to be accepted; even if the deviation

is lower in the latter case. We ask for algorithms that enumerate over all biased subcubes and leave it to subject experts to decide how interesting these are, just as in list-decoding we do not worry about how the receiver chooses from the list of possible codewords returned by the decoder. Another reason to favor enumeration is that in real-world datasets, we may not expect ψ to be truly k-wise independent; we might expect correlations between certain sets of attributes. But even so, an exhaustive list of significant correlations might lead us to discover interesting new properties of the distribution and refine our model for ψ. The restriction to subcubes of bounded codimension is natural since intersectionalities of few attributes are more interesting.

Anomaly Detection

Anomaly detection is a ubiquitous unsupervised learning problem [5]. Isolation based methods for anomaly detection have proven to be extremely effective in practice [15, 7, 11]. Building on this, the recent work of [10] proposes an approach to anomaly detection based on a notion called Partial Identification. It assigns a score denoted $\mathrm{PIDScore}(x, P)$ to each point $x \in P$ which measures how easy it is to distinguish x from other points in P. They give a heuristic to compute $\mathrm{PIDScore}(x, P)$, and show that the resulting anomaly detection algorithm outperforms several popular anomaly detection methods, across a broad range of benchmarks.

Formally, given a set of points $P \subseteq \mathcal{D}^n$ and a subcube $C \in \mathcal{D}^n$, define the sparsity of C as

$$\rho(C) = \frac{\mathrm{vol}(C)}{|C \cap P|}$$

The PIDScore of a point $x \in P$ is the maximum value of $\rho(C)$ over all subcubes that contain it.

$$\mathrm{PIDScore}(x, P) = \max_{C \ni x} \rho(C).$$

Anomalous points are those for which $\mathrm{PIDScore}(x, P) \geq t$ for some threshold t. Equivalently, it suffices to find all C such that $\rho(C) \geq t$, and then take all the points contained in them.

To relate this to our problem, let us take μ to be the uniform measure over $\mathcal{D}^n$ and ψ to be the measure induced by P. Rescaling ρ by a factor of $|P|/\mathrm{vol}(\mathcal{D}^n)$, we get

$$\rho'(C) = \frac{\mathrm{vol}(C)}{\mathrm{vol}(\mathcal{D}^n)} \frac{|P|}{|C \cap P|} \approx \frac{\mu(C)}{\psi(C)}.^{2}$$

If we also scale the threshold t by the same factor, then the set of outliers stays the same. But $\rho'(C) \geq t'$ implies $\psi(C) \leq \mu(C)/t'$, hence

$$\frac{\mu(C) - \psi(C)}{\mu(C)} \geq 1 - \frac{1}{t'}.$$

Thus this is an instance of the problem that we consider, where our goal is to find non-empty subcubes that are underrepresented in ψ, when compared to μ. Enumeration over all sparse subcubes is natural in this setting, since we wish to list all points with high scores.

[2] Actually $|C \cap P|/|P| = \psi(C)$ only in expectation, but since subcubes have small VC dimension, we get tight concentration.

2 Our Results

In this paper, we focus on the case when $\mathcal{D}^n = \{\pm 1\}^n$ and μ is the uniform distribution. We believe that several of our techniques apply to more general product distributions. A subcube of codimension k is obtained by restricting the values of some subset K of coordinates. For subcubes $C \subseteq D$ we refer to C as a child of D and D as a parent of C.

As a warm-up we first consider the following problem:

▶ **Problem 1** (Finding skewed subcubes). *Given sample access to a distribution ψ over $\{\pm 1\}^n$ and $\gamma \in (0, 2^k - 1]$ find all subcubes C with codimension $j \leq k$ such that*

$$\left| \frac{\Pr_{\mathbf{x}\sim\psi}[\mathbf{x} \in C] - 2^{-j}}{2^{-j}} \right| \geq \gamma.$$

There is a trivial $\tilde{O}(n^k)$ algorithm that enumerates over all subcubes. To beat this naive bound, we first need to bound the list-size of the output, or rather a bound on the number of skewed subcubes. However, we show in Lemma 12 that there exist distributions where the number of skewed subcubes is $\Omega((n/k)^k)$, which is not far from the trivial upper bound.

The proof of Lemma 12 demonstrates that one source for the abundance of skewed subcubes is that skew is easily inherited by children from their parents: if a subcube C of codimension j is skewed, for every choice of $k - j$ additional coordinates, by simple averaging, there is at least one restriction that results in a skewed subcube. So even if we consider the uniform distribution over points with $x_1 = 1$, there are $\Omega(n^{k-1})$ skewed subcubes by this definition, while really the only interesting subcube is the $x_1 = 1$ subcube. Our first contribution is a definition which captures only those subcubes that do not inherit their skew from a parent.

▶ **Problem 2** (Finding minimal skewed subcubes). *Given sample access to a distribution ψ over $\{\pm 1\}^n$, $\gamma \in (0, 2^k - 1]$ and $\epsilon \in (0, 1)$ find all subcubes C with codimension $j \leq k$ such that*

$$\left| \frac{\Pr_{\mathbf{x}\sim\psi}[\mathbf{x} \in C] - 2^{-j}}{2^{-j}} \right| \geq \gamma.$$

and for every parent $C' \supsetneq C$ of codimension i,[3]

$$\left| \frac{\Pr_{\mathbf{x}\sim\psi}[\mathbf{x} \in C] - 2^{-i}}{2^{-i}} \right| \leq \gamma(1 - \epsilon).$$

We refer to such a subcube as a (γ, ϵ)-minimal skewed subcube. This notion is motivated by our applications: if we already know that $\Pr_{\mathbf{x}\sim\psi}[(\mathbf{x}_1 = 1) \wedge (\mathbf{x}_2 = 1)] = 3/4$ (rather than $1/4$), then knowing that $\Pr_{\mathbf{x}\sim\psi}[(\mathbf{x}_1 = 1) \wedge (\mathbf{x}_2 = 1) \wedge (\mathbf{x}_3 = 1)] = 3/8$ should not surprise us, given our prior.

A natural question to ask is whether focusing in minimal skewed subcubes suffices to make the problem (or at least the list size) more tractable. Our second contribution is a bound on the number of minimal skewed subcubes which is independent of the dimension n. Instead we have a dependence on the max norm of the probability distribution defined below.

[3] The formal definition of minimal skewed subcubes (Definition 13) is a little more involved, we only care about those parents of C which are skewed the same way as C.

Given a distribution ψ on $\{\pm 1\}^n$, let $\|\psi\|_\infty := 2^n \cdot \max_{x\in\{\pm1\}^n} \Pr_{\mathbf{x}\sim\psi}[\mathbf{x} = x]$. The parameter $\|\psi\|_\infty$ lies in the range $[1, 2^n]$ and is a measure of how well-spread the distribution is. It is referred to as the smoothness of a distribution in the literature on boosting, and is closely related to min-entropy. The uniform distribution has $t = 1$, whereas $t = 2^n$ when the entire distribution is concentrated on a single point.

▶ **Theorem 1.** *For a distribution ψ on $\{\pm1\}^n$, the number of (γ, ϵ)-minimal skewed subcubes of codimension at most k is bounded by $k^{O(k)} \left(\ln(e\,\|\psi\|_\infty)\,\mathrm{poly}(1/\epsilon, 1/\gamma)\right)^k$.*

For constant ϵ, γ, the asymptotic dependence on n is never worse than $O(n^k)$, which happens when ψ is concentrated on a point. But when $\|\psi\|_\infty = O(1)$, the above bound is $O_k(1)$ and when $\|\psi\|_\infty = \mathrm{poly}(n)$, the bound is $O_k(\ln(n)^k)$ improving substantially over the $O(n^k)$ bound.

There are two key elements in the proof of Theorem 1. We first use a novel Fourier based algorithm to reduce the problem to that of finding large, low-degree Fourier coefficients in a series of restrictions of the distribution ψ to various subcubes. We then use the powerful hypercontractive inequality to bound the number of such coefficients in any distribution in terms of $\|\psi\|_\infty$. This latter bound generalizes the level-k inequalities for indicators of small sets in the Boolean hypercube [16, Chapter 9], and the proof follows similar lines. We also construct distributions showing that for various values of ϵ, γ, the dependency of $(\ln(\|\psi\|_\infty))^k$ is optimal. The distributions are constructed using the Tribes function and BCH codes.

We now turn to the algorithmic problem of finding the list of minimal skewed subcubes. We observe that even when the list-size is constant, there is a significant algorithmic barrier to a $n^{o(k)}$ algorithm, namely the k-sparse noisy parity problem [8, 18]. In this problem, we are given points x and labels y which are the XOR of some k-subset S with random noise of rate η added. There is a simple reduction from this problem to finding skewed subcubes, if we consider the distribution of $(x, y) \in \{\pm1\}^{n+1}$ the only skewed subcubes involve the coordinates $S \cup \{n + 1\}$.

▶ **Theorem 2.** *For $\eta \in (0, 1/2)$, an algorithm that given a distribution ψ and k can find a $(1 - 2\eta, 1)$-minimal skewed subcube of co-dimension k in time $T(n, k, \eta)$ can be used to solve the k-sparse noisy parity problem with noise rate η in time $T(n, k, \eta)$.*

Given this reduction, there are two *lower bounds* on the running time of any list-decoder: the list-size given in Theorem 1, and the running time of the best known algorithm for the k-sparse noisy parity problem, which is $O(n^{0.8k})$ due to [18]. We give an algorithm that nearly gets the sum of these two bounds.

▶ **Theorem 3.** *For any measure ψ on $\{\pm1\}^n$, integer $k \le n$, and parameters $0 \le \gamma \le 2^k - 1$ and $0 \le \epsilon \le 1$, there are algorithms that return all (γ, ϵ)-minimal skewed subcubes of codimension at most k in time*

$$\tilde{O}\left(n^{0.8k}\right) + \tilde{O}\left(n^{k/3}\right) \cdot \frac{k^{O(k)}}{(\epsilon\gamma)^{4/\lambda+2k}} \left(\ln\left(\frac{e\,\|\psi\|_\infty}{\epsilon\gamma}\right)\right)^k$$

Finally, to circumvent the noisy parity problem, we consider stronger models where we have query access to the distribution ψ: in addition to random samples, we can also query the value of $\psi(x)$ for any $x \in \{\pm1\}^n$. The noisy parity problem becomes trivial to solve once one has query access. In this model, we are able to get an algorithm whose running time is is $\mathrm{poly}(n, \|\psi\|_\infty)$. Thus when $\|\psi\|_\infty < n^{\alpha k}$ for some $\alpha > 0$, this improves over the trivial algorithm. We show some dependence on $\|\psi\|_\infty$, possibly of the form $\ln(\|\psi\|_\infty)^k$ is inherent even in the query model, by constructing a distribution ψ (with large $\|\psi\|_\infty$) where the query model and random samples model are equivalent, and where finding skewed subcubes lets us solve the k-sparse noisy parity problem.

2.1 Related Work

In nearby work, [1] study the problem of testing whether a distribution is δ close in statistical distance to (ϵ, k)-wise uniform. In our language, a distribution D is said to be (ϵ, k)-wise uniform if all subcubes of codimension $j \leq k$ have skew no greater than $2^j\epsilon$. They provide a sample complexity upper bound of $O((k \log n)/\epsilon^2\delta^2)$. They then provide evidence, based on the conjectured hardness of finding planted cliques, that no polynomial in n time algorithm for this problem exists (one can also base this hardness on sparse noisy parity, as we do here). Indeed, their testing algorithm essentially reduces the problem to the optimization version: find the subcube of codimension k such that the skew is maximized.

Fourier analytic techniques have found widespread use in a variety of supervised learning problems under the uniform distribution [16]. Our work differs from this in that the problem we consider is an unsupervised learning problem, and that we use Fourier analysis over the uniform distribution to reason about the deviation from an arbitrary distribution. In this aspect, our work is similar to the work of [1, 17].

Finally, there have been a line of recent results in machine learning which have a list-decoding flavor to them, see for instance [6, 12].

Outline of the paper. Section 3 introduces definitions and notation. Section 4 contains Fourier analytic results that are required for our results. Section 5 proves our main combinatorial bounds on the number of skewed subcubes, and gives examples that show these bounds are tight. Section 6 describes the efficient algorithm for enumerating minimal skewed subcubes. Section 7 gives lower bounds due to the reduction from the noisy parity problem. Section 8 considers the problem in the membership query model. Some proofs are deferred from the main body to the Appendix 9.

3 Definitions

In this section we present basic definitions and facts.

Distributions

We denote the n-dimensional Hamming cube by $\{\pm 1\}^n$. Given a probability distribution ψ on $\{\pm 1\}^n$, it is convenient to identify it with the probability measure $\psi : \{\pm 1\}^n \to \mathbb{R}^{\geq 0}$ which satisfies

$$\mathop{\mathbb{E}}_{\mathbf{x}\sim\{\pm 1\}^n}[\psi(\mathbf{x})] = 1.$$

We write $\mathbf{x} \sim \psi$ to denote $\mathbf{x}$ is a random variable with the distribution

$$\Pr_{\mathbf{x}\sim\psi}[\mathbf{x} = x] = \frac{\psi(x)}{2^n}$$

Henceforth, we will interchangeably refer to ψ as a distribution and a measure. We will use μ to denote the uniform distribution over $\{\pm 1\}^n$, where $\mu(x) = 1$ for all $x \in \{\pm 1\}^n$. Given functions $f, g : \{\pm 1\}^n \to \mathbb{R}$, we define their inner product by $\langle f, g\rangle := \mathbb{E}_{\mathbf{x}\sim\mu}[f(\mathbf{x})g(\mathbf{x})]$. We define $\|f\|_p := \mathbb{E}_{\mathbf{x}\sim\mu}[f(\mathbf{x})^p]^{1/p}$. For two probability measures ψ, θ, we have

$$\langle \psi, \theta\rangle = \sum_{x\in\{\pm 1\}^n} \frac{\psi(x)\theta(x)}{2^n} = \mathop{\mathbb{E}}_{\mathbf{x}\sim\psi}[\theta(\mathbf{x})] = \mathop{\mathbb{E}}_{\mathbf{x}\sim\theta}[\psi(\mathbf{x})].$$

Let $A \subseteq \{\pm 1\}^n$ and let $\alpha = |A|/2^n$ denote its fractional density. We use μ_A to denote the uniform distribution over A. The corresponding measure is defined as

$$\mu_A(\mathbf{x}) = \begin{cases} \frac{1}{\alpha} & \text{if } x \in A \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

For a distribution ψ, we define

$$\|\psi\|_\infty = \max_{x \in \{\pm 1\}^n} \psi(x)$$

It follows from the definition that

$$\|\psi\|_\infty = 2^n \max_{x \in \{\pm 1\}^n} \Pr_{\mathbf{x} \sim \psi}[\mathbf{x} = x] = \max_{x \in \{\pm 1\}^n} \frac{\Pr_{\mathbf{x} \sim \psi}[\mathbf{x} = x]}{\Pr_{\mathbf{x} \sim \mu}[\mathbf{x} = x]}$$

A bound on $\|\psi\|_\infty$ implies that no point is too likely.

Subcubes

A subcube is a subset of $\{\pm 1\}^n$ obtained by fixing some subset of bits to a particular value. Formally, a subcube $C \subseteq \{\pm 1\}^n$ is specified by a pair (K, y) where $K \subseteq [n]$ and $y \in \{\pm 1\}^K$. We have

$$C = \{x \in \{\pm 1\}^n \text{ s.t. } x_i = y_i \ \forall\ i \in K\}.$$

We refer to coordinates in K as the fixed coordinates of C, and to the rest as the free coordinates of C. For $C = (K, y)$ we define the codimension of C to be $|K|$ and denote it $\mathrm{codim}(C)$. We use $\mathcal{C}^{\leq k}$ to denote the set of all subcubes of codimension at most k. By Equation (1), μ_C the uniform measure over C is given by

$$\mu_C(x) = \begin{cases} 2^{\mathrm{codim}(C)} & \text{if } x \in C \\ 0 & \text{if } x \notin C \end{cases}$$

For subcubes $C = (K, y), D = (L, z)$, we have $D \subset C$ iff $K \subset L$ and $z_i = y_i$ for all $i \in K$. We refer to D as a child of C and C as a parent of D.

▶ **Definition 4** (Restriction). *For a distribution ψ on $\{\pm 1\}^n$ and a subcube $C \subseteq \{\pm 1\}^n$ such that ψ assigns non-zero probability to C we define $\psi|_C : C \to \mathbb{R}^{\geq 0}$, the restriction of ψ to C, as*

$$\psi|_C(x) = \frac{\psi(x)}{\langle \psi, \mu_C \rangle}.$$

Since $\langle \psi, \mu_C \rangle = \mathbb{E}_{x \sim \mu_C}[\psi(x)]$, ψ assings non-zero probaility to C iff $\langle \psi, \mu_C \rangle > 0$. The restriction is itself a legal probability measure; it satisfies $\mathbb{E}_{x \in C}[\psi|_C] = 1$. This definition immediately implies the relationship:

▶ **Fact 5.** *For a distribution ψ on $\{\pm 1\}^n$ and a subcube $C \subset \{\pm 1\}^n$*

$$\|\psi|_C\|_\infty = \frac{\|\psi\|_\infty}{\langle \psi, \mu_C \rangle}$$

▶ **Lemma 6.** *Given subcubes C and D such that $D \subseteq C \subseteq \{\pm 1\}^n$, and a density function ψ, it holds that:*

$$\langle \psi, \mu_D \rangle = \langle \psi, \mu_C \rangle \cdot \langle \psi|_C, \mu_D|_C \rangle$$

3.1 Skewed subcubes

▶ **Definition 7** (Skew). *We define the **skew** of a subcube C with respect to measure ψ as*

$$\textsc{Skew}_\psi(C) = \langle \psi, \mu_C \rangle - 1$$

The next two lemmas state some simple facts about the skew of subcubes. First we show the skew of a subcube measures the deviation of the measure on the subcube from the uniform distribution.

▶ **Lemma 8.** *Let $\mathrm{codim}(C) = k$. We have*

$$\begin{aligned}\textsc{Skew}_\psi(C) &= 2^k \Pr_{\mathbf{x}\sim\psi}[\mathbf{x} \in C] - 1 \\ &= \frac{1}{\Pr_{x\sim\mu}[\mathbf{x} \in C]} \left(\Pr_{x\sim\psi}[\mathbf{x} \in C] - \Pr_{x\sim\mu}[\mathbf{x} \in C] \right).\end{aligned}$$

▶ **Corollary 9.** *For any distribution ψ, $\textsc{Skew}_\psi(C)$ lies in the range $[-1, 2^k - 1]$.*

If $\textsc{Skew}_\psi(C) < 0$, we say that C is negatively skewed while if $\textsc{Skew}_\psi(C) > 0$ we say that it is positively skewed. An averaging argument shows that the existence of negatively skewed subcubes implies the existence of positively skewed subcubes and vice versa.

▶ **Lemma 10.** *For any $K \subseteq [n]$, we have*

$$\sum_{\substack{D=(K,w) \\ w\in\{\pm 1\}^K}} \textsc{Skew}(D) = 0.$$

Given a cube $C = (K, y)$ of codimension k, we can partition it into 2^ℓ subcubes of codimension $k + \ell$, where we pick a set L of ℓ additional coordinates outside of K to fix and enumerate over all settings of these coordinates.

▶ **Lemma 11.** *If $\{C_1, \ldots, C_{2^\ell}\}$ is a partition of C, then*

$$\textsc{Skew}_\psi(C) = \frac{1}{2^\ell} \sum_{i=1}^{2^\ell} \textsc{Skew}_\psi(C_i).$$

Having established basic properties of the skew function, we next turn to bounding the number of subcubes with a given skew. We show that this number may be quite large in the worst case.

▶ **Lemma 12.** *Let $\gamma = 2^f - 1$ for $f \in \{1, \ldots, k\}$. There exists a distribution ψ such that there are $\Omega((n/k)^k)$ many subcubes of codimension k with $\textsc{Skew}(C) \geq \gamma$.*

Proof. Let C be the subcube where the first $t \geq f$ bits are fixed to 1, and let μ_C be the uniform distribution over it. Consider any subcube D where we choose f indices from $[t]$ and $k - f$ indices from $[n] \setminus [t]$, and set them to 1. We have

$$\langle \mu_C, \mu_D \rangle = \mathop{\mathbb{E}}_{\mu_C}[\mu_D] = 2^k \Pr_{\mathbf{x}\sim\mu_C}[\mathbf{x} \in D] = 2^k \frac{1}{2^{k-f}} = 2^f \geq 1 + \gamma$$

since a point from μ_C lies in D iff the $k - f$ bits from $[n] \setminus [t]$ are all set to 1. We now optimize the choice of t. Let $\alpha = f/k$ for $\alpha \leq 1$. We choose $t = \alpha n$ (ignoring floors and ceilings which will not affect the asymptotics). The number of choices for D is given by

$$\binom{t}{f} \cdot \binom{n-t}{k-f} = \binom{\alpha n}{\alpha k} \cdot \binom{(1-\alpha)n}{(1-\alpha)k} \geq \left(\frac{n}{k}\right)^{\alpha k} \cdot \left(\frac{n}{k}\right)^{(1-\alpha)k} \geq \left(\frac{n}{k}\right)^k. \qquad \blacktriangleleft$$

While the above bound is proved for positive skew, Lemma 10 can be used to derive a similar bound for negative skew. Given that this bound is not too far from the trivial upper bound of $\binom{n}{k}$, we need to refine our notion of skew, and also to restrict the set of distributions we consider.

3.2 Minimal skewed subcubes

Lemma 11 tells us that if there exists $C = (J, y)$ such that $|J| = j < k$ and $\textsc{Skew}(C) \geq \gamma$, then for any $L \subseteq [n] \setminus J$ of size $k - j$, there exists some further restriction of bits in L such that the resulting subcube $D \subseteq C$ has $\textsc{Skew}(D) \geq \gamma$. This suggests that we ought to ignore subcubes such as D that can be viewed as *inheriting* skew from some parent C, and instead focus on subcubes whose skew is larger than any parent. One technical issue is that we now need to handle the case of positive and negative skew separately. This motivates the following definitions.

▶ **Definition 13.** *Let $\gamma \in (0, 2^k - 1]$ and $\epsilon \in (0, 1]$. A subcube $C \subseteq \{\pm 1\}^n$ is a (γ, ϵ)-minimally skewed subcube if $\textsc{Skew}(C) \geq \gamma$ and for all its parent subcubes $D \supsetneq C$, we have*

$$\textsc{Skew}_\psi(D) \leq (1 - \epsilon)\gamma. \tag{2}$$

Let $\gamma \in (0, 1]$ and $\epsilon \in (0, 1]$. A subcube $C \subseteq \{\pm 1\}^n$ is a $(-\gamma, \epsilon)$-minimally skewed subcube if $\textsc{Skew}(C) \leq -\gamma$ and for all its parent subcubes $D \supsetneq C$, we have

$$\textsc{Skew}_\psi(D) \geq -(1 - \epsilon)\gamma. \tag{3}$$

Note that our convention is to always use $\gamma > 0$ for the magnitude of the skew, and specify its sign explicitly. Note that the allowable values of γ are different for the case of positive and negative skew. We restrict $\epsilon \in (0, 1]$. The case $\epsilon = 1$ corresponds to the case where every subcube of C has no skew.

The crux of this definition is that minimal skew cannot be inherited from a parent. Given a minimal skewed subcube C, and a parent $D \supsetneq C$, we show that C has noticeable skew in the restriction $\psi|_D$.

▶ **Lemma 14.** *If C is a (γ, ϵ)-minimal skewed subcube and $D \supseteq C$ is a parent of C, then*

$$\textsc{Skew}_{\psi|_D}(C) \geq \frac{\epsilon\sqrt{\gamma}}{2}.$$

If C is a $(-\gamma, \epsilon)$-minimal skewed subcube and $D \supseteq C$ is a parent of C, then

$$\textsc{Skew}_{\psi|_D}(C) \leq -\epsilon\gamma.$$

Proof. We first consider the case when $\gamma > 0$. By Lemma 6

$$\langle \psi|_D , \mu_C|_D \rangle = \frac{\langle \psi, \mu_C \rangle}{\langle \psi, \mu_D \rangle} = \frac{1 + \textsc{Skew}_\psi(C)}{1 + \textsc{Skew}_\psi(D)} \geq \frac{1 + \gamma}{1 + (1 - \epsilon)\gamma}. \tag{4}$$

We have

$$\textsc{Skew}_{\psi|_D}(C) = \langle \psi|_D , \mu_C|_D \rangle - 1 \geq \frac{\epsilon\gamma}{1 + (1 - \epsilon)\gamma} \geq \frac{\epsilon\gamma}{2\sqrt{(1 - \epsilon)\gamma}} \geq \frac{\epsilon\sqrt{\gamma}}{2}$$

where the first inequality is by Equation (4) and the second is by the AM-GM inequality.

Next we consider the case where $\gamma < 0$. By Lemma 6

$$\langle \psi|_D , \mu_C|_D \rangle = \frac{\langle \psi, \mu_C \rangle}{\langle \psi, \mu_D \rangle} = \frac{1 + \textsc{Skew}_\psi(C)}{1 + \textsc{Skew}_\psi(D)} \leq \frac{1-\gamma}{1-(1-\epsilon)\gamma}.$$

Hence

$$\textsc{Skew}_{\psi|_D}(C) = \langle \psi|_D , \mu_C|_D \rangle - 1 \leq \frac{1-\gamma}{1-(1-\epsilon)\gamma} - 1 = \frac{\epsilon\gamma}{1-(1-\epsilon)\gamma} \leq -\epsilon\gamma. \quad \blacktriangleleft$$

4 Fourier Analysis

Given $S \subseteq [n]$, let $\chi_S : \{\pm 1\}^n \to \{\pm 1\}$ be given by $\chi_S(x) = \prod_{i \in S} x_i$. These functions form a basis so we can write $\psi = \sum_S \widehat{\psi}(S)\chi_S$, where the Fourier coefficients of ψ are given by

$$\widehat{\psi}(S) = \mathop{\mathbb{E}}_{\mathbf{x}\sim\mu}[\psi(\mathbf{x})\chi_S(\mathbf{x})] = \sum_{x \in \{\pm 1\}^n} \frac{\psi(x)\chi_S(x)}{2^n} = \sum_{x \in \{\pm 1\}^n} \Pr_{\mathbf{x}\sim\psi}[\mathbf{x} = x]\chi_S(x) = \mathop{\mathbb{E}}_{\mathbf{x}\sim\psi}[\chi_S(\mathbf{x})]$$

which is simply the bias of χ_S under the distribution ψ. Thus we have

$$\psi(x) = \sum_{S \subseteq [n]} \widehat{\psi}(S)\chi_S(x)$$

where $\widehat{\psi}(\emptyset) = \mathbb{E}_{\mathbf{x}\sim\psi}[\psi(x)] = 1$. Given two distributions ψ and ω, their inner product is given by

$$\langle \psi, \omega \rangle = \mathop{\mathbb{E}}_{\mathbf{x}\sim\{\pm 1\}^n}[\psi(\mathbf{x})\omega(\mathbf{x})] = \sum_{x \in \{\pm 1\}^n} \frac{\psi(x)\omega(x)}{2^n} = 1 + \sum_{\emptyset \neq S \subseteq [n]} \widehat{\psi}(S)\widehat{\omega}(S)$$

Skew implies heavy low-degree coefficients

We show that large skew in the subcube (K, y) implies non-trivial Fourier mass on subsets of K.

▶ **Lemma 15.** *For $C = (K, y)$,*

$$\textsc{Skew}_\psi(C) = \sum_{\emptyset \neq S \subseteq K} \widehat{\psi}(S)\chi_S(y).$$

Proof. Given $C = (K, y)$, μ_C the uniform measure over C is given by

$$\mu_C(x) = \prod_{i \in K}(1 + x_i y_i) = 1 + \sum_{\emptyset \neq S \subseteq K} \chi_S(y)\chi_S(x).$$

Hence we have

$$\langle \psi, \mu_C \rangle = 1 + \sum_{\emptyset \neq S \subseteq [n]} \widehat{\psi}(S)\widehat{\mu}_A(S) = 1 + \sum_{\emptyset \neq S \subseteq K} \widehat{\psi}(S)\chi_S(y)$$

from which the claim follows. ◀

Given the above lemma, our approach is to reduce bounding the number of skewed subcubes to bounding the number of large Fourier coefficients of ψ at level k.

We define

$$W^{\leq k}(\psi) = \sum_{\substack{S\subseteq[n]\\|S|\leq k}} \widehat{\psi}(S)^2.$$

A trivial bound is obtained from Parseval's identity:

$$W^{\leq k}(\psi) \leq \sum_{S\subseteq[n]} \widehat{\psi}(S)^2 = \mathop{\mathbb{E}}_{\mathbf{x}\sim\{\pm1\}^n}[\psi(\mathbf{x})^2] \leq \|\psi\|_\infty \mathop{\mathbb{E}}_{\mathbf{x}\sim\{\pm1\}^n}[\psi(\mathbf{x})] = \|\psi\|_\infty$$

If we restrict the summation to sets S of cardinality at most k, then a much stronger bound of $O(\ln(\|\psi\|_\infty)^k)$ holds, it is proved using the powerful HyperContractivity Theorem. These bounds generalize the Level-k inequalities for the Fourier spectrum of small-sets, indeed the proof is identical.

For a $f : \{\pm1\}^n \to \mathbb{R}$ and $0 \leq \rho \leq 1$ set

$$T_\rho f = \sum_S \rho^{|S|} \widehat{f}(S)\chi_S$$

T_ρ is known as the noise operator. Recall that for $p > 0$ we have $\|f\|_p = \mathbb{E}[f^p]^{1/p}$. The hypercontractive inequality quantifies the extent to which the noise operater reduces the norm of a function. See for instance. [16, Chapter 2] for a detailed exposition.

▶ **Theorem 16.** *Let $f : \{\pm1\}^n \to \mathbb{R}$ and $\rho \in [0, 1]$. Then*

$$\|T_\rho f\|_2 \leq \|f\|_{1+\rho^2}.$$

We use the hypercontractive inequality to bound the mass of the low level coefficients.

▶ **Theorem 17.** *Let ψ be a distribution. Then*

$$W^{\leq k}(\psi) \leq e^2(\ln(e\,\|\psi\|_\infty))^k.$$

Proof. By Theorem 16, we have

$$\begin{aligned}
\|T_\rho\psi\|_2 &\leq \|\psi\|_{1+\rho^2} \\
&= \left(\mathop{\mathbb{E}}_{\mathbf{x}\sim\{\pm1\}^n}[\psi(x)^{\rho^2}\psi(x)]\right)^{1/(1+\rho^2)} \\
&\leq \|\psi\|_\infty^{\rho^2/(1+\rho^2)} \mathop{\mathbb{E}}_{\mathbf{x}\sim\{\pm1\}^n}[\psi(x)]^{1/(1+\rho^2)} \\
&= \exp\left(\frac{\ln(\|\psi\|_\infty)\rho^2}{1+\rho^2}\right)
\end{aligned}$$

where we used Holder's inequality with $p = \infty$, $q = 1$. Taking $\rho = \min(1, 1/\sqrt{\ln(\|\psi\|_\infty)})$, we have

$$\|T_\rho\psi\|_2 \leq \exp(1/(1+\rho^2)) \leq e$$

But note that

$$\|T_\rho\psi\|_2 \geq (\rho^{2k} W^{\leq k}(\psi))^{1/2}$$

Hence we conclude that

$$W^{\leq k}(\psi)) \leq e^2(1/\rho)^{2k} = e^2 \max(1, \ln\|\psi\|_\infty)^k \leq e^2(\ln(e\,\|\psi\|_\infty))^k.$$ ◀

We also need a bound for the Fourier mass at level k where we do not count coordinates from some set $J \subseteq [n]$ in the degree of a coefficient.

$$W^{\leq k}(\psi, J) = \sum_{T \subseteq J} \sum_{\substack{S \subseteq [n] \setminus J \\ |S| \leq k}} \widehat{\psi}(S \cup T)^2.$$

▶ **Corollary 18.** *For $J \subseteq [n]$ and a distribution ψ over $\{\pm 1\}^n$,*

$$W^{\leq k}(\psi, J) \leq 2^{|J|} e^2 (\ln(e \|\psi\|_\infty))^k.$$

Projection, Extension, Restriction

Given $x \in \{\pm 1\}^n$ and a set of coordinates $P \subseteq [n]$, let x_P denote the projection of x onto coordinates in P. Given a distribution ψ over $\{\pm 1\}^n$ and a set of coordinates $P \subseteq [n]$, let ψ_P denote the marginal distribution over the set P. The Fourier expansion is especially convenient for marginals, we simply restrict the sum to subsets of P.

▶ **Lemma 19.** *For $P \subseteq [n]$ and a distribution ψ over $\{\pm 1\}^n$, the restriction ψ_P is given by*

$$\psi_P(y) = \sum_{S \subseteq P} \widehat{\psi}(S) \chi_S(y).$$

Coversely we can *extend* a distribution ψ' defined on $\{\pm 1\}^P$ for $P \subseteq [n]$ to all of $\{\pm 1\}^n$ while preserving its important properties.

▶ **Lemma 20.** *Let $P \subsetneq [n]$. Let ψ' be a distribution on $\{\pm 1\}^P$. Define a distribution ψ on $\{\pm 1\}^n$ by $\psi(x) = \psi'(x_P)$. Then*

1. *ψ is the product distribution of ψ' with the uniform distribution on $\{\pm 1\}^{\bar{P}}$.*
2. *$\|\psi\|_\infty = \|\psi'\|_\infty$.*
3. *C is a minimal skewed subcube under ψ iff it is a minimal skewed subcube under ψ'.*

Finally, we derive an expression for the Fourier expansion of $\psi|_C$ in terms of the coefficients of ψ.

▶ **Lemma 21.** *Let $C = (J, z)$. Then*

$$\psi|_C(x) = \sum_{S \subseteq [n] \setminus J} \chi_S(x) \frac{\sum_{T \subseteq J} \widehat{\psi}(S \cup T) \chi_T(z)}{\langle \psi, \mu_C \rangle}.$$

5 A Combinatorial Bound for minimal skewed subcubes

In this section, we show bounds on the number of minimal skewed subcubes that is dimension independent.

▶ **Theorem 22** (Combinatorial Bound for Positive Skew). *For any measure ψ on $\{\pm 1\}^n$, integer $k \leq n$, and $\gamma \in (0, 2^k - 1]$ and $\epsilon \in (0, 1]$, the number of (γ, ϵ)-minimal skewed subcubes of codimension at most k is bounded by*

$$k^{O(k)} \left(\frac{1}{\epsilon^2 \gamma} \ln(e \|\psi\|_\infty) \right)^k.$$

▶ **Theorem 23** (Combinatorial Bound for Negative Skew). *For any measure ψ on $\{\pm 1\}^n$, integer $k \leq n$, and $\gamma \in (0,1]$ and $\epsilon \in (0,1]$, the number of $(-\gamma,\epsilon)$-minimal skewed subcubes of codimension at most k is bounded by*

$$k^{O(k)} \left(\frac{1}{\epsilon^2 \gamma^2} \ln \left(\frac{e \, \|\psi\|_\infty}{\epsilon \gamma} \right) \right)^k .$$

We now outline our approach for proving these bounds.

1. We give an algorithm to enumerate all minimal skewed subcubes, given the list of large, low-degree Fourier coefficients in an adaptively chosen sequence of restrictions of the original distribution ψ. The algorithm recursively "grows" skewed subcubes by finding heavy Fourier coefficients and restricting the bits in that coefficient, and showing that this algorithm discovers all minimal skewed subcubes.
2. We bound the number of large low-degree Fourier coefficients of ψ using Theorem 17.

The details of the algorithm in Step 1 are different for the cases of positive and negative skew, so we present them in Subsections 5.1 and 5.2. To go from a combinatorial bound to an efficient algorithm, we need to make Step 2 algorithmic. We will consider this problem under different learning models in Sections 6 and 8.

5.1 Positive skew

We first present an algorithm FindSkew$^+$ for enumerating minimal skewed subcubes where the skew is positive.

To prove the combinatorial bound, we allow the algorithm to make certain *guesses* in Lines 6 and 7. We think of the set of all possible outputs over all possible guesses as the list that is returned by the algorithm. In Lemma 24, we will show that all minimal skewed subcubes are contained in this list. We bound the list size in Lemma 26. Together, these complete the proof of Theorem 22.

We start the recursion with $R_0 = \emptyset$ and $z_0 = \emptyset$ the null string. The routine either returns FAIL or returns $S_t \subset [n]$ and $z_t \in \{\pm 1\}^{S_t}$ such that (R_t, z_t) is a γ-skewed subcube. The algorithm also takes as inputs the input the distribution ψ, a bound k on the codimension, and skew parameters $\gamma \in (0, 2^k - 1]$ and $\epsilon \in (0,1]$. These stay constant through the recursion, so we suppress the dependence on them. Consider the list of all possible choices returned by the algorithm.

Algorithm 1 FindSkew$^+(R_t, y_t)$.

1: Let $D_t = (R_t, y_t)$. Let $\psi_t = \psi|_{D_t}$. Let $k_t = k - |R_t|$.
2: **if** $\text{Skew}_\psi(D_t) > \gamma(1-\epsilon)$ **then**
3: **return** D_t
4: **if** $\langle \psi, \mu_{D_t} \rangle < (1+\gamma) \cdot 2^{-k_t}$ **then**
5: **return** FAIL
6: Pick S_t such that $|S_t| \leq k_t$ and

$$|\widehat{\psi_t}(S_t)| \geq \frac{\epsilon\sqrt{\gamma}}{k_t \cdot \binom{k_t}{|S_t|}}. \tag{5}$$

7: Pick $z_t \in \{\pm 1\}^{S_t}$.
8: **return** FindSkew$^+(R_t \cup S_t, y_t \circ z_t)$.

We need some notation for the analysis. Let the sequence of subcubes produced by the algorithm be $D_0 \supsetneq D_1 \cdots \supsetneq D_\ell$. Let $s_t = |S_t|$.

▶ **Lemma 24.** *For every (γ, ϵ)-minimal skewed subcube C with $\mathrm{codim}(C) \le k$ there are sequences of choices of S_t and z_t (in Lines 6 and 7) so that C is returned by* FindSkew$^+$.

Proof. For every $C = (K^*, z^*)$ that is a (γ, ϵ)-minimal skewed subcube where $\mathrm{codim}(C) \le k$, we will show that for every t, if $D_t \supsetneq C$ is parent of C, and is not equal to C there is a choice of S_t, z_t that leads to a parent D_{t+1} of C with a larger codimension. Since $t \le \mathrm{codim}(D_t) \le \mathrm{codim}(C) \le k$, in $\ell \le k$ steps we must have $D_t = C$, at which point we return at Line 3. Thus the claim implies the lemma.

At $t = 0$, we have $D_0 = \{\pm 1\}^n$ so the parent condition holds trivially. Assume that we have $D_t \supsetneq C$.

By the definition of a minimal skewed subcube, $\textsc{Skew}(D_t) \le (1-\epsilon)\gamma$, hence the procedure will not return at Line 3.

Next we show that $\langle \psi, \mu_{D_t} \rangle \ge (1+\gamma)2^{-k_t}$, the algorithm will not return FAIL at Line 5:

$$\begin{aligned}
\langle \psi, \mu_{D_t} \rangle &= \Pr_{\mathbf{x}\sim\psi}[\mu_{D_t}(\mathbf{x})] = 2^{k-k_t} \Pr_{\mathbf{x}\sim\psi}[\mathbf{x} \in D_t] \\
&\ge 2^{k-k_t} \Pr_{\mathbf{x}\sim\psi}[\mathbf{x} \in C] \\
&\ge 2^{-k_t} \langle \psi, \mu_{C_t} \rangle \\
&\ge (1+\gamma)2^{-k_t}.
\end{aligned}$$

The first inequality holds because $D_t \supset C$, the second because $\mathrm{codim}(C) \le k$ and the last because we assume that $\textsc{Skew}(C) \ge \gamma$.

Recall that $\psi_t = \psi|_{D_t}$, and let $K_t = K \setminus S_t$. By Lemma 15,

$$\textsc{Skew}_{\psi_t}(C) = \sum_{\emptyset \ne S \subseteq K_t} \widehat{\psi_t}(S)\chi_S(y) = \sum_{k'} \sum_{\substack{\emptyset \ne S \subseteq K_t \\ |S|=k'}} \widehat{\psi_t}(S)\chi_S(y)$$

By Lemma 14, $\textsc{Skew}_{\psi_t}(C) \ge \epsilon\sqrt{\gamma}/2$ which implies that for some $k' \le k_t$, we have

$$\sum_{\substack{\emptyset \ne S \subseteq K_t \\ |S|=k'}} \widehat{\psi_t}(S)\chi_S(y) \ge \epsilon\sqrt{\gamma}/k_t$$

which in turn implies that for at least one $\emptyset \ne S_t \subseteq K_t$, we have

$$|\widehat{\psi_t}(S_t)| \ge \epsilon\sqrt{\gamma} / \left(k_t \cdot \binom{k_t}{k'}\right).$$

Assume that we pick this S_t in Line 6 and $z_t = z^*|_S$ in Line 7. This ensures that D_{t+1} is a parent of C of larger codimension. ◀

We next bound the number of all possible outputs of the algorithm. The crux of the argument is to bound the number of large low-degree Fourier coefficients using Theorem 17. This in turn requires a bound on the infinity norm of ψ_t which comes from passing the test in Line 4.

▶ **Lemma 25.** *The number of choices for S_t satisfying Equation* (5) *is bounded by*

$$\frac{e^2}{\epsilon^2\gamma}(\ln(2^{k_t} e \|\psi\|_\infty))^{s_t} k_t^{4s_t+2}.$$

Proof. We bound $\|\psi_t\|_\infty$ as

$$\|\psi_t\|_\infty = \left\|\psi|_{D_t}\right\|_\infty \leq \frac{\|\psi\|_\infty}{\langle\psi, \mu_{D_t}\rangle} \leq \frac{\|\psi\|_\infty}{(1+\gamma)2^{-k_t}} \leq \|\psi\|_\infty 2^{k_t}$$

where the first inequality is from Fact 5 and we have $\langle\psi, \mu_{D_t}\rangle \geq (1+\gamma)2^{-k_t}$ since we check for this condition in Line 5. We now use Theorem 17 which gives

$$W^{\leq s_t}(\psi) \leq e^2(\ln(e\|\psi\|_\infty \cdot 2^{k_t}))^{s_t}.$$

Hence the number of choices for S_t satisfying (5) is bounded by

$$W^{\leq s_t}(\psi_t)\left(\frac{k_t\binom{k_t}{s_t}}{\epsilon\sqrt{\gamma}}\right)^2 \leq \frac{e^2}{\epsilon^2\gamma}(\ln(2^{k_t}e\|\psi\|_\infty))^{s_t}k_t^{4s_t+2}.$$ ◀

▶ **Lemma 26.** *The total number of subcubes of codimension k output by* FindSkew$^+$ *is at most:*

$$k^{O(k)}\left(\frac{\ln(e\|\psi\|_\infty)}{\epsilon^2\gamma}\right)^k.$$

Proof. Since $\sum_{t\leq\ell} s_t = k$, the sequence $\{s_t\}_{t=1}^{\ell}$ is a partition of k, and there are at most k^k of them. Let us fix the sequence. The number of choices for S_t is bounded by Lemma 25. Since $z_t \in \{\pm 1\}^{S_t}$, the number of choices for z is 2^{s_t}. Taking the product over all t, the number of possible outputs for FindSkew$^+$ is bounded by

$$\prod_{t=1}^{\ell}\frac{e^2}{\epsilon^2\gamma}(\ln(2^{k_t}e\|\psi\|_\infty))^{s_t}k_t^{4s_t+2}\cdot 2^{s_t}$$

We can bound

$$\prod_{t=1}^{\ell}(\ln(2^{k_t}e\|\psi\|_\infty))^{s_t} \leq \ln(2^k e\|\psi\|_\infty)^k \leq (k+\ln(e\|\psi\|_\infty))^k \leq (2k)^k(\ln(e\|\psi\|_\infty))^k.$$

$$\prod_{t=1}^{\ell}k_t^{4s_t+2}2^{s_t} \leq k^{5k+2}.$$

Including the k^k choices for $s_1, \ldots, s_t$, the output list size is bounded by

$$\left(\frac{e^2}{\epsilon^2\gamma}\right)^k(\ln(e\|\psi\|_\infty))^k k^{7k+2} = k^{O(k)}\left(\frac{\ln(e\|\psi\|_\infty)}{\epsilon^2\gamma}\right)^k.$$ ◀

Together Lemma 24 and Lemma 26 complete the proof of Theorem 22.

5.2 Negative Skew

We now present an algorithm FindSkew$^-$ for the negative skewed case. The algorithm takes as input $\gamma \in (0,1]$ and $[\epsilon \in (0,1]$ and the goal is to list all $(-\gamma,\epsilon)$-minimal negatively skewed subcubes.

The main differences from FindSkew$^+$ are that once the skew is less than $-\gamma(1-\epsilon)$, we can return. Thus we can combine the Return statement (Line 3, and the the check in Line 5. Also, the bound on the coefficient size in Equation(6) now reflects the bound for the negative skew case in Lemma 14.

We have the following claim about the correctness of FindSkew$^-$.

Algorithm 2 FindSkew$^-(R_t, y_t)$.

1: Let $D_t = (S_t, z_t)$. Let $\psi_t = \psi|_{D_t}$. Let $k_t = k - |S_t|$.
2: **if** $\mathrm{Skew}_\psi(D_t) < -\gamma(1-\epsilon)$ **then**
3: **return** D_t
4: Pick S_t such that $|S_t| \le k_t$ and

$$|\widehat{\psi_t}(S_t)| \ge \frac{\epsilon\gamma}{k_t \cdot \binom{k_t}{|S_t|}}. \tag{6}$$

5: Pick $z_t \in \{\pm 1\}^{S_t}$.
6: **return** FindSkew$^-(R_t \cup S_t, y_t \circ z_t)$.

▶ **Lemma 27.** *For every (γ,ϵ)-minimal skewed subcube C with $\mathrm{codim}(C) \le k$ there are choices of subsets S_t and z_t (in Lines 6 and 7) so that C is returned by* FindSkew$^-$.

We prove this by showing that for every t, if $D_t \supsetneq C$ is parent of C, and is not equal to C there is a choice of S_t, z_t that gives a parent D_{t+1} of C with a larger codimension. Indeed, we know that for any parent of C, inner product $\langle \psi, \mu_{D_t}\rangle$ is large enough to pass the test in Line 3. The rest of the proof is identical to that of Lemma 24 for the case of positive skew, so we do not repeat it.

The crux of the proof is to bound the number of choices for S_t satisfying Equation (6).

▶ **Lemma 28.** *The number of choices for S_t satisfying Equation* (6) *is bounded by*

$$\frac{e^2}{\epsilon^2\gamma^2}\left(\ln\left(\frac{e\,\|\psi\|_\infty}{\epsilon\gamma}\right)\right)^{s_t} k^{2s_t+2}.$$

Proof. To pass Line 3, it must hold that $\mathrm{Skew}_\psi(D_t) \ge -\gamma(1-\epsilon)$, hence

$$\langle \psi, \mu_{D_t}\rangle \ge 1 - \gamma(1-\epsilon) \ge \epsilon\gamma$$

since $\gamma \le 1$. So we bound $\|\psi_t\|_\infty$ as

$$\|\psi_t\|_\infty = \left\|\psi|_{D_t}\right\|_\infty = \frac{\|\psi\|_\infty}{\langle \psi, \mu_{D_t}\rangle} \le \frac{\|\psi\|_\infty}{\gamma\epsilon}.$$

Using Theorem 17 gives

$$W^{\le s_t}(\psi_t) \le e^2\left(\ln\left(\frac{e\,\|\psi\|_\infty}{\epsilon\gamma}\right)\right)^{s_t}.$$

Hence the number of choices for S_t satisfying Equation (6) is bounded by

$$\begin{aligned}\frac{W^{\le s_t}\cdot\left(k_t\cdot\binom{k_t}{|S_t|}\right)^2}{\epsilon^2\gamma^2} &\le e^2\left(\ln\left(\frac{e\,\|\psi\|_\infty}{\epsilon\gamma}\right)\right)^{s_t}\frac{\left(k_t\cdot\binom{k_t}{|S_t|}\right)^2}{\epsilon^2\gamma^2}\\ &\le \frac{e^2}{\epsilon^2\gamma^2}\left(\ln\left(\frac{e\,\|\psi\|_\infty}{\epsilon\gamma}\right)\right)^{s_t} k^{2s_t+2}.\end{aligned}$$

◀

We can now conclude as before.

▶ **Lemma 29.** *The total number of subcubes of codimension k output by* FindSkew$^-$ *is bounded by*

$$k^{O(k)}\left(\frac{1}{\epsilon^2\gamma^2}\ln\left(\frac{e\,\|\psi\|_\infty}{\epsilon\gamma}\right)\right)^k.$$

Proof. Using Lemma 28 and the fact that there are are 2^{s_t} choices for z_t and k^k choices of the partition $s_1, \ldots, s_t$, the overall list size is bounded by

$$k^k \prod_{t=1}^{\ell} \frac{e^2}{\epsilon^2\gamma^2} \left(\ln\left(\frac{e\,\|\psi\|_\infty}{\epsilon\gamma}\right)\right)^{s_t} k^{2s_t+2}2^{s_t} \leq k^{O(k)} \left(\frac{1}{\epsilon^2\gamma^2} \ln\left(\frac{e\,\|\psi\|_\infty}{\epsilon\gamma}\right)\right)^k. \quad \blacktriangleleft$$

Combining Lemmas 27 and 29 completes the proof of Theorem 23.

5.3 Tightness of our bounds

We show that the dependence on $\|\psi\|_\infty$ in Theorem 22 is nearly optimal. To simplify our constructions, we will construct distributions on n variables where $n = n(\|\psi\|_\infty, k)$. But one can then use Lemma 20 to extend the construction to all larger values of n.

▶ **Theorem 30.** *There exists a distribution μ_C on $\{\pm1\}^n$ which has $\Omega_k((\ln(\|\mu_C\|_\infty))^k)$ many $(2^k, 1/2)$-minimal skewed subcubes of codimension k.*

Proof. Let C be the subcube where all the bits are fixed to 1, and let μ_C be the uniform distribution over it. It follows that $\|\mu_C\|_\infty = 2^n$ hence $\ln(\|\mu_C\|_\infty) = n$. We claim that all $\binom{n}{k}$ subcubes where a k out of the first t bits are fixed to 1 are $(2^k - 1, 1/2)$-minimal skewed subcubes. Fix one such cube D. We have

$$\textsc{Skew}_{\mu_C}(D) = 2^k \Pr_{\mathbf{x}\sim\mu_C}[\mathbf{x} \in D] - 1 = 2^k - 1.$$

Since the maximum skew of any subcube of codimension $k-1$ is at most $2^{k-1} - 1$, D satisfies the definition of (γ, ϵ)-minimal skew for $\gamma = 2^k - 1$ and ϵ such that $\gamma(1-\epsilon) \geq 2^{k-1} - 1$. In particular, we can take $\epsilon = 1/2$.

Thus the number of $(2^k - 1, 1/2)$-minimal skewed subcubes is $\binom{n}{k} = \Omega_k(n^k)$. The only dependence on n in Theorem 22 comes from the $\|\mu_C\|_\infty$ since ϵ is a constant and $\gamma \leq 2^k$. Thus the number of cubes is $\Omega((\ln(\|\mu_C\|_\infty))^k)$. $\blacktriangleleft$

For Theorem 23 dealing with negative skew, we show a similar bound, though with a smaller value of $\epsilon = 1/k$. The distribution we use is derived from the Tribes function.

▶ **Theorem 31.** *There exists a distribution τ on $\{\pm1\}^n$ which has $\Omega_k((\ln(\|\tau\|_\infty))^k)$ many $(-1, 1/k)$-minimal skewed subcubes of codimension k.*

Proof. Let $n = tk$. We label the coordinates as $\{x_{i,j}\}_{i\in[k],j\in[t]}$. Consider the DNF formula

$$\text{Tribes}(x) = \bigvee_{i=1}^{k} \bigwedge_{j=1}^{t} x_{i,j}$$

We now define a distribution τ where we pick a $i^* \in [k]$ at random, set $x_{i^*,j} = 1$ for all $j \in [t]$ and set all the other variables randomly. Clearly the distribution τ is supported on the satisfying assignments of Tribes(x). It is also easy to see that

$$\|\tau\|_\infty = \tau(1^{tk}) = 2^{tk} \sum_{i=1}^{k} \frac{1}{k} 2^{-(k-1)t} = 2^t.$$

Now consider the set of minimal 0 certificates of Tribes, which are subcubes where we pick a single variables from each term and set it to 0. There are t^k such subcubes, fix one such subcube C. Clearly $\Pr_{\mathbf{x}\sim\tau}[\mathbf{x} \in C] = 0$, hence $\textsc{Skew}_\tau(C) = -1$. Now consider any parent subcube D of C. Assume it has codimension $\ell < k$, and let $L \subset [k]$ denote the set of terms that it sets to 0. For $\mathbf{x} \sim \tau$ to lie in L, two events need to happen:

- $i^* \notin L$, which happens with probability $1 - \ell/k$.
- The variables in L which are set to 0 in D are also set to 0 by the random assignment, which happens with probability $2^{-\ell}$.

As these two events are independent, we have $\Pr_{\mathbf{x}\sim\tau}[\mathbf{x} \in D] = 2^{-\ell}(1 - \ell/k)$ hence

$$\textsc{Skew}_\tau(D) = 2^\ell \Pr_{\mathbf{x}\sim\tau}[\mathbf{x} \in D] - 1 = -\frac{\ell}{k}.$$

Thus the maximum skew of any parent D is $-1 + 1/k$. Hence C is $(-1, 1/k)$-minimally skewed.

As before we note that $\gamma = 1$ and $\epsilon = 1/k$, hence the only dependence on t comes from $\log(\|\tau\|_\infty) = t$, which gives the claimed bound. ◀

Finally, we present a distribution that has a large number of $(-1, 1)$ and $(1, 1)$ minimally skewed subcubes. Recall that $\epsilon = 1$ means that every parent of the cube has skew 0.

The construction is based on (dual) BCH codes. We think of linear codes as subsets of $\mathbb{F}_2^n$ where $\mathbb{F}_2 = \{0, 1\}$ which we can identify with $\{\pm 1\}^n$ via the usual mapping $x \to (-1)^x$. For $x \in \mathbb{F}_2^n$ let the weight of x denoted $\mathrm{wt}(x)$ be the number of 1s in x. Let $\mathrm{supp}(x) \subseteq [n]$ denote the set of cordinates where x is non-zero. We will use the following fact about BCH codes communicated to us by Sergey Yekhanin [19].

▶ **Lemma 32** ([19]). *Let $d \geq 2$ be even and let $n + 1 = 2^l \geq d$. There exists a $\mathbb{F}_2$-linear code $\mathcal{C}_{BCH} \subseteq \{0, 1\}^n$ with minimum distance d, which contains $\Omega(n^{d/2+1})$ minimum weight codewords.*

▶ **Theorem 33.** *For any even $k \geq 2$ and large enough n, there exists a distribution ψ_k on $\{\pm 1\}^n$ where the numbers of $(-1, 1)$-minimal skewed subcubes and $(1, 1)$-minimal skewed subcubes of codimension k are both $\Omega_k((\log(\|\psi_k\|_\infty))^{k/2+1})$.*

Proof. Set $k = d$, and take n as in Lemma 32. Let ψ^k be the uniform distribution on the dual space to $\mathcal{C}_{BCH}$. Using standard facts about the Fourier expansion of a subspace, we can write

$$\psi^k = \sum_{\substack{c \in \mathcal{C}_{BCH} \\ S = \mathrm{supp}(c)}} \chi_S(x). \tag{7}$$

Since $\mathcal{C}_{BCH}$ has minimum distance k, ψ^k is $(k-1)$-wise independent, so for any subcube C where $\mathrm{codim}(C) \leq k - 1$, we have $\textsc{Skew}_{\psi^k}(C) = 0$. This relies on a standard construction of k-wise independent spaces from codes [2, Chapter 16], it can also be seen using Lemma 15 combined with Equation (7).

Fix $S \subset [n]$ to be the support of a minimum weight codeword in $\mathcal{C}_{BCH}$. By Lemma 19 and Equation (7), the projection of ψ^k to coordinates in S is given by $\psi^k_S(x) = 1 + \chi_S(x)$. Hence it is uniform over the 2^{k-1} settings $y \in \{\pm 1\}^S$ such that $\chi_S(y^+) = 1$.

For every such y and $D^+ = (S, y)$, we have $\Pr_{\mathbf{x}\sim\psi^k}[\mathbf{x} \in D^+] = 2^{-(k-1)}$ hence

$$\textsc{Skew}_{\psi^k}(D^+) = 2^k \Pr_{\mathbf{x}\sim\psi^k}[\mathbf{x} \in D^+] - 1 = 1.$$

On the other hand, for every $y \in \{\pm 1\}^S$ such that $\chi_S(y) = -1$ and $D^- = (S, y)$, we have $\Pr_{\mathbf{x}\sim\psi^k}[\mathbf{x} \in D^-] = 0$ hence

$$\textsc{Skew}_{\psi^k}(D^-) = 2^k \Pr_{\mathbf{x}\sim\psi^k}[\mathbf{x} \in D^-] - 1 = -1.$$

Since every parent of D^+ has 0 skew, every such D^+ is a $(1, 1)$-minimal skewed subcube, and similary for every D^-.

Trivially, we have $\|\psi^k\|_\infty \le 2^n$, hence $\log(\|\psi^k\|_\infty) \le n$ (in fact it equals $n - O(\log(n))$. Since the number of minimal weight codewords is $\Omega_k(n^{k/2+1})$ by Lemma 32, and γ, ϵ are both 1, the number of codewords is $\Omega_k((\log(\|\psi^k\|_\infty))^{k/2+1})$. Hence the number of minimal skewed subcubes is as claimed. ◀

6 Algorithms for Finding Skewed Subcubes

In this section, we present an algorithm that find skewed subcubes efficiently in the random sample model, where we have access to random samples from ψ.

To make Algorithm 1 efficient, we need to replace the step of guessing S (Line 6 in Algorithm 1, and Line 4 in Algorithm 2) with an algorithm to find large low degree Fourier coefficients[4]. We restate the problem below:

▶ **Problem 3** (Finding large low-degree biases). *Given a distribution ψ on $\{\pm 1\}^n$, an integer k and $\rho \ge 0$, find all $S \subseteq [n]$ such that $|S| \le k$ and*

$$\hat{\psi}(S) := \mathop{\mathbb{E}}_{\mathbf{x}\sim\psi}[\chi_S(x)] \ge \rho.$$

Our main result is the following pair of theorems.

▶ **Theorem 34** (Algorithm for Positive Skew). *Given sample access to a distribution ψ on $\{\pm 1\}^n$, integer $k \le n$, and parameters $\gamma \in (0, 2^k - 1]$, $\epsilon \in (0, 1]$ and $\lambda \in [0, 1]$, there is an algorithm that returns all (γ, ϵ)-minimal skewed subcubes of codimension at most k in time:*

$$\tilde{O}\left(n^{k\left(\frac{\omega}{3-\lambda}\right)}\right) + k^{O(k)} \cdot (\ln(e\,\|\psi\|_\infty))^k \left(\frac{\tilde{O}(n^{k/3})}{(\epsilon\sqrt{\gamma})^{4/\lambda}} + \frac{\mathrm{poly}(n)}{(\epsilon\sqrt{\gamma})^{2k}}\right)$$

where ω is the matrix multiplication exponent, and $\tilde{O}$ hides $\mathrm{poly}\log n$ factors.

▶ **Theorem 35** (Algorithm for Negative Skew). *Given sample access to a distribution ψ on $\{\pm 1\}^n$, integer $k \le n$, and parameters $\gamma \in (0, 1]$, $\epsilon \in (0, 1]$ and $\lambda \in [0, 1]$, there is an algorithm that returns all $(-\gamma, \epsilon)$-minimal skewed subcubes of codimension at most k in time:*

$$\tilde{O}\left(n^{k\left(\frac{\omega}{3-\lambda}\right)}\right) + k^{O(k)} \cdot (\ln(e\,\|\psi\|_\infty))^k \left(\frac{\tilde{O}(n^{k/3})}{(\epsilon\gamma)^{4/\lambda}} + \frac{\mathrm{poly}(n)}{(\epsilon\gamma)^{2k}}\right)$$

where ω is the matrix multiplication exponent, and $\tilde{O}$ hides $\mathrm{poly}\log n$ factors.

Theorem 3 follows from using (a), setting $\lambda = 0.01$ and $\omega \le 2.38$.

In both algorithms, we will find large low-degree biases using a breakthrough algorithm of [18] for detecting pairs of vectors that are highly correlated from a set of weakly correlated vectors. The algorithm was subsequently improved by [13]).

▶ **Theorem 36** ([13]). *Given two sets of vectors $V_1, V_2 \subseteq \{\pm 1\}^n$ for which there are at most q pairs $(v_1, v_2) \in V_1 \times V_2$ with correlation larger than τ, and a parameter $\rho \ge \tau^{1/\lambda}$ (for $\lambda \in [0, 1]$), there is an algorithm $\textsc{FindCorr}(V_1, V_2, \rho, \tau)$ that with high probability outputs all pairs $(v_1, v_2) \in V_1 \times V_2$ with correlation at least ρ. Furthermore, algorithm runs in time*

$$\tilde{O}\left(n^{\frac{2\omega}{3-\lambda}} + qdn^{\frac{2(1-\lambda)}{3-\lambda}}\right).$$

[4] We note that technically to implement the algorithm we also need to estimate $\langle \psi, \mu_C \rangle$ for every C to an additive accuracy of $\min(\gamma, 2^{-k})$. If done via sampling, these only incur $2^{2k} k \log n/\gamma^2$ additional cost per call, and will be absorbed into our runtime bounds anyway.

The essence of the reduction is as follows. For each set $S \subseteq [n]$ less than $k/2$ we associate a vector y_S, for which each coordinate is a random sample of $\chi_S(x)$ where x is drawn from ψ. If $Q, R \subseteq [n]$ are disjoint, the correlation coefficient $\mathbb{E}[y_Q \cdot y_R]/d$ is precisely the value of the Fourier coefficient $\widehat{\psi}(Q \cup R)$. Thus as long as the algorithm of [13] succeeds and is not overwhelmed by sample error, every ρ correlated pair (y_Q, y_R) corresponds to a Fourier coefficient $\widehat{\psi}(Q \cup R)$ of size less than k and absolute value ρ. We now describe this more formally.

Algorithm 3 FindFourierCoefficients(ψ, k, ρ, λ).

$\mathcal{S} \leftarrow \emptyset$
$\tau \leftarrow (\rho/2)^{1/\lambda}$
Draw a set of $d = O(k \log n/\tau^2)$ samples $x_1, \ldots x_d$ from ψ.
for $T = O(k^{3/2} \log n)$ rounds **do**
  Randomly partition $[n]$ into two subsets N_1 and N_2.
  For every subset $S \subseteq N_1$ of size $\leq \lceil k/2 \rceil$, form a vector $y_S \in [-1, 1]^d$ for which the ith bit is set to $\chi_S(x_i)$. Call this set of vectors V_1.
  Do the same for N_2 for sets of size $\leq \lfloor k/2 \rfloor$, and call the set of vectors V_2.
  Run FindCorr$(V_1, V_2, \rho/2, \tau)$ from [13] to find all pairs y_Q and y_R such that $Q \subseteq N_1$, $R \subseteq N_2$, and y_Q and y_R are $\rho/2$ correlated. For each of these, add $Q \cup R$ to $\mathcal{S}$.
return $\mathcal{S}$.

We first prove some simple lemmas using standard concentration of measure results.

▶ **Lemma 37.** *For every $Q \in N_1$ and $R \in N_2$, w.h.p. $\left|\langle y_Q, y_R \rangle / d - \widehat{\psi}(Q \cup R)\right| \leq \tau/2$.*

Proof. For a single $x \sim \{\pm 1\}^n$ and disjoint $Q, R \subseteq N$, we have $\mathbb{E}[\chi_Q(x) \cdot \chi_R(x)] = \widehat{\psi}(Q \cup R)$. Applying the Hoeffding bound with our choice of $d = 32k \log n/\tau^2$, we have that $\langle y_Q, y_R \rangle / d$ is within $\tau/2$ of $\widehat{\psi}(Q \cup R)$ with probability at least $1 - n^{-2k}$. By a union bound, this hold for all $O(n^k)$ choices of pairs (Q, R) with probability $1 - n^{-k}$. ◀

▶ **Lemma 38.** *For every $S \subseteq [n]$ with $|S| \leq k$, w.h.p. in at least one round $t \in [T]$ there are $Q \subseteq N_1$, $R \subseteq N_2$ with $Q \cap R = \emptyset$ such that $Q \cup R = S$.*

Proof. Fix a set S of size ℓ. For a random bipartition of $[n]$, the probability S is perfectly bisected is at least $1/(8\sqrt{\ell}) \geq 1/(8\sqrt{k})$. The probability it is never bisected over $T = 16k^{3/2} \log n$ rounds is upper bounded by

$$\left(1 - \frac{1}{8\sqrt{k}}\right)^T \leq e^{-2k \log n} = n^{-2k}.$$

By a union bound, every S of size $\leq k$ is bisected at least once with high probability. ◀

▶ **Lemma 39.** *The algorithm* FindFourierCoefficients(ψ, k, ρ, λ) *returns all Fourier coefficients of ψ of degree at most k of absolute value at least ρ in time*

$$\tilde{O}\left(n^{k\omega/(3-\lambda)}\right) + \tilde{O}(n^{k/3}) 2^{O(k)} (\ln(e \|\psi\|_\infty))^k \rho^{-4/\lambda}.$$

Proof. Consider any set $S \subseteq [n]$ of size $\leq k$ of magnitude at least ρ. By Lemma 38, w.h.p. for some round $t \in [T]$, the algorithm will form two vectors y_Q and y_R such that $Q \subseteq N_1$, $R \subseteq N_2$ and $Q \cup R = S$. Furthermore, by Lemma 37, we have $\langle y_Q, y_R \rangle / d \geq \rho - \tau \geq \rho/2$. In turn, this means that FindCorr$(V_1, V_2, \rho/2, \tau)$ will detect these w.h.p.

To bound the running time, we need a bound on the number q of pairs with correlation higher than τ. By Lemma 37, we have $\langle y_Q, y_R \rangle / d \geq \tau$ implies $\hat{\psi}(Q \cup R) \geq \tau/2$. The number of such coefficients is bounded by

$$\frac{W^{\leq k}(\psi)}{(\tau/2)^2} \leq 2^8 \left(\ln(e \|\psi\|_\infty)\right)^k \rho^{-2/\lambda}.$$

For each such coefficient S, there are 2^k ways to write it as $S = Q \cup R$ for disjoint Q, R. Hence

$$q \leq 2^{k+8} (\ln(e \|\psi\|_\infty))^k \rho^{-2/\lambda}.$$

Observe that $\log_\tau \rho \geq \lambda$ by our choice of τ. By Theorem 36, [13] will find a list containing all $\rho/2$ correlated pairs in time at most

$$\begin{aligned}
&\tilde{O}\left(\left(n^{k/2}\right)^{2\omega/(3-\lambda)} + qd\left(n^{k/2}\right)^{(2-2\lambda)/(3-\lambda)}\right) \\
&\leq \tilde{O}\left(n^{k\omega/(3-\lambda)} + 2^{O(k)}(\ln(e\|\psi\|_\infty))^k \rho^{-2/\lambda} k \cdot \log(n) \cdot \rho^{-2/\lambda} n^{k(1-\lambda)/(3-\lambda)}\right) \\
&\leq \tilde{O}\left(n^{k\omega/(3-\lambda)}\right) + \tilde{O}(n^{k/3}) 2^{O(k)} (\ln(e\|\psi\|_\infty))^k \rho^{-4/\lambda}.
\end{aligned}$$

◀

We relegate the remainder of the proofs of Theorem 34 to the appendix.

7 Reduction from Noisy Parity

Recall that given $S \subseteq [n]$, a parity function $\chi_S : \{\pm 1\}^n \to \{\pm 1\}$ is given by $\chi_S(x) = \prod_{i \in S} x_i$. A noisy parity is a parity function with random noise of rate η added to it. In other words, we say $f : \{\pm 1\}^n \to \{\pm 1\}$ is an η-noisy parity if $\Pr[f(x) = \chi_S(x)] = 1 - \eta$ an $\Pr[f(x) = -\chi_S(x)] = \eta$. In the sparse noisy parity problem, we are given access to samples $(\mathbf{x}, f(\mathbf{x}))$ where $\mathbf{x} \sim \mu$ is sampled uniformly from $\{\pm 1\}^n$ and f is noisy parity χ_S with $|S| = k$. The goal is to recover the parity function, or equivalently the set S.

Given a set $S' \subseteq [n]$, we have

$$\mathop{\mathbb{E}}_{\mathbf{x} \sim \mu}[\widehat{f}(S')] = \begin{cases} 0 & \text{if } S' \neq S \\ 1 - 2\eta & \text{if } S' = S \end{cases}$$

This leads to a naive enumeration algorithm that runs in time $O(n^k)$. The current best algorithm due to [18] runs in time $O(n^{0.8k})\,\mathrm{poly}(1/(1-2\eta)))$. A series of reductions due to [8] show that efficient algorithms for sparse noisy parity imply algorithms with similar running times for learning k-juntas, decision trees and DNFs under the uniform distribution. This suggests the following conjecture (which we consider to be folklore).

▶ **Conjecture.** *There is no algorithm for the sparse noisy parity problem which runs in time* $n^{o(k)}$.

We now prove Theorem 2 which we restate below.

▶ **Theorem 2.** *For $\eta \in (0, 1/2)$, an algorithm that given a distribution ψ and k can find a $(1-2\eta, 1)$-minimal skewed subcube of co-dimension k in time $T(n, k, \eta)$ can be used to solve the k-sparse noisy parity problem with noise rate η in time $T(n, k, \eta)$.*

Proof. Finding a noisy parity reduces to finding a minimal skewed subcube. Given an instance of sparse noisy parity, consider the distribution ψ on $\{\pm 1\}^{n+1}$ obtained by appending the label to the sample. Thus $\psi = (\mathbf{x}, f(\mathbf{x}))_{\mathbf{x}\sim\mu}$. We show that all skewed subcubes must restrict the set $S \cup \{n+1\}$, hence they are all minimal, and finding any skewed subcube solves the noisy parity problem.

For any $z \in \{\pm 1\}^S$, consider the subcube $D^+(z)$ given by $x_S = z$, $x_{n+1} = \chi_S(z)$. We have

$$\textsc{Skew}(D^+(z)) = 2^{k+1}\left(\Pr_{\mathbf{x}'\sim\psi}[\mathbf{x}' \in D] - \Pr_{\mathbf{x}'\sim\mu}[\mathbf{x}' \in D]\right) = 2^{k+1}\left(\frac{1-\eta}{2^k} - \frac{1}{2^{k+1}}\right) = 1 - 2\eta.$$

Similarly if we define $D^-(z)$ by $x_S = z$ and $x_{n+1} = -\chi_S(z)$, then

$$\textsc{Skew}(D^-(z)) = 2^{k+1}\left(\Pr_{\mathbf{x}'\sim\psi}[\mathbf{x}' \in D] - \Pr_{\mathbf{x}'\sim\mu}[\mathbf{x}' \in D]\right) = 2^{k+1}\left(\frac{\eta}{2^k} - \frac{1}{2^{k+1}}\right) = -(1-2\eta).$$

It is easy to verify that if the set of restricted variables is not $S \cup \{n+1\}$, then the skew is 0, which shows that the subcubes above all $(\pm(1-2\eta), 1)$-minimal skewed subcubes. ◀

This shows the hardness of finding subcubes with skew $(1-2\eta) < 1$. The reduction could be extended to show the hardness of finding subcubes with larger skew, simply by concatenating ℓ different samples of ψ. Now an algorithm that finds a subcube of skew $(2(1-\eta))^\ell - 1$ and codimension k can be used to solve a k/ℓ-sparse noisy parity problem.

8 The Membership Query Model

Theorem 2 suggests that a much better algorithm does not exist in the model where we only get random samples from ψ. However, noisy parity becomes trivial when we are given query access to f, by repeatedly querying the function at x and $x \cdot e_i$. This motivates us to consider the query model where in addition to getting random samples from ψ, we are allowed to query $\psi(x)$ for points x of our choosing. As we will see, this does make finding skewed subcubes easier for distributions where $\|\psi\|_\infty$ is small. We first show how this improvement arises, and then give evidence that queries do not add too much power over random samples when $\|\psi\|_\infty$ is large.

Algorithmically, all we need is a procedure to find all large low-degree Fourier coefficients of ψ under the query model. Such a procedure is given by a classic result [14] of Kushilevitz and Mansour, which uses the algorithm of Goldreich and Levin [9] to compute large Fourier coefficients when given a query access to a function.

▶ **Theorem 40** ([9]). *Given query access to $f : \{\pm 1\}^n \to [-t, t]$ and a parameter $\rho > 0$, there is an algorithm running in time poly$(n, t/\rho)$ that with high probability outputs a list containing all subsets S, such that $\widehat{f}(S) \geq \tau$.* [5]

If we apply it to ψ, then we get an algorithm whose running time is poly$(n, \|\psi\|_\infty, \rho)$. Thus, the algorithm is faster than the trivial exhaustive search algorithm only when $\|\psi\|_\infty < n^{\alpha k}$ for some $\alpha > 0$. The polynomial dependence on $\|\psi\|_\infty$ in the running time is inevitable since the algorithm finds all $\widehat{\psi}(S) \geq \rho$, and not just those with $|S| \leq k$. The number of such coefficients can be $\|\psi\|_\infty / \rho^2$. In contrast, when we restrict to $|S| \leq k$, the list-size only grows as $\ln(e\|\psi\|_\infty)^k/\rho^2$. This raises the following natural open question:

[5] The theorem is typically stated for functions with range $\{\pm 1\}$, however a similar bound is true for the range $[-1, 1]$. The version stated here follows by scaling the function by t so it lies in the range $[-1, 1]$, and replacing ρ by ρ/t.

▶ **Problem 4.** *Given query access to a probability measure, ψ such parameter $\rho > 0$, does there exist an algorithm that can find all S such that $|S| \leq k$ and $|\widehat{\psi}(S)| \geq \rho$ in time* $\mathrm{poly}(n, 1/\rho, \ln(\|\psi\|_\infty)^k)$*?*

We conclude by observing that some dependence on $\|\psi\|_\infty$ (at least logarithmic) seems inevitable, even in cases where the list-size is 1. This is seen by a reduction from sparse noisy parity. A sample of size $O(k \log n/\epsilon^2)$ from an instance of noisy parity preserves all correlations of sets of k variables up to an additive ϵ. Define ψ to be the uniform measure of these samples alone. Finding S such that $|S| \leq k+1$ and $\widehat{\psi}(S) \geq 1 - 2\eta$ will solve the noisy parity problem. Note that for ψ the query model and the random samples model are equivalent as we have the support explicitly. So if we believe that sparse noisy parity requires time $n^{\Omega(k)}$ time, then any algorithm for finding large low-degree Fourier coefficients in ψ must require as much time. Since $\|\psi\|_\infty = \epsilon^2 2^n/k \log(n)$, we have $\ln(\|\psi\|_\infty) = n - O(\log(1/\epsilon)$. This is consistent with a dependence of $\ln(\|\psi\|_\infty)^{\Omega(k)}$.

References

1. Noga Alon, Alexandr Andoni, Tali Kaufman, Kevin Matulef, Ronitt Rubinfeld, and Ning Xie. Testing k-wise and almost k-wise independence. In *STOC*, 2007.
2. Noga Alon and Joel H. Spencer. *The Probabilistic Method*. Wiley Publishing, 4th edition, 2016.
3. Joy Buolamwini and Timnit Gebru. Gender Shades: Intersectional Accuracy Disparities in Commercial Gender Classification. In *Conference on Fairness, Accountability and Transparency, FAT 2018, 23-24 February 2018, New York, NY, USA*, pages 77–91, 2018. URL: `http://proceedings.mlr.press/v81/buolamwini18a.html`.
4. Ángel Cabrera, Will Epperson, Fred Hohman, Minsuk Kahng, Jamie Morgenstern, and Duen Horng Chau. FairVis: Visual Analytics for Discovering Intersectional Bias in Machine Learning. *IEEE Conference on Visual Analytics Science and Technology (VAST)*, 2019. URL: `https://poloclub.github.io/FairVis/`.
5. Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3):15:1–15:58, 2009. `doi:10.1145/1541880.1541882`.
6. Moses Charikar, Jacob Steinhardt, and Gregory Valiant. Learning from untrusted data. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 47–60, 2017.
7. Andrew F Emmott, Shubhomoy Das, Thomas Dietterich, Alan Fern, and Weng-Keen Wong. Systematic construction of anomaly detection benchmarks from real data. In *Proceedings of the ACM SIGKDD workshop on outlier detection and description*, pages 16–21. ACM, 2013.
8. Vitaly Feldman, Parikshit Gopalan, Subhash Khot, and Ashok Kumar Ponnuswami. On Agnostic Learning of Parities, Monomials, and Halfspaces. *SIAM J. Comput.*, 39(2):606–645, 2009. `doi:10.1137/070684914`.
9. Oded Goldreich and Leonid A. Levin. A Hard-Core Predicate for all One-Way Functions. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing, May 14-17, 1989, Seattle, Washigton, USA*, pages 25–32, 1989.
10. Parikshit Gopalan, Vatsal Sharan, and Udi Wieder. PIDForest: Anomaly detection via partial identification. In *Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019*, 2019.
11. Sudipto Guha, Nina Mishra, Gourav Roy, and Okke Schrijvers. Robust Random Cut Forest Based Anomaly Detection on Streams. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, pages 2712–2721, 2016.
12. Sushrut Karmalkar, Adam R. Klivans, and Pravesh K. Kothari. List-Decodable Linear Regression. *CoRR*, abs/1905.05679, 2019. `arXiv:1905.05679`.

13 Matti Karppa, Petteri Kaski, and Jukka Kohonen. A Faster Subquadratic Algorithm for Finding Outlier Correlations. *ACM Trans. Algorithms*, 14(3):31:1–31:26, June 2018. doi:10.1145/3174804.

14 Eyal Kushilevitz and Yishay Mansour. Learning Decision Trees Using the Fourier Spectrum. *SIAM J. Comput.*, 22(6):1331–1348, 1993. doi:10.1137/0222080.

15 Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation Forest. In *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM 2008), December 15-19, 2008, Pisa, Italy*, pages 413–422, 2008.

16 Ryan O'Donnell. *Analysis of boolean functions*. Cambridge University Press, 2014.

17 Ryan O'Donnell and Yu Zhao. On Closeness to k-Wise Uniformity. In *APPROX-RANDOM*, 2018.

18 Gregory Valiant. Finding Correlations in Subquadratic Time, with Applications to Learning Parities and the Closest Pair Problem. *J. ACM*, 62(2):13:1–13:45, May 2015. doi:10.1145/2728167.

19 Sergey Yekhanin. Personal Communication, 2019.

9 Missing Proofs

▶ **Lemma 6.** *Given subcubes C and D such that $D \subseteq C \subseteq \{\pm 1\}^n$, and a density function ψ, it holds that:*

$$\langle \psi, \mu_D \rangle = \langle \psi, \mu_C \rangle \cdot \left\langle \psi|_C, \mu_D|_C \right\rangle$$

Proof. We have

$$\langle \psi, \mu_D \rangle = \langle \psi, \mu_C \rangle \cdot \left\langle \frac{\psi}{\langle \psi, \mu_C \rangle}, \mu_D \right\rangle = \langle \psi, \mu_C \rangle \cdot \left\langle \psi|_C, \mu_D|_C \right\rangle$$ ◀

where the second equality follows from $D \subseteq C$.

▶ **Lemma 8.** *Let $\mathrm{codim}(C) = k$. We have*

$$\begin{aligned}\textsc{Skew}_\psi(C) &= 2^k \Pr_{\mathbf{x}\sim\psi}[\mathbf{x} \in C] - 1 \\ &= \frac{1}{\Pr_{x\sim\mu}[\mathbf{x} \in C]} \left(\Pr_{x\sim\psi}[\mathbf{x} \in C] - \Pr_{x\sim\mu}[\mathbf{x} \in C] \right).\end{aligned}$$

Proof. We have

$$\begin{aligned}\langle \psi, \mu_C \rangle &= \mathop{\mathbb{E}}_{\mathbf{x}\in\mu_C}[\psi(\mathbf{x})] \\ &= \sum_{x\in C} \frac{\psi(x)}{2^{n-k}} \\ &= 2^k \sum_{x\in C} \Pr_{\mathbf{x}\sim\psi}[\mathbf{x} = x] \\ &= 2^k \Pr_{\mathbf{x}\sim\psi}[\mathbf{x} \in C].\end{aligned}$$

Hence

$$\begin{aligned}\langle \psi, \mu_C \rangle - 1 &= 2^k \Pr_{\mathbf{x}\sim\psi}[\mathbf{x} \in C] - 1 \\ &= \frac{\Pr_{\mathbf{x}\sim\psi}[\mathbf{x} \in C] - 2^{-k}}{2^{-k}}.\end{aligned}$$

Since $\Pr_{\mathbf{x}\sim\mu}[\mathbf{x} \in C] = 2^{-k}$, the claim follows. ◀

▶ **Lemma 10.** *For any $K \subseteq [n]$, we have*

$$\sum_{\substack{D=(K,w)\\ w\in\{\pm1\}^K}} \textsc{Skew}(D) = 0.$$

Proof. Consider the sum:

$$\sum_{\substack{D=(K,w)\\ w\in\{\pm1\}^K}} \textsc{Skew}(D) = \sum_{\substack{D=(K,w)\\ w\in\{\pm1\}^K}} 2^k \left(\Pr_{x\sim\psi}[\mathbf{x}\in D] - \Pr_{x\sim\mu}[\mathbf{x}\in D] \right) = 0.$$

The first equality comes from Lemma 8, the second follows since the set of cubes D form a partition of $\{\pm1\}^n$. ◀

▶ **Lemma 11.** *If $\{C_1,\ldots,C_{2^\ell}\}$ is a partition of C, then*

$$\textsc{Skew}_\psi(C) = \frac{1}{2^\ell}\sum_{i=1}^{2^\ell} \textsc{Skew}_\psi(C_i).$$

Proof. We have

$$\begin{aligned}
\textsc{Skew}_\psi(C) &= 2^k \left(\Pr_{\mathbf{x}\sim\psi}[\mathbf{x}\in C] - \frac{1}{2^k} \right) \\
&= \frac{2^{k+\ell}}{2^\ell} \left(\sum_{i=1}^{2^\ell} \left(\Pr_{\mathbf{x}\sim\psi}[\mathbf{x}\in C_i] - \frac{1}{2^{k+\ell}} \right) \right) \\
&= \frac{1}{2^\ell} \left(\sum_{i=1}^{2^\ell} 2^{k+\ell} \left(\Pr_{\mathbf{x}\sim\psi}[\mathbf{x}\in C_i] - \frac{1}{2^{k+\ell}} \right) \right) \\
&= \frac{1}{2^\ell}\sum_{i=1}^{2^\ell} \textsc{Skew}_\psi(C_i).
\end{aligned}$$

◀

▶ **Corollary 18.** *For $J\subseteq[n]$ and a distribution ψ over $\{\pm1\}^n$,*

$$W^{\le k}(\psi, J) \le 2^{|J|} e^2 (\ln(e\,\|\psi\|_\infty))^k.$$

Proof. Recall that

$$W^{\le k}(\psi, J) = \sum_{T\subseteq J} \sum_{\substack{S\subseteq[n]\setminus J\\ |S|\le k}} \widehat{\psi}(S\cup T)^2$$

We will show that for any $T\subseteq J$, we have

$$\sum_{\substack{S\subseteq[n]\setminus J\\ |S|\le k}} \widehat{\psi}(S\cup T)^2 \le e^2(\ln(e\,\|\psi\|_\infty))^k. \tag{8}$$

The claim will then follow by summing over all $2^{|J|}$ choices of $T\subseteq J$. To prove Equation (8), we define $\psi^{(T)}:\{\pm1\}^{[n]\setminus J}\to\mathbb{R}$ as

$$\psi^{(T)}(x) = \mathop{\mathbb{E}}_{\mathbf{z}\sim\{\pm1\}^J}[\psi(x\circ\mathbf{z})\chi_T(\mathbf{z})].$$

Note that unlike ψ, $\psi^{(T)}$ can be negative. By orthogonality of characters, we have

$$\psi^{(T)}(x) = \sum_{S \subseteq [n] \setminus J} \widehat{\psi}(S \cup T)\chi_S(x).$$

Since $\psi^{(T)}$ is a signed average of ψ which is non-negative, we have $\left\|\psi^T\right\|_\infty \leq \|\psi\|_\infty$ and

$$\begin{aligned}\left\|\psi^T\right\|_1 &= \mathop{\mathbb{E}}_{\mathbf{x} \in \{\pm 1\}^{[n]\setminus J}} \left[\left| \mathop{\mathbb{E}}_{\mathbf{z} \sim \{\pm 1\}^J} [\psi(x \circ \mathbf{z})\chi_T(\mathbf{z})] \right|\right] \\ &\leq \mathop{\mathbb{E}}_{\mathbf{x} \in \{\pm 1\}^{[n]\setminus J}} \left[\left| \mathop{\mathbb{E}}_{\mathbf{z} \sim \{\pm 1\}^J} [|\psi(x \circ \mathbf{z})|] \right|\right] = \mathop{\mathbb{E}}_{\mathbf{x} \in \{\pm 1\}^n} [|\psi(\mathbf{x})|] = 1.\end{aligned}$$

The proof of Theorem 17 only uses bounds on the 1 and ∞ norms of ψ. Hence we can repeat the same proof with $\psi^{(T)}$ to get an identical bound. This proves Equation (8). ◀

▶ **Lemma 19.** *For $P \subseteq [n]$ and a distribution ψ over $\{\pm 1\}^n$, the restriction ψ_P is given by*

$$\psi_P(y) = \sum_{S \subseteq P} \widehat{\psi}(S)\chi_S(y).$$

Proof. Let $|P| = p < n$. For $x \in \{\pm 1\}^P$, we have

$$\psi_P(y) = 2^p \Pr_{\mathbf{x} \sim \psi}[\mathbf{x}_P = y] = \sum_{z \in \{\pm 1\}^{\bar{P}}} \frac{\psi(y \circ z)}{2^{n-p}} = \mathop{\mathbb{E}}_{\mathbf{z} \sim \mu_{\bar{P}}} [\psi(y \circ \mathbf{z})].$$

where the last expectation is over the bits $\mathbf{z}$ assigned to $\bar{P}$ being chosen uniformly at random. Using the Fourier expansion of ψ,

$$\mathop{\mathbb{E}}_{\mathbf{z} \sim \mu_{\bar{P}}} [\psi(y \circ \mathbf{z})] = \sum_{S \subseteq [n]} \widehat{\psi}(S) \mathop{\mathbb{E}}_{\mathbf{z} \sim \mu_{\bar{P}}} \chi_S(y \circ \mathbf{z}) = \sum_{S \subseteq P} \widehat{\psi}(S)\chi_S(y).$$ ◀

▶ **Lemma 20.** *Let $P \subsetneq [n]$. Let ψ' be a distribution on $\{\pm 1\}^P$. Define a distribution ψ on $\{\pm 1\}^n$ by $\psi(x) = \psi'(x_P)$. Then*

1. *ψ is the product distribution of ψ' with the uniform distribution on $\{\pm 1\}^{\bar{P}}$.*
2. *$\|\psi\|_\infty = \|\psi'\|_\infty$.*
3. *C is a minimal skewed subcube under ψ iff it is a minimal skewed subcube under ψ'.*

Proof. It follows that ψ is a distribution since it is non-negative and $\|\psi\|_1 = 1$. Since the uniform distribution on $\{\pm 1\}^{(\bar{P})}$ is given by $\mu_{\bar{P}}(y) = 1$ for all $y \in \{\pm 1\}^{\bar{P}}$, it follows that $\psi = \psi' \times \mu_{\bar{P}}$ is the product of ψ' and $\mu_{\bar{P}}$.

Claim (2) follows trivially from the definition of ψ.

For claim (3), we show that if $C = (K, y)$ is a minimal skewed subcube under ψ, then $K \subseteq P$. Indeed, suppose $i \in K \setminus P$. Consider the parent subcube $D \supsetneq C$ obtained by *freeing* the coordinate i. Then $\Pr_{\mathbf{x} \in \psi}[\mathbf{x} \in C] = \Pr_{\mathbf{x} \in \psi}[\mathbf{x} \in D]/2$, whereas $\operatorname{codim}(C) = \operatorname{codim}(D) + 1$, hence

$$\textsc{Skew}_\psi(C) = 2^{\operatorname{codim}(C)} \Pr_{\mathbf{x} \in \psi}[\mathbf{x} \in C] - 1 = \textsc{Skew}_\psi(D).$$

This violates the definition of minimality. In the other direction, it is easy to see that a minimal skewed subcube under ψ' is also a minimal skewed subcube under ψ. ◀

▶ **Lemma 21.** *Let $C = (J, z)$. Then*

$$\psi|_C(x) = \sum_{S \subseteq [n] \setminus J} \chi_S(x) \frac{\sum_{T \subseteq J} \widehat{\psi}(S \cup T)\chi_T(z)}{\langle \psi, \mu_C \rangle}.$$

Proof. Recall that $\psi|_C(x) = \psi(x)/\langle \psi, \mu_C \rangle$ for $x \in C$. For $x \in \{\pm 1\}^{[n]\setminus J}$ let $x \circ z$ denote the string obtained by setting coordinates in J to z and those in $[n] \setminus J$ to x. Then

$$\begin{aligned}
\psi|_C(x) &= \frac{\psi(x \circ z)}{\langle \psi, \mu_C \rangle} \\
&= \frac{1}{\langle \psi, \mu_C \rangle} \sum_{S \subset [n]\setminus J} \sum_{T \subset J} \chi_{S \cup T}(x \circ z)\widehat{\psi}(S \cup T) \\
&= \frac{1}{\langle \psi, \mu_C \rangle} \sum_{S \subset [n]\setminus J} \sum_{T \subset J} \chi_S(x)\chi_T(z)\widehat{\psi}(S \cup T) \\
&= \sum_{S \subset [n]\setminus J} \chi_S(x) \frac{\sum_{T \subset J} \chi_T(z)\widehat{\psi}(S \cup T)}{\langle \psi, \mu_C \rangle}
\end{aligned}$$

◀

▶ **Lemma 32** ([19]). *Let $d \geq 2$ be even and let $n + 1 = 2^l \geq d$. There exists a $\mathbb{F}_2$-linear code $\mathcal{C}_{BCH} \subseteq \{0,1\}^n$ with minimum distance d, which contains $\Omega(n^{d/2+1})$ minimum weight codewords.*

Proof. Let $d = 2e + 2$ for $e \geq 0$. We use the fact that BCH codes are $[n, n - el - 1, 2e + 2)]$ codes [2, Theorem 16.21]. We need to show that there are many minimum weight codewords. For this, let us consider the parity check matrix H which has dimension $(el + 1) \times n$ so that $\mathcal{C}_{BCH} = \{x \in \mathbb{F}_2^n : Hx = 0\}$.

Now let us consider the mapping $x \to Hx$ for all $x : \mathrm{wt}(x) = e + 1$. This maps each x to a vector $y \in \{0,1\}^{el+1}$. For each $y \in \{0,1\}^{el+1}$, let $b_y = |\{x : \mathrm{wt}(x) = e + 1, Hx = 0\}|$ be the number of vectors of weight $e + 1$ mapped to y. Then we have $\sum_y b_y = \binom{n}{e+1}$, hence

$$\sum_y b_y^2 \geq \frac{\left(\sum_y b_y\right)^2}{2^{el+1}} = \frac{\binom{n}{e+1}^2}{2(n+1)^e}.$$

For $x_1 \neq x_2$ both of weight $e + 1$, if $Hx_1 = Hx_2$ then $H(x_1 + x_2) = 0$, hence $x_1 + x_2$ is a non-zero codeword of weight at most $2e + 2$, hence it is in fact a minimum weight codeword. Since there are $\binom{2e+2}{e+1} < 2^{2e+2}$ ways to write each vector of weight $2e + 2$ as such as sum, hence the number of vectors of codewords of weight $2e + 2$ is at least

$$\frac{1}{2^{2e+2}} \sum_y \binom{b_y}{2} = \frac{1}{2^{2e+3}} \sum_y (b_y^2 - b_y) \geq \frac{1}{2^{2e+3}} \left(\frac{\binom{n}{e+1}^2}{2(n+1)^e} - \binom{n}{e+1} \right) = \Omega_e(n^{e+2}).$$ ◀

▶ **Theorem 34** (Algorithm for Positive Skew). *Given sample access to a distribution ψ on $\{\pm 1\}^n$, integer $k \leq n$, and parameters $\gamma \in (0, 2^k - 1]$, $\epsilon \in (0, 1]$ and $\lambda \in [0, 1]$, there is an algorithm that returns all (γ, ϵ)-minimal skewed subcubes of codimension at most k in time:*

$$\tilde{O}\left(n^{k\left(\frac{\omega}{3-\lambda}\right)}\right) + k^{O(k)} \cdot (\ln(e \|\psi\|_\infty))^k \left(\frac{\tilde{O}(n^{k/3})}{(\epsilon\sqrt{\gamma})^{4/\lambda}} + \frac{\mathrm{poly}(n)}{(\epsilon\sqrt{\gamma})^{2k}} \right)$$

where ω is the matrix multiplication exponent, and $\tilde{O}$ hides $\mathrm{poly}\log n$ *factors.*

▶ **Theorem 35** (Algorithm for Negative Skew). *Given sample access to a distribution ψ on $\{\pm 1\}^n$, integer $k \leq n$, and parameters $\gamma \in (0,1]$, $\epsilon \in (0,1]$ and $\lambda \in [0,1]$, there is an algorithm that returns all $(-\gamma,\epsilon)$-minimal skewed subcubes of codimension at most k in time:*

$$\tilde{O}\left(n^{k\left(\frac{\omega}{3-\lambda}\right)}\right) + k^{O(k)} \cdot (\ln(e\,\|\psi\|_\infty))^k \left(\frac{\tilde{O}(n^{k/3})}{(\epsilon\gamma)^{4/\lambda}} + \frac{\mathrm{poly}(n)}{(\epsilon\gamma)^{2k}}\right)$$

where ω is the matrix multiplication exponent, and $\tilde{O}$ hides $\mathrm{poly}\log n$ *factors.*

The algorithm is FindSkew$^+(\emptyset,\emptyset)$ in the positive case and FindSkew$^-(\emptyset,\emptyset)$ in the negative case with the nondeterminism replaced. In the positive case, one could replace the enumeration of Fourier coefficients (this is Line 6 in Algorithm 1 and Line 4 in Algorithm 2) by a call to FindFourierCoefficients. However this would naively yield a running time bound of $O(n^{0.8k}) \cdot k^{O(k)} \left(\ln(e\,\|\psi\|_\infty/\epsilon\gamma)\right)^k/(\epsilon\gamma)^{2k+O(1)}$. We show the stronger bound claimed in the theorem by making the following modification. Instead of running FindFourierCoefficients at every recursive call, we run it once at the top level to get the list L of heavy coefficients for ψ, and "deduce" the heavy Fourier coefficients for each restricted distribution $\psi|_C$ from the original list L. To do so efficiently, we require a data structure which we now explain.

We preprocess L by creating a graph G_L. Vertices of this graph are indexed by elements of the power set $2^{[n]}$. For each coefficient $S \in L$, and each subset $T \subset S$, add the directed edge $T \to S$. Furthermore, each T stores k lists, where the i^{th} list contains all sets S in the out-neighborhood of T such that $|S \backslash T| = i$. Since $2^k|L|$ is a bound on both the total number of edges and the total number of vertices in the graph, creating the graph takes $O(2^k|L|)$ time. Creating the partitions of the out-neighborhoods also takes $O(2^k|L|)$ times since each edge in the graph need only be processed once.

We summarize the algorithm below for reference.

Algorithm 4 PreprocessCoefficients(L).

```
V, E ← ∅
for S ∈ L do
    for T ⊆ S do
        V ← V ∪ {S, T}
        E ← E ∪ {(T → S)}
for edge (T → S) ∈ E do
    Let i = |S\T|, and add S to the i^th list stored at T.
return G_L = (V, E)
```

Next, given G_L the preprocessed form of L, a target subcube $C = (J, z)$ and a threshold τ, we show how to output all Fourier coefficients $\widehat{\psi|_C}(S)$ such that $\widehat{\psi|_C}(S) \geq \tau$.

▶ **Lemma 41.** *Let $L = \{S \subseteq [n] : |S| \leq k, |\widehat{\psi}(S)| \geq \tau/4^k\}$, and G_L be the output of* Preprocess(L). *Then* DeduceSubcubeCoefficients(G_L, C, τ) *returns the list of Fourier coefficients of $\psi|_C$ of degree at most k and magnitude at least τ. Furthermore,* DeduceSubcubeCoefficients(G_L, C, τ) *runs in time*

$$\mathrm{poly}(n) \cdot O(|L'|) \leq \mathrm{poly}(n) \cdot 2^{O(k)} \cdot \frac{(\ln(e\,\|\psi\|_\infty))^{k-|J|}}{\tau^2}.$$

Algorithm 5 DeduceSubcubeCoefficients(G_L, C, τ).

1: Let J be the coordinates fixed by C.
2: $L' \leftarrow \emptyset$
3: **for** $T \subseteq J$ **do**
4: | **for** S out-neighbor of T in G_L with $|S \backslash T| \leq k - |J|$ **do**
5: | | Check by sampling if $\left|\widehat{\psi|_C}(S \backslash J)\right| \geq 3\tau/4$ to accuracy $\tau/4$. If so, add $S \backslash J$ to L'.
6: **return** L'.

Proof. The output L' consists only of sets S' such that $S' \cup J \in L$, $S' \cap J = \emptyset$ and $|S'| \leq k - |J|$. Furthermore it contains all S' meeting these criteria such that $\left|\widehat{\psi|_C}(S')\right| \geq \tau$, note that for any set $H \in L$ such that $|H| \leq k$, if T is chosen to be $H \cap J$, then $|H \backslash J| = |H \backslash T| \leq k - |J|$ and thus the algorithm will add $H \backslash J$ to the output.

To obtain the claim of the lemma, we need to argue that for every Fourier coefficient $\widehat{\psi|_C}(S)$ of absolute value at least τ, the set S must appear in L'. We need the following consequence of Lemma 21: let ψ be a distribution on $\{\pm 1\}^n$ and let $C = (J, z)$ be a subcube. Then

$$\widehat{\psi|_C}(S) = \sum_{T \subseteq J} \frac{\chi_T(z)\widehat{\psi}(S \cup T)}{\langle \psi, \mu_C \rangle}.$$

It follows that every coefficient $\widehat{\psi|_C}(S)$ is the signed sum of at most 2^k coefficients $\widehat{\psi}(R) = \widehat{\psi}(S \cup T)$, which is then scaled by $1/\langle \psi, \mu_C \rangle$ (at most 2^k). Furthermore, if $|S| \leq k - |J|$, then each such coefficient has $|R| \leq |S| + |T| \leq k$. Thus if $\widehat{\psi|_C}(S) \geq \tau$, there is at least one $R \subseteq J$ with $|R| \leq k$ such that $|\widehat{\psi}(R)| \geq \tau/4^k$, which means that $R \in L$.

Finally, the running time claim follows from Corollary 18 and the fact that for every $S \in L$ we have $\widehat{\psi}(S) \geq \tau/4^k$. ◀

We are now ready to prove our main theorem. We start with the positive case.

Proof of Theorem 34. We start by running FindFourierCoefficients$(\psi, k, \rho^+, \lambda)$ once, with $\rho^+ = \epsilon\sqrt{\gamma}/16^k$. This outputs a list L^+ containing all S where $|S| \leq k$ and $\widehat{\psi}(S) \geq \epsilon\sqrt{\gamma}/4^k$. Subsequently, we compute $G_{L^+} \leftarrow$ PreprocessCoefficients(L^+) using the output. This set up phase has running time R^+ bounded by

$$R^+ \leq \tilde{O}\left(n^{k\left(\frac{\omega}{3-\lambda}\right)}\right) + \frac{k^{O(k)}\tilde{O}\left(n^{k/3}\right)\ln(e\,\|\psi\|_\infty)^k}{(\epsilon\sqrt{\gamma})^{4/\lambda}}.$$

Next we run FindSkew$^+(\emptyset, \emptyset)$ but we replace the nondeterministic enumeration of Fourier coefficients (this is Line 6 in Algorithm 1) by DeduceSubcubeCoefficients(G_{L^+}, C, τ^+) where $\tau^+ := \epsilon\sqrt{\gamma}/\left(k_t \cdot \binom{k_t}{|S_t|}\right)$. We also replace the nondeterministic choice of z (Line 7 in Algorithm 1) by simple enumeration over all possible choices. Correctness follows from Lemma 24 together with Lemmas 39 and 41 and it remains to show the running time bound.

By Lemma 41, at every subcube C the algorithm spends time at most

$$\mathrm{poly}(n) \cdot 2^{O(k)} \cdot \frac{(\ln(e\,\|\psi\|_\infty))^{k-|J|}}{(\tau^+)^2} = \mathrm{poly}(n) \cdot 2^{O(k)} \cdot \frac{(\ln(e\,\|\psi\|_\infty))^{k-|J|}}{(\epsilon\sqrt{\gamma})^2}$$

to run DeduceSubcubeCoefficients.

On the other hand, the proof of Lemma 26 requires that the branching factor of FindSkew$^+$ be bounded as in Lemma 25. Since this bound is at least $(\ln(e\,\|\psi\|_\infty))^{k-|J|}/(\epsilon\sqrt{\gamma})$ (and we may assume WLOG that the branching factor is *at least* this threshold), we may amortize the cost of each call to DeduceSubcubeCoefficients by charging to each child call the average running time per child. The time spent per child is $\mathrm{poly}(n)\cdot 2^{O(k)}$, and we argued in Lemma 26 that the total number of recursive calls to FindSkew$^+$ is at most

$$k^{O(k)}\left(\frac{\ln(e\,\|\psi\|_\infty)}{\epsilon^2\gamma}\right)^k.$$

Thus we may bound the running time of FindSkew$^+(\emptyset,\emptyset)$ by this expression as well.

To conclude, the final running time of the algorithm in the positive skew case is:

$$\begin{aligned}
&R^+ + \mathrm{poly}(n,k^k)\cdot k^{O(k)}\left(\frac{\ln(e\,\|\psi\|_\infty)}{\epsilon^2\gamma}\right)^k\\
&\le \tilde{O}\left(n^{k\left(\frac{\omega}{3-\lambda}\right)}\right) + \frac{k^{O(k)}\tilde{O}\left(n^{k/3}\right)\ln(e\,\|\psi\|_\infty)^k}{(\epsilon\sqrt{\gamma})^{4/\lambda}} + \mathrm{poly}(n,k^k)\cdot k^{O(k)}\left(\frac{\ln(e\,\|\psi\|_\infty)}{\epsilon^2\gamma}\right)^k\\
&\le \tilde{O}\left(n^{k\left(\frac{\omega}{3-\lambda}\right)}\right) + k^{O(k)}\cdot(\ln(e\,\|\psi\|_\infty))^k\left(\frac{\tilde{O}\left(n^{k/3}\right)}{(\epsilon\sqrt{\gamma})^{4/\lambda}} + \frac{\mathrm{poly}(n)}{(\epsilon\sqrt{\gamma})^{2k}}\right).
\end{aligned}$$

◀

The negative case is identical, but with a different setting of parameters.

Proof of Theorem 35. In this case we run FindFourierCoefficients(ψ,k,ρ^-,λ) with $\rho^- = \epsilon\gamma/16^k$. This outputs the list L^-, and we set $G_{L^-} \leftarrow$ PreprocessCoefficients(L^-). This all has running time R^- bounded by

$$R^- \le \tilde{O}\left(n^{k\left(\frac{\omega}{3-\lambda}\right)}\right) + \frac{k^{O(k)}\tilde{O}\left(n^{k/3}\right)\ln(e\,\|\psi\|_\infty)^k}{(\epsilon\gamma)^{4/\lambda}}.$$

Now we run FindSkew$^-(\emptyset,\emptyset)$, with Fourier coefficient enumeration (Algorithm 2 in Algorithm 2) replaced with DeduceSubcubeCoefficients(G_{L^-},C,τ^-), and we set $\tau^- = \epsilon\gamma/\left(k_t\cdot\binom{k_t}{|S_t|}\right)$. The analysis is identical to the positive case, and the final running time is:

$$\tilde{O}\left(n^{k\left(\frac{\omega}{3-\lambda}\right)}\right) + k^{O(k)}\cdot(\ln(e\,\|\psi\|_\infty))^k\left(\frac{\tilde{O}\left(n^{k/3}\right)}{(\epsilon\gamma)^{4/\lambda}} + \frac{\mathrm{poly}(n)}{(\epsilon\gamma)^{2k}}\right).$$

◀

Combinatorial Lower Bounds for 3-Query LDCs

Arnab Bhattacharyya
National University of Singapore, Singapore
arnabb@nus.edu.sg

L. Sunil Chandran
Indian Institute of Science, Bangalore, India
sunil@iisc.ac.in

Suprovat Ghoshal
Indian Institute of Science, Bangalore, India
suprovat@iisc.ac.in

Abstract

A code is called a *q-query locally decodable code* (LDC) if there is a randomized decoding algorithm that, given an index i and a received word w close to an encoding of a message x, outputs x_i by querying only at most q coordinates of w. Understanding the tradeoffs between the dimension, length and query complexity of LDCs is a fascinating and unresolved research challenge. In particular, for 3-query binary LDC's of dimension k and length n, the best known bounds are: $2^{k^{o(1)}} \geq n \geq \tilde{\Omega}(k^2)$.

In this work, we take a second look at binary 3-query LDCs. We investigate a class of 3-uniform hypergraphs that are equivalent to *strong* binary 3-query LDCs. We prove an upper bound on the number of edges in these hypergraphs, reproducing the known lower bound of $\tilde{\Omega}(k^2)$ for the length of strong 3-query LDCs. In contrast to previous work, our techniques are purely combinatorial and do not rely on a direct reduction to 2-query LDCs, opening up a potentially different approach to analyzing 3-query LDCs.

2012 ACM Subject Classification Theory of computation → Error-correcting codes

Keywords and phrases Coding theory, Graph theory, Hypergraphs

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.85

Funding *Arnab Bhattacharyya*: Partially supported by NUS Startup Grant R-252-000-A33-133.

Acknowledgements AB thanks Sivakanth Gopi, Nikhil Srivastava, and Luca Trevisan for many useful discussions about this problem.

1 Introduction

A code $\mathcal{C}$ is said to be a *q-query locally decodable code (LDC)* if it is possible to recover any symbol x_i of a message x by querying $\mathcal{C}(x)$ on at most q locations, such that even if a constant fraction of $\mathcal{C}(x)$ is corrupted, the decoder returns x_i with high probability. LDCs already appeared in the PCP literature (e.g., implicitly in [1]) but they were first explicitly formulated by Katz and Trevisan in [10]. LDCs have attracted attention not only because of their immediate relevance to data transmission and data storage but also because of their surprising connections to complexity theory and cryptography ([3, 13, 8, 11]). In more recent years, the analysis of LDCs has led to a greater understanding of basic problems in incidence geometry, the construction of design matrices and the theory of matrix scaling, e.g. [2, 7, 6].

Although LDCs have been studied now for two decades, some basic questions remain stubbornly open. In particular, we have the following open question for 3-query LDCs:

▶ **Open Question 1.** *What is the length of the shortest* 3*-query LDC that can encode all* k*-bit binary messages?*

A wild variety of techniques have been used to study the problem. For a while, it was believed that the length n should be exponential in k for 3-query LDCs (indeed, for any constant number of queries). This belief was shattered by a breakthrough work of Yekhanin that designed 3-query LDCs of length subexponential in k (conditional on some number-theoretic conjectures). Subsequent work ([9, 5] reformulated the construction in terms of *matching vector codes* and established an unconditional upper bound of $\exp\left(\exp(O(\sqrt{\log k \log\log k}))\right) = \exp(k^{o(1)})$ on the length.

As for lower bounds on the length of 3-query LDCs, which is the focus of this work, Katz and Trevisan [10] first gave a super linear lower bound of $\Omega(k^{3/2})$, which was then improved to $\Omega\left(k^2/(\log k)^2\right)$ by Kerenedis and de Wolf [12] using quantum information theoretic techniques. The current state-of-the-art is due to Woodruff [14] from over a decade ago where he showed that $n \geq \Omega(k^2/\log k)$.

Given the state of affairs, it is natural to try to prove lower bounds for stronger variants[1] of LDCs where the task should be easier. In this work, we study a restricted form of LDCs which seem to capture most of the challenges associated with general LDCs.

▶ **Definition 2.** *For a given $\delta \in (0,1)$, a code $\mathcal{C} : \{\pm 1\}^k \to \{\pm 1\}^n$ is a* $(3,\delta)$-strong LDC *if for every $i \in [k]$, there exists a set M_i of $\geq \delta n$ disjoint triples in $\binom{[n]}{3}$ such that for every $x \in \{\pm 1\}^k$ and for every triple $(j_1, j_2, j_3) \in M_i$, $x_i = \mathcal{C}(x)_{j_1} \cdot \mathcal{C}(x)_{j_2} \cdot \mathcal{C}(x)_{j_3}$. Moreover, if $i \neq i'$, a triple in M_i intersects a triple in $M_{i'}$ in at most 1 coordinate.*

Known constructions of 3-query LDCs are strong. Conceptually, the main[2] restriction that the above definition makes is that each triple in the matching M_i successfully decodes x_i for *every* x. On the other hand, Katz and Trevisan [10] show that general LDCs yield matchings $M_1, \ldots, M_k$ such that each triple in the matching M_i sucessfully decodes x_i for most (not all) x's.

We show a combinatorial proof of the known $\Omega(k^2/\log k)$ lower bound for the length of code words of 3-query strong LDCs. Here is the main theorem stating the lower bound.

▶ **Theorem 3.** *Let $C : \{\pm 1\}^k \mapsto \{\pm 1\}^n$ be a $(3,\delta)$-strong LDC. Then, $n \geq \Omega_\delta(k^2/\log k)$.*

1.1 Proof Overview

As we already noted, Theorem 3 follows from [14]. Of more interest is our proof technique. Woodruff's lower bound reduces 3-query LDCs to 2-query LDCs and applies known analytic proofs giving tight bounds for 2-query LDCs [12]. On the other hand, our proof is purely combinatorial and does not seem to be a reduction to 2 queries.

Our starting point is the observation that strong 3-query LDCs are equivalent to *even-colored 3-uniform hypergraphs*. A 3-uniform hypergraph is called *linear* if any two edges intersect in at most one vertex.

▶ **Definition 4.** *An* (n,k,δ)-even-colored 3-uniform hypergraph *is a linear edge-colored hypergraph H on n vertices with each edge having a color in $\{1,\ldots,k\}$ such that:*

- **(i)** *For each $i \in [k]$, the edges of color i form a matching of size at least δn, and*
- **(ii)** *If H' is a subgraph of H such that every vertex has even degree in H', then there are an even number of edges of each color in H'.*

[1] For instance, Woodruff in [15] gave an $\Omega(k^2)$ lower bound for the special case of *linear* 3-query LDCs.

[2] The decoding scheme of taking the product (xor) of the codeword bits is without loss of generality (see [14]). The additional condition in Definition 2 about triples in different matchings intersecting only at single coordinates is made for technical convenience and should be avoidable.

Given a $(3, \delta)$-strong LDCs, define the hypergraph H which is the union of the matchings $M_1, \ldots, M_k$ given by Definition 2, and let the color of an edge be the matching it comes from. Then, it is easy to check that both conditions (i) and (ii) are met (see Claim 5). The correspondence naturally goes in the other direction too, although this is not needed in the present work.

We prove an upper bound $k^2 \leq O_\delta(n \log n)$ for (n, k, δ)-even-colored 3-uniform hypergraphs, proving our main theorem. To motivate our proof, let us sketch the corresponding argument for 2-query LDCs (which is also new to the best of our knowledge). Suppose we have a (2-uniform) graph which is the union of k matchings, with edges from the i'th matching having color i. Analogously to condition (ii) of Definition 4, also suppose that any cycle contains an even number of edges of each color. Then, we prove that the number of vertices n is at least $\exp(k)$. For simplicity, suppose the matchings of each color are perfect. Our argument is through coding (ironically!). Fix an arbitrary vertex s. For any vertex $v \neq s$, let its *signature* $S(v)$ be defined as $(n_1, \ldots, n_k)$ where n_i is the parity of the number of edges of color i on a path P from s to v. We claim that $S(v)$ does not depend on the path chosen. This is because if two paths from s to v gave different signatures, this would yield a cycle in which some color occurred an odd number of times. On the other hand, there are at least 2^k different signatures because for any signature $(s_1, \ldots, s_k) \in \{0, 1\}^k$, there is a path from s with exactly s_i edges of color i (since the matchings are perfect). Hence, the number of vertices is at least $\exp(k)$.

Our proof for 3-uniform hypergraphs is in a similar spirit. Instead of a path to define a signature, we use a sequence of *cherries*, borrowing an idea from [4]. A cherry is a pair of hyperedges which uniquely intersect at a hyperedge; see Figure 1. We observe that if the number of edges is sufficiently large, then there are many cherries. We then use this structure to show that there are even subgraphs (i.e., subgraphs in which all vertices have even degrees) which have an odd number of edges of some color. Namely, we construct a "cycle of cherries" in which we know there is a color that appears on a unique edge, yielding the contradiction. More details follow.

Formally, given an (n, k, δ)-even colored hypergraph H which is a union of matchings $M_1, \ldots, M_k$, define the *signature graph* $G = (V', E')$ as follows. The vertex set $V' = V \times V$, and there is an edge in G between the vertices (u_1, v_1) and $(u_2, v_2) \in V'$ whenever there exists a $w \in V$ such that $\{u_1, u_2, w\}$ and $\{v_1, v_2, w\}$ are hyperedges forming a cherry in H. Moreover, such an edge is labeled by the pair of colors $\{i_1, i_2\}$ if $\{u_1, u_2, w\} \in M_{i_1}$ and $\{v_1, v_2, w\} \in M_{i_2}$. The signature graph enjoys the following useful structural property:

> For any vertex x in G and for any color $i \in [k]$, there are at most 4 edges incident to x that have i in their label. $(\star)$

The proof of $(\star)$ follows from the definition of G in terms of cherries (see Claim 8) .

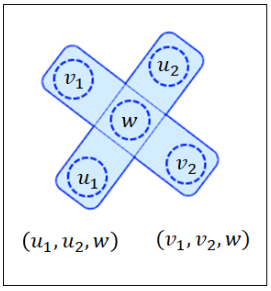


Figure 1 A cherry formed from the edges $\{u_1, w, u_2\}$ and $\{v_1, w, v_2\}$ intersecting at w.

For the sake of contradiction, assume that $k \geq \sqrt{Cn \log n}$ for some large constant C. This, along with a standard averaging argument, implies that there exists a large subgraph G' of the signature graph with minimum degree at least $k^2 n/4n^2 \geq (C/4)\log n$. Now fixing an arbitrary vertex $r \in V(G')$ as root, we iteratively grow a sequence of trees $T_1, T_2, \ldots, T_{\ell+1}$ using edges in G', while maintaining the following *rainbow* condition: For any vertex $x \in V(T_i)$, no color appears more than once among the colors labeling the unique path from r to x in T_i.

We explain how to construct T_{i+1} from T_i so that the above rainbow condition is met. Let L_i denote the leaves of the tree T_i, and let N_i denote the neighbors y of vertices $x \in L_i$ so that the colors labeling the edge (x, y) do not occur on the path from the root r to x in T_i. A short argument (Claim 9) allows us to deduce that N_i must be disjoint from $V(T_i)$, because otherwise, condition (ii) of Definition 4 is violated. Hence, the next tree T_{i+1} can be built by letting $L_{i+1} = N_i$ and adding one edge from each vertex in L_{i+1} to a vertex in L_i.

We continue this process until $|L_\ell| < 2|L_{\ell-1}|$ for some iteration ℓ. We now sketch how to arrive at a contradiction. From the stopping criteria, we know that for every $i < \ell - 1$ we have $|L_{i+1}| \geq 2|L_i| \geq 2^{i+1}$, and therefore the depth of the tree is at most $\ell = O(\log n)$. Therefore, for any $x \in L_{\ell-1}$, the number of colors labeling the path from r to x is at most $O(\log n)$. From property $(\star)$, we get that there are at least $(C/4)\log n - O(\log n) = C' \log n$ neighbors of x that are in L_ℓ (for some other constant C'). Since $|L_\ell| < 2|L_{\ell-1}|$, there exists a vertex $w \in L_\ell$ with at least $(C'/2)\log n$ neighbors in $L_{\ell-1}$. Again, invoking property $(\star)$, for C' large enough, there will be a neighbor $w' \in L_{\ell-1}$ such that that the colors labeling (w, w') do not appear among the $O(\log n)$ labels of the path from r to w. From here, we can conclude that the unique path between w and w' in T_ℓ along with the edge (w, w') forms a cycle in G in which some color appears exactly once. This structure corresponds to a subgraph in H that violates condition (ii) of Definition 4.

In the rest of the paper, we present the argument formally with all the details. It is unclear currently how to extend the analysis to q-query LDCs or how to improve the analysis for 3-query LDCs. But we remain hopeful that by looking at more intricate combinatorial structures than cherries, we can make some progress.

2 Preliminaries

In this section and later, we do not invoke the notion of even-colored subgraphs, and we define objects directly in reference to strong 3-query LDCs.

Given a $(3, \gamma)$-strong LDC, we define the *recovery hypergraph* H, where $V(H) = [n]$ and $E(H) := \cup_{i \in [k]} M_i$ to be the 3-uniform hypergraph which is the union of matchings $M_1, M_2, \ldots, M_k$. For any edge $e \in E(H)$, we say that the color of the hyperedge e is i if e belongs to matching M_i. We use the notation $\mathsf{col}(e)$ to denote the color of the hyperedge e. We additionally assume that H is linear, i.e. no two hyperedges of H intersect in more than one element.

Let L be a hypergraph. Then we define an *augmentation* L' of L as follows: $V(L') = V(L)$ and $E(L')$ is a multiset where each member $e \in E(L')$ also belongs to $E(L)$ but can possibly have a higher multiplicity than the multiplicity of e in $E(L)$. With respect to a hypergraph L where a hyperedge e is allowed to have multiplicity greater than 1, we denote by $\mathsf{mult}_L(e)$ the multiplicity of e in $E(L)$. We may drop the subscript L, if the hypergraph under consideration is clear from the context. Also for $v \in V(L), \deg_L(v) := \sum_{e \in E(L): v \in e} \mathsf{mult}_L(e)$. If for all $v \in V(L), \deg_L(v)$ is even, then L is called an *even hypergraph*. We use $\mathsf{col}(L)$ to denote the multiset of colors associated with edges in $E(L)$.

If L is an augmentation of the recovery hypergraph H, for $1 \leq i \leq k$, we define

$$\lambda_L(i) := \sum_{e \in E(L): \mathsf{col}(e) = i} \mathsf{mult}_L(e).$$

▷ Claim 5. Let L be an augmentation of the recovery hypergraph H. If L is even, then for $1 \leq i \leq k$, $\lambda_L(i)$ is even.

Proof. Suppose for contradiction that there exists i, $1 \leq i \leq k$ such that $\lambda_L(i)$ is odd. Recall that the indices of the code word bits correspond to the vertices of the recovery hypergraph. Let us assume that $y = (y_1, y_2, \ldots, y_n)$ is the code word of a message $(x_1, x_2, \ldots, x_k)$, where $x_i = -1$ and $x_j = 1$ for $j \neq i$. For an edge $e = \{a, b, c\}$ of the recovery hypergraph, $Y^e = y_a.y_b.y_c$. Now it is clear that $\prod_{e \in E(L)} Y^e = 1$, since L is an even augmentation of H.

On the other hand, by definition of the recovery hypergraph, if $\mathsf{col}(e) = t$, then $Y^e = x_t$ for $1 \leq t \leq k$. Therefore $\prod_{e \in E(L)} Y^e = \prod_{1 \leq t \leq k} x_t^{\lambda_L(t)}$. Clearly since for the selected message $x_j = 1$ for $j \neq i$, we infer that $\prod_{e \in E(L)} Y^e = x_i^{\lambda_L(i)} = -1$, if $\lambda_L(i)$ is odd. This is a contradiction. We conclude that for $1 \leq i \leq k$, $\lambda_L(i)$ is even. ◁

The Signature Graph

We define a graph called the *signature graph* G as follows: $V(G) = \{(u, v) : u, v \in V(H); u \neq v\}$ and an edge exists between two vertices (u_1, v_1) and (u_2, v_2) of G if and only if $\{u_1, v_1\} \cap \{u_2, v_2\} = \emptyset$ and there exists a vertex $w \in V(H)$ such that $\{u_1, u_2, w\}, \{v_1, v_2, w\} \in E(H)$. Note that since the recovery hypergraph H is linear, if there exists an edge between two vertices (u_1, v_1) and (u_2, v_2), there is a unique vertex w such that $\{u_1, u_2, w\}, \{v_1, v_2, w\} \in E(H)$. We may say that the vertex w *causes* the edge $((u_1, v_1), (u_2, v_2))$. Given an edge $e \in E(G)$, we define $T(e) = (\{u_1, u_2, w\}, \{v_1, v_2, w\})$ if w causes the edge e. We may abuse the notation and use $T(e)$ to denote the corresponding unordered set. We define $\mathsf{col}(e) = \{i_1, i_2\}$ if $(u_1, u_2, w) \in M_{i_1}$ and $(v_1, v_2, w) \in M_{i_2}$. Note that $i_1 \neq i_2$ since w cannot be in two different edges of the same matching.

▷ Claim 6. The number of edges in the signature graph G is at least $12\gamma^2 nk^2$.

Proof. Since each matching is of size at least γn, the number of hyperedges m in H is at least γnk. It follows that $\sum_{v \in V(H)} \deg(v) = 3m \geq 3\gamma nk$. For any vertex $w \in V(H)$ consider a pair of incident edges, say $\{u_1, u_2, w\}$ and $\{v_1, v_2, w\}$. Since H is linear, $\{u_1, u_2\} \cap \{v_1, v_2\} = \emptyset$. It is easy to see that based on this pair of incident edges, w can *cause* 4 distinct edges of the signature graph G. Therefore the vertex w *causes* $4\binom{d(v)}{2}$ distinct edges of G, where $d(v) = \deg(v)$. As we have mentioned earlier, two different vertices w and w' cannot cause the same edge in G. Therefore $|E(G)| \geq 4\sum_{v \in V(H)} \binom{d(v)}{2} = 2\sum_{v \in V(H)} (d(v)^2 - d(v))$. Recall that $\sum_{v \in V(H)} d(v) = 3m$. Using Cauchy-Schwarz inequality, [3] we get $\sum_{v \in V(H)} (d(v)^2) \geq \frac{9m^2}{n}$. It follows that $|E(G)| \geq \frac{6m}{n}(3m - n) \geq \frac{12m^2}{n} \geq 12\gamma^2 nk^2$. Here we have used $m \geq \gamma nk$ and $m \geq n$, since $\gamma k \geq 1$. ◁

For a subgraph J of the signature graph G, we define H_J to be the augmentation of H, with $V(H_J) = V(H)$ and $E(H_J) = \cup_{e \in E(J)} T(e)$. Note that when we take the union here, we retain multiple copies of a hyperedge if that hyperedge appears in multiple sets $T(e)$ taking part in the union operation. Thus $E(H_J)$ is by definition a multi-set. We extend some of the notation used for hypergraphs to subgraphs of signature graphs also in the following way: We use $\mathsf{col}(J)$ to denote the multiset $\mathsf{col}(H_J)$. A hypergraph H' is *rainbow colored* with respect to an edge coloring if there exist no two hyperedges having the same color. (In

[3] For vectors X, Y, the Cauchy-Schwarz inequality states that $\|X\| \cdot \|Y\| \geq X^\top Y$. Now take $X = (d(v_1), d(v_2), \ldots, d(v_n))^\top$, where $V(H) = \{v_1, v_2, \ldots, v_n\}$ and $Y = (1, 1, \ldots, 1)^T$ to get the required lower bound.

particular, there will not be any hyperedge with multiplicity greater than 1.) A subgraph J of the signature graph is rainbow colored, if H_J is rainbow colored. We may also say J (or H) is *rainbow*, shortening the phrase *rainbow colored*.

▷ Claim 7. Let J be an even subgraph of the signature graph G, i.e. $\forall v \in V(J)$, $\deg(v)$ is even. Then H_J is an *even* augmentation of H.

Proof. Recall that each edge $e = ((u_1, v_1), (u_2, v_2)) \in E(G)$ corresponds to exactly 2 edges in $E(H_J)$, namely the two edges of $T(e) = \{\{u_1, u_2, w\}, \{v_1, v_2, w\}\}$, where w is the unique vertex which *caused* the edge e. We say that w appears in the role of an intermediate vertex and u_1, v_1, u_2 and v_2 appear in the role of signature vertices in $T(e)$. It is easy to see that since in $T(e)$ itself the $degree(w)$ is even, each vertex plays the role of an intermediate vertex an even number of times. Noting that in $T(e)$ each vertex appears in the role of a signature vertex exactly once, it is easy to see that if $x = (u, v)$ is a vertex of J, then u (also v) plays the role of a signature vertex in $\cup_{e \in E_J(x)} T(e)$ (where $E_J(x)$ denotes the set of edges incident on x in J) exactly $\deg_J(x)$ times. Since $\deg_J(x)$ is even, it follows that $\deg_{H_J}(u)$ and $\deg_{H_J}(v)$ are even numbers. ◁

▷ Claim 8. Let x be a vertex of the signature graph G and let $E(x)$ be the set of edges incident on x in G. Let $\mathcal{C} \subseteq [k]$ be a subset of colors. Let $E(x, \mathcal{C}) = \{e \in E(x) : \mathsf{col}(e) \cap \mathcal{C} \neq \emptyset\}$. Then $|E(x, \mathcal{C})| \leq 4|\mathcal{C}|$.

Proof. Let $x = (u, v)$. For an edge $e \in E(x)$, $T(e)$ contains 2 hyperedges, exactly one of which contains u and the other one contains v: Let us denote by $T(e)_1$ and $T(e)_2$ the hyperedges in $T(e)$ that contain u and v respectively. For $1 \leq i \leq k$, $E(x, \mathcal{C}) = \cup_{i \in \mathcal{C}} E_1^i \cup E_2^i$, where $E_j^i = \{e \in E(x) : \mathsf{col}(T(e)_j) = i\}$ for $j = 1, 2$. First we will show that $|E_1^i| \leq 2$. To see this, note that if $\mathsf{col}(T(e)_1) = i$, then $T(e)_1 \in M_i$ and $T(e)_1$ contains u as mentioned earlier. There is a unique hyperedge with these properties since M_i is a matching. Let $T(e)_1 = (u, a, b)$. Then either a or b could have *caused* the edge e. If a caused the edge e, then $T(e)_2$ contains both v and a, and there is a unique edge in $E(H)$ that is a superset of $\{v, a\}$ since H is linear. Similarly if b caused e, then $T(e)_2$ is uniquely determined, since it should contain both v and b. It follows that $|E_1^i| \leq 2$ for all i, $1 \leq i \leq k$. A similar argument shows that $|E_2^i| \leq 2$. It follows that $|E(x, \mathcal{C})| \leq 4|\mathcal{C}|$. ◁

Now we are ready to prove Theorem 3.

3 Proof of Theorem 3

For contradiction, we shall assume that $k^2 > Cn \log n$ where $C = C(\gamma)$ is a sufficiently large constant. We can then lower bound the average degree of G as follows. From claim 6, we know that $|E(G)| \geq 12\gamma^2 nk^2$. On the other hand, the number of vertices in G is at most n^2. Therefore, for $k^2 \geq Cn \log n$, for C large enough, the average degree of G is at least $(2C') \log n$ so that we can find a subgraph $G' \subseteq G$ with minimum degree at least $C' \log n$, where C' is a sufficiently large constant.

Now we fix a vertex $r \in V(G')$, and we grow a rainbow tree rooted at r in G' level by level as follows. Let T_0 be the tree consisting only of the root r and T_1 be the tree consisting of r and all its neighbors. At the ith stage, we will have a tree T_i where $V(T_i) = \dot{\cup}_{i=0}^{i} L_i$ where L_i is the set of vertices in level i. Note that $V_0 = L_0 = \{r\}$ and T_1 consists of 2 levels, $L_0 = \{r\}$ and $L_1 = N_{G'}(r)$. For two vertices x and y, the unique path in T_i from x to y will be denoted by $P(x, y)$.

Moreover at the ith stage, we will make sure that the tree T_i satisfies the following property:

For any vertex $x \in V(T_i)$, $P(r, x)$ is a rainbow path. (1)

▷ Claim 9. If T_i satisfies property (1), and if $e = (x, y)$ is an edge of G' such that $x \in L_i$ and $y \in L_j$, where $i \geq j$, then $\mathsf{col}(e) \cap \mathsf{col}(P(r, x)) \neq \emptyset$.

Proof. Suppose not. Let z be the least common ancestor of x and y in T_i. Since $P(z, x) \subseteq P(r, x)$ is a rainbow path by property (1), $P(z, x) \cup \{(x, y)\}$ is also rainbow. Since $|P(z, y)| \leq |P(z, x)|$, clearly we have $|\mathsf{col}(P(z, x) \cup \{(x, y)\})| > |\mathsf{col}(P(z, y)|$. It follows that there is at least one matching color, say i in the cycle $C = P(z, x) \cup \{(x, y)\} \cup P(y, z)$, such that $\lambda_{H_C}(i) = 1$. But since C is an even subgraph of G, H_C is an even augmentation of H by Claim 7. Then by Claim 5, $\lambda_{H_C}(i)$ should be an even number, a contradiction. ◁

Now we describe how to construct T_{i+1} from T_i by adding a new level L_{i+1}. For a vertex $x \in L_i$ define $N'(x) = \{y \in N_{G'}(x) : \mathsf{col}(x, y) \cap \mathsf{col}(P(r, x)) = \emptyset\}$. Observe that $N'(x) \cap V(T_i) = \emptyset$: This follows from Claim 9, since if there is an edge in G' from a vertex $x \in L_i$ to a vertex $y \in L_j$, $j \leq i$ then $\mathsf{col}(x, y) \cap \mathsf{col}(P(r, x)) \neq \emptyset$, and therefore $y \notin N'(x)$. Define $L_{i+1} = \cup_{x \in L_i} N'(x)$. Clearly $L_{i+1} \cap V(T_i) = \emptyset$. If $L_{i+1} \neq \emptyset$, define a bipartite graph $B_i = (L_i, L_{i+1})$ such that for $x \in L_i$ and $y \in L_{i+1}$, $(x, y) \in E(B_i)$ if and only if $y \in N'(x)$. Now for each $y \in L_{i+1}$, select one vertex y' from L_i such that $(y, y') \in E(B_i)$ to be its parent thus obtaining the new tree T_{i+1}. From the way we defined $N'(x)$ for $x \in L_i$, it is clear that property (1) is satisfied by T_{i+1}. If $|L_{i+1}| \geq 2|L_i|$, we proceed to add the next level. Otherwise we stop the procedure and define the final tree T to be T_{i+1}.

Let L_ℓ be the last level added to the tree. Clearly $\ell > 2$. We observe that $\ell \leq \log n + 1$. Otherwise $|L_{\ell-1}| \geq 2^{t-1} > n$, since $|L_{i+1}| \geq 2|L_i|$ for $i \leq \ell - 2$. Now consider the bipartite graph $B_{\ell-1}$: For each vertex $x \in L_{\ell-1}$, we know by applying Claim 8 with $\mathcal{C} = \mathsf{col}(P(r, x))$ that $E(x, \mathcal{C}) \leq 4|\mathsf{col}(P(r, x))| \leq 8 \log n$. But $|E_{G'}(x)| = \deg_{G'}(x) \geq C' \log n$ and therefore $|N'(x)| \geq (C' - 8) \log n$. Therefore $B_{\ell-1}$ has at least $|L_{\ell-1}|(C' - 8) \log n$ edges. Since $|L_\ell| < 2|L_{\ell-1}|$, there exists a vertex $w \in L_\ell$ such that its degree in $B_{\ell-1}$ is at least $(C'/2 - 4) \log n$. Again by applying Claim 8, this time with $\mathcal{C} = \mathsf{col}(P(r, w))$, at most $8(\log n + 1)$ of these edges can have a common color with any edge in $P(r, w)$. It follows that if C is taken large enough, w has a neighbor $w' \in L_{\ell-1}$ such that $\mathsf{col}(w, w') \cap \mathsf{col}(P(r, w)) = \emptyset$. This contradicts Claim 9 applied to the tree $T = T_\ell$. The situation is depicted in Figure 2. Thus we infer that $k^2 \leq Cn \log n$, which in turn implies that $n = \Omega\left(\frac{k^2}{\log k}\right)$.

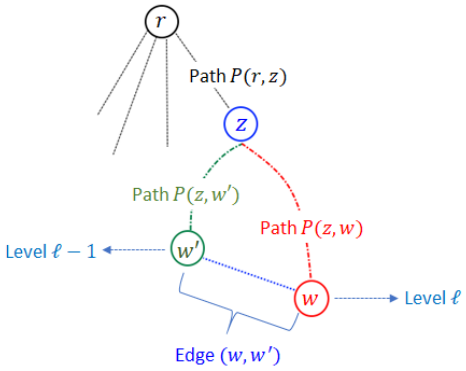


Figure 2 The cycle formed by the concatenation of $P(z, w'), P(z, w)$, and (w, w') corresponds to an even subgraph in H with an odd number of edges having a particular color.

References

1 Laszlo Babai, Lance Fortnow, Leonid Levin, and Mario Szegedy. Checking computations in polylogarithmic time. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing*, pages 21–31, 1991.

2 Boaz Barak, Zeev Dvir, Amir Yehudayoff, and Avi Wigderson. Rank bounds for design matrices with applications to combinatorial geometry and locally correctable codes. In *Proceedings of the 43rd annual ACM symposium on Theory of computing*, pages 519–528. ACM, 2011.

3 Benny Chor, Eyal Kushilevitz, Oded Goldreich, and Madhu Sudan. Private information retrieval. *Journal of the ACM (JACM)*, 45(6):965–981, 1998.

4 D Dellamonica, P Haxell, Tomasz Łuczak, Dhruv Mubayi, Brendan Nagle, Yury Person, Vojtech Rödl, Mathias Schacht, and Jacques Verstraëte. On even-degree subgraphs of linear hypergraphs. *Combinatorics, Probability and Computing*, 21(1-2):113–127, 2012.

5 Zeev Dvir, Parikshit Gopalan, and Sergey Yekhanin. Matching vector codes. *SIAM Journal on Computing*, 40(4):1154–1178, 2011.

6 Zeev Dvir, Shubhangi Saraf, and Avi Wigderson. Breaking the quadratic barrier for 3-LCC's over the reals. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, pages 784–793. ACM, 2014.

7 Zeev Dvir, Shubhangi Saraf, and Avi Wigderson. Improved rank bounds for design matrices and a new proof of Kelly's theorem. In *Forum of Mathematics, Sigma*, volume 2, page e4. Cambridge Univ Press, 2014.

8 Zeev Dvir and Amir Shpilka. Locally decodable codes with two queries and polynomial identity testing for depth 3 circuits. *sicomp*, 36(5):1404–1434, 2007.

9 Klim Efremenko. 3-query locally decodable codes of subexponential length. *SIAM Journal on Computing*, 41(6):1694–1703, 2012.

10 Jonathan Katz and Luca Trevisan. On the efficiency of local decoding procedures for error-correcting codes. In *Proceedings of the 32nd annual ACM symposium on Theory of Computing*, pages 80–86. ACM, 2000.

11 Neeraj Kayal and Shubhangi Saraf. Blackbox polynomial identity testing for depth 3 circuits. In *Proceedings of the 50th annual IEEE symposium on Foundations of Computer Science*, pages 198–207. IEEE, 2009.

12 Iordanis Kerenidis and Ronald de Wolf. Exponential lower bound for 2-query locally decodable codes via a quantum argument. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 106–115. ACM, 2003.

13 Madhu Sudan, Luca Trevisan, and Salil Vadhan. Pseudorandom Generators without the XOR Lemma. *Journal of Computer and System Sciences*, 62(2):236–266, 2001.

14 David Woodruff. New lower bounds for general locally decodable codes. In *Electronic Colloquium on Computational Complexity (ECCC)*, number 006 in 14, 2007.

15 David P. Woodruff. A Quadratic Lower Bound for Three-Query Linear Locally Decodable Codes over Any Field. *Journal of Computer Science Technology*, 27(4):678–686, 2012.

On the Complexity of Decomposable Randomized Encodings, Or: How Friendly Can a Garbling-Friendly PRF Be?

Marshall Ball
Columbia University, New York, NY, USA
marshall@cs.columbia.edu

Justin Holmgren
Simons Institute, Berkeley, CA, USA
holmgren@alum.mit.edu

Yuval Ishai
Technion – Israel Institute of Technology, Haifa, Israel
yuvali@cs.technion.ac.il

Tianren Liu
University of Washington, Seattle, WA, USA
liutr@mit.edu

Tal Malkin
Columbia University, New York, NY, USA
tal@cs.columbia.edu

Abstract

Garbling schemes, also known as decomposable randomized encodings (DRE), have found many applications in cryptography. However, despite a large body of work on constructing such schemes, very little is known about their limitations.

We initiate a systematic study of the DRE complexity of Boolean functions, obtaining the following main results:

- **Near-quadratic lower bounds**. We use a classical lower bound technique of Nečiporuk [Dokl. Akad. Nauk SSSR '66] to show an $\Omega(n^2/\log n)$ lower bound on the size of any DRE for many explicit Boolean functions. For some natural functions, we obtain a corresponding upper bound, thus settling their DRE complexity up to polylogarithmic factors. Prior to our work, no superlinear lower bounds were known, even for non-explicit functions.
- **Garbling-friendly PRFs.** We show that any exponentially secure PRF has $\Omega(n^2/\log n)$ DRE size, and present a plausible candidate for a "garbling-optimal" PRF that nearly meets this bound. This candidate establishes a barrier for super-quadratic DRE lower bounds via natural proof techniques. In contrast, we show a candidate for a *weak* PRF with near-exponential security and linear DRE size.

Our results establish several qualitative separations, including near-quadratic separations between computational and information-theoretic DRE size of Boolean functions, and between DRE size of weak vs. strong PRFs.

2012 ACM Subject Classification Theory of computation → Computational complexity and cryptography

Keywords and phrases Randomized Encoding, Private Simultaneous Messages

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.86

Funding The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either express or implied, of ODNI, IARPA, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein.
Marshall Ball: Supported by an IBM Research PhD Fellowship. This work is based upon work supported in part by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced

Research Projects Activity (IARPA) via Contract No. 2019-1902070006. Part of this work was performed while this author was visiting the third author at Technion in Haifa, Israel.
Justin Holmgren: Research done in part while at IBM Research and in part while at Princeton University supported by the Simons Collaboration on Algorithms and Geometry.
Yuval Ishai: Supported by ERC Project NTSC (742754), NSF-BSF grant 2015782, BSF grant 2018393, and a grant from the Ministry of Science and Technology, Israel and Department of Science and Technology, Government of India.
Tianren Liu: This work was mostly done in Massachusetts Institute of Technology. Research supported by NSF Grants CNS-1350619, CNS-1414119 and CNS-1718161, an MIT-IBM grant and Vinod Vaikuntanathan's DARPA Young Faculty Award.
Tal Malkin: This research is based upon work supported in part by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA) via Contract No. 2019-1902070006.

Acknowledgements We thank Igor Shparlinski for helpful discussions and pointers to the literature on the hidden shift problem. We also thank Siyao Guo, Lucas Kowlaczyk, Ron Rothblum, Jonathan Ullman, and Vinod Vaikuntanathan for related discussions and collaborations.

1 Introduction

Originating from Yao's garbled circuit construction [65], garbling schemes have played an important role in different sub-areas of cryptography. A garbled representation of $f(x)$ is a randomized function $\hat{f}(x;r)$ such that: (1) a sample from the output of $\hat{f}(x;r)$ reveals $f(x)$ and no additional information about x; and (2) each output bit of $\hat{f}$ depends on at most *one* bit of x (but can depend arbitrarily on r); equivalently, each bit of x acts as a selector between two strings that are determined by r. We refer to such a garbled representation $\hat{f}$ for f as a *decomposable randomized encoding* (DRE)[1] of f, and refer to the output length of $\hat{f}$ as its *size*.

Garbling schemes were initially motivated by the goal of efficient secure computation [65, 44, 30, 40]. This still serves as a primary motivation for their study, which has led to many optimized constructions (see, e.g., [12] and references therein).

Over the years, different flavors of garbling schemes have found applications in many other areas of cryptography, including parallel cryptography [8], one-time programs and leakage-resilient cryptography [36], verifiable computation [33, 10], key-dependent message security [13, 5], identity-based encryption [29], and more. See [18, 39, 6] for surveys.

Despite the large body of work on constructing and applying such garbling schemes, very little is known about their *limitations*. Previous relevant works show very limited lower bounds for more liberal notions of garbling. These include either conditional lower bounds that apply to computationally efficient garbling of intractable functions [5, 1] or linear size lower bounds for so-called "2-party PSM protocols" [30, 25, 7].

In this work, we initiate a complexity theoretic study of standard ("DRE-style") garbling schemes, providing *lower bounds* in both *information-theoretic* and *computational settings*.

[1] This notion of garbling roughly corresponds to a *projective garbling scheme* in the terminology of Bellare et al. [18]. We use the DRE terminology when we want to emphasize that we are not interested in the process of "garbling" a given representation of f, but only in the *existence* of a garbled representation $\hat{f}$ with a given complexity.

1.1 Our Contribution

We make two types of contributions: (1) obtaining the first super-linear lower bounds on the DRE size of Boolean functions (with some matching upper bounds), and (2) studying the minimal DRE size of (strong and weak) pseudorandom functions. We now detail both types of results.

1.1.1 Near-quadratic lower bounds and matching upper bounds

We adapt a classical lower bound technique of Nečiporuk [49] to show an $\Omega(n^2/\log n)$ lower bound on the size of any DRE for many explicit Boolean functions $f : \{0,1\}^n \to \{0,1\}$. Nečiporuk showed that functions with many subfunctions cannot have small formulas or branching programs. We provide matching lower bounds on DRE for the same class. In particular, this yields $\Omega(n^2/logn)$ lower bounds on DRE size for almost all functions, including the explicit examples of Element Distinctness, Indirect Storage Access, Clique, Determinant, Matching, and others. These bounds hold in both the information theoretic setting and the exponentially-secure computational setting, provided the DRE admits a sub-exponential decoding algorithm in the latter case.

For the explicit example of Element Distinctness, we obtain a corresponding upper bound, thus settling its DRE complexity up to polylogarithmic factors. Furthermore, since some of the functions that admit nearly quadratic lower bounds on DRE size can be computed by linear-size circuits, our lower bounds establish a near-quadratic gap between the size of computationally secure and information-theoretic DRE in a setting where the input size is polynomially bigger than the computational security parameter. In fact, given that our nearly quadratic lower bounds also apply to computational DREs with security parameter nearly that of the input size, this means, in a concrete sense, that a tradeoff between DRE size and security parameter is inherent!

The only previous lower bounds we are aware of are *linear* lower bounds that also apply to the more liberal 2-party Private Simultaneous Messages (PSM)[2] setting [30, 25, 7] and quadratic lower bounds for *non-Boolean* functions $f : \{0,1\}^n \to \{0,1\}^n$ that follow from the input locality lower bounds of [9]. In contrast to the other classes lower bounded by Nečiporuk's method, such as formulas and branching programs, no superlinear lower bounds on DRE size were known prior to our work, even for a *non-explicit* (e.g., random or worst-case) Boolean function.

1.1.2 Garbling-friendly PRFs

There is a recent line of work on "MPC-friendly" block ciphers [3, 37, 54, 28, 27, 2] and pseudorandom functions (PRFs) [48, 32, 37, 19]. In this context, DRE size is a highly relevant complexity measure that is often used as a benchmark. The question of minimizing the DRE size of PRFs is motivated by the goal of securely evaluating a PRF in a setting where the input (and possibly also the key) is secret-shared between two or more parties. This is useful for natural applications that involve secure keyword search and distributed forms of searchable symmetric encryption; see [19] for discussion.

[2] A DRE can be viewed as an n-party PSM protocol in which each party holds just one bit. Any 2-party PSM lower bound implies a similar DRE lower bound, but the converse is not true.

For the case of exponentially secure (strong) PRFs, we show that the DRE size must be $\Omega(n^2/\log n)$.[3] Finally, we conjecture that a candidate PRF based on the "hidden shift problem" is exponentially secure PRF with almost matching DRE size $O(n^2)$. That is, the function outputs the quadratic character of a hidden shift of the input, determined by the secret key. To defeat known attacks (both quantum and classical), we restrict inputs bounded interval rather than the entire domain. A similar PRF (without the input restriction) has been proposed in [37] as an attractive candidate for MPC-friendly PRF, but in an interactive setting of arithmetic MPC, rather than in the context of garbling. We also present a similar PRF construction with $\Omega(n)$ bits of output, for which we can still obtain a near-quadratic DRE size upper bound.[4] Consequently, modulo the validity of the conjectured security, these PRFs are nearly garbling-optimal.

Interpreted differently, our garbling-friendly PRF candidate establishes a barrier for super-quadratic DRE lower bounds on *explicit* Boolean functions via *natural proof* techniques [53]. In contrast, we show that a recent candidate for a *weak* PRF with near-exponential security due to Boneh et al. [19] has a *linear* DRE size.

Our results imply several qualitative separations, including near-quadratic separations between computational and information-theoretic DRE size of Boolean functions, and between the DRE size of weak vs. strong PRFs.

2 Preliminaries

2.1 Cryptography

▶ **Definition 1** (Pseudorandom Functions [35]). *An* $(s(\cdot), \delta(\cdot))$-secure pseudorandom function (PRF) family *is an ensemble* $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{Z}^+}$*, where each* $\mathcal{F}_\lambda$ *is a keyed family of functions* $\mathcal{F}_\lambda = \{f_k : \{0,1\}^{n(\lambda)} \to \{0,1\}^{m(\lambda)}\}_{k\in\{0,1\}^{\kappa(\lambda)}}$*, satisfying the following security property:*

Pseudorandomness. *For every* $\lambda \in \mathbb{Z}^+$ *and every size-*s *(ensemble) of oracle circuits* $\mathcal{A}$ *(with output in* $\{0,1\}$*),*

$$\left| \mathop{\mathbb{E}}_{\substack{k\leftarrow\{0,1\}^{\kappa(\lambda)} \\ U:\{0,1\}^{n(\lambda)}\to\{0,1\}^{m(\lambda)}}} [\mathcal{A}^{f_k}(1^\lambda) - \mathcal{A}^{U}(1^\lambda)] \right| \leq \delta(\lambda).$$

$n(\cdot)$*,* $m(\cdot)$*, and* $\kappa(\cdot)$ *are respectively called the* input length*,* output length*, and* key length *of* $\mathcal{F}$*.*

▶ **Definition 2** (Weak PRFs [34]). *An* $(s(\cdot), \delta(\cdot))$-secure weak PRF family *is a relaxation of a PRF family as in Definition 1, with the "pseudorandomness" security property replaced by the following notion of "weak pseudorandomness":*

Weak Pseudorandomness. *For every* λ*, the tuples*

$$\big(X_1, \ldots, X_{s(\lambda)}, f_K(X_1), \ldots, f_K(X_{s(\lambda)})\big)$$

[3] This is almost immediate in the non-uniform setting, given our lower bounds. In the appendix we give a constructive proof for this fact in the uniform setting by exhibiting a sublinear test for an average-case variant of the natural property used in Nečiporuk's method.

[4] For this case, multi-bit output, we use input locality bounds of [9] to prove a slightly stronger (and nearly tight) quadratic lower bound (contrast with our $\Omega(n^2/\log n)$ bounds for single bit output).

and

$$\left(X_1, \ldots, X_{s(\lambda)}, Y_1, \ldots, Y_{s(\lambda)}\right)$$

are $(s(\lambda), \delta(\lambda))$*-indistinguishable in the probability space defined by sampling*

$$\begin{aligned} &K \leftarrow \{0,1\}^{\ell(\lambda)} \\ &X_1, \ldots, X_{s(\lambda)} \leftarrow \{0,1\}^{n(\lambda)} \\ &Y_1, \ldots, Y_{s(\lambda)} \leftarrow \{0,1\}^{m(\lambda)}. \end{aligned}$$

▶ **Definition 3.** *Random variables* X *and* Y *are* (s, ϵ)*-indistinguishable if the advantage of every size-*s *circuit in distinguishing* X *from* Y *is at most* ϵ*. We denote this by* $X \approx^{(s,\epsilon)} Y$*.*

2.2 Information Theory

▶ **Definition 4.** *The* min-entropy *of a random variable* X *is* $H_\infty(X) \stackrel{\text{def}}{=} \min_{x \in \text{Supp}(X)} \log_2 \left(\frac{1}{\Pr[X=x]}\right)$.

2.3 Decomposable Randomized Encodings

▶ **Definition 5** (Randomized Encodings)**.** *A* randomized encoding *for a function* $f : \{0,1\}^n \to \mathcal{Y}$ *consists of a "randomness" distribution* $\mathcal{R}$*, an encoding function* $\mathsf{Enc} : \{0,1\}^n \times \mathcal{R} \to \{0,1\}^\ell$*, and a decoding function* $\mathsf{Dec} : \{0,1\}^\ell \to \mathcal{Y}$*.* ℓ *is called* the size of the randomized encoding*.*

A randomized encoding $(\mathcal{R}, \mathsf{Enc}, \mathsf{Dec})$ *for function* $f : \{0,1\}^n \to \mathcal{Y}$ *should satisfy:*

Correctness. *For any input* $x \in \{0,1\}^n$*,*

$$\Pr_{R \leftarrow \mathcal{R}}[\mathsf{Dec}(\mathsf{Enc}(x, R)) = f(x)] = 1.$$

Security. *For all* $x, y \in \{0,1\}^n$ *with* $f(x) = f(y)$*, the distribution of* $\mathsf{Enc}(x, R)$ *is identical to the distribution of* $\mathsf{Enc}(y, R)$ *when sampling* $R \leftarrow \mathcal{R}$*.*

The security can be relaxed to require only that $\mathsf{Enc}(x, R)$ *and* $\mathsf{Enc}(y, R)$ *cannot be effectively distinguished by small circuits.*

(s, δ)***-Security.*** *For all* $x, y \in \{0,1\}^n$ *such that* $f(x) = f(y)$*, for any circuit* $\mathcal{A} : \{0,1\}^\ell \to \{0,1\}$ *of size at most* s*,*

$$\left| \Pr_{R \leftarrow \mathcal{R}}[\mathcal{A}(\mathsf{Enc}(x, R)) = 1] - \Pr_{R \leftarrow \mathcal{R}}[\mathcal{A}(\mathsf{Enc}(y, R)) = 1] \right| \leq \delta.$$

In this paper, we focus on decomposable randomized encoding (DRE)*, which is a randomized encoding that also satisfies an additional property:*

Decomposability. *Each output bit of* $\mathsf{Enc}(x, r)$ *is determined by* r *and 1 bit of input* x*.*

To ease presentation, we also introduce an equivalent definition of DRE. The equivalent definition is used when we prove lower bounds on the size of DRE.

▶ **Definition 6.** *An* (s, δ)*-secure* decomposable randomized encoding (DRE) for a function $f : \{0,1\}^n \to \{0,1\}$ *is a family of random variables*

$$\mathcal{X} = \begin{pmatrix} \mathcal{X}_0^1, \ldots, \mathcal{X}_0^n \\ \mathcal{X}_1^1, \ldots, \mathcal{X}_1^n \end{pmatrix}$$

such that

Correctness. *There is an algorithm* Dec *such that for every* $x \in \{0,1\}^n$,

$$\Pr[\mathsf{Dec}(\mathcal{X}^1_{x_1}, \ldots, \mathcal{X}^n_{x_n}) = f(x)] = 1.$$

Dec *is called a* decoding algorithm *for* $\mathcal{X}$.

(s,δ)-***Security.*** *For all* $x, y \in \{0,1\}^n$ *such that* $f(x) = f(y)$,

$$(\mathcal{X}^1_{x_1}, \ldots, \mathcal{X}^n_{x_n}) \approx^{(s,\delta)} (\mathcal{X}^1_{y_1}, \ldots, \mathcal{X}^n_{y_n}).$$

The size of $\mathcal{X}$ *is*

$$|\mathcal{X}| \stackrel{\mathsf{def}}{=} \sum_{i\in[n], b\in\{0,1\}} \log_2 \left|\operatorname{Supp}(\mathcal{X}^i_b)\right|.$$

2.4 Function Restrictions

▶ **Definition 7** ([50]). *For any function* $f : X^n \to Y$, *any set* $S \subseteq [n]$ *with complement* $\bar{S}$, *and any* $z \in X^{\bar{S}}$, *the* restriction of f to S using z *is the function*

$$f_{S|z} : X^S \to Y$$

defined by fixing the coordinates in $\bar{S}$ *to the value* z. *More formally, for any* $x \in X^S$, *we define*

$$f_{S|z}(x) \stackrel{\mathsf{def}}{=} f(x'),$$

where for each $i \in [n]$,

$$x'_i = \begin{cases} x_i & \text{if } i \in S \\ z_i & \text{otherwise.} \end{cases}$$

3 Lower Bounds on DRE Size

Over 50 years ago, Nečiporuk published a two-page note titled "On a boolean function." [49] Within these two pages, Nečiporuk introduced an elegant combinatorial measure of a function related to the number of ways a function can be restricted distinctly. To this day, Nečiporuk's method still provides the strongest lower bounds known for formulas over arbitrary finite bases, deterministic branching programs, non-deterministic branching programs, parity branching programs, switching networks, span programs, and more [16].

In this section we recall Nečiporuk's measure and add decomposable randomized encoding (DRE) size to the list of complexity measures that are lower bounded by Nečiporuk's measure. Specifically, we show that for any function f, the DRE complexity of f is at least Nečiporuk's measure (which for explicit functions is as large as $n^2/\log n$). Prior to this work no super linear lower bounds on DRE size were known.

3.1 Technical Overview

To lower bound the DRE size of a function $f : \{0,1\}^n \to \{0,1\}$, we first consider all possible restrictions of f, using notation as in Definition 7. For simplicity, suppose that

$$\mathcal{X} = \begin{pmatrix} \mathcal{X}^1_0, \ldots, \mathcal{X}^n_0 \\ \mathcal{X}^1_1, \ldots, \mathcal{X}^n_1 \end{pmatrix}$$

is a *perfect* DRE for f. Then for all $S \subseteq [n]$ (with complement denoted by $\bar{S}$), we observe that:

1. The distribution of $(\mathcal{X}^i_{z_i})_{i\in\bar{S}}$ does not depend on $z \in \{0,1\}^{\bar{S}}$ (as long as $f_{S|z}$ is non-constant). This follows from DRE security.
2. Given $(\mathcal{X}^i_{z_i})_{i\in\bar{S}}$, the values $(X^i_b)_{i\in S, b\in\{0,1\}}$ are sufficient to reconstruct the truth table of $f_{S|z}$. This follows from DRE correctness.

Together, these properties imply that the size of the support of $(X^i_b)_{i\in S,b\in\{0,1\}}$ is at least the number of non-constant truth tables of the form $f_{S|z}$ for some $z \in \{0,1\}^{\bar{S}}$. We obtain a bound on the size of $\mathcal{X}$ by partitioning $[n]$ into sets $S_1,\ldots,S_m$, and lower bounding the size of each $(\mathcal{X}^i_b)_{i\in S_j,b\in\{0,1\}}$. The maximum bound on the *bit length* of $\mathcal{X}$ that can be achieved in this way is essentially Nečiporuk's measure of f.

We elaborate further below, defining a somewhat more general computational analogue of Nečiporuk's measure (that will suffice for lower bounds on computationally secure DREs).

3.2 Nečiporuk's Measure

Let $f : \{0,1\}^n \to \{0,1\}$ be any boolean function. For any subset $S \subseteq [n]$, let $\bar{S}$ denote $[n] \setminus S$, and define

$$g_S(f) \stackrel{\text{def}}{=} \log(\#\{f_{S|z} : z \in \{0,1\}^{\bar{S}}\}).$$

Let $V = (V_1,\ldots,V_m)$ denote a partition of $[n]$. That is, $V_1,\ldots,V_m$ are pairwise disjoint subsets of $[n]$ whose union is $[n]$. Then, the Nečiporuk measure of f is

$$G(f) \stackrel{\text{def}}{=} \max_V \sum_{V_i\in V} g_{V_i}(f).$$

▶ Remark 8. It is well known that for any function f, $G(f) \le n^2/\log n$ [57].

3.3 Functions with Maximal Measure

We recall several functions whose Nečiporuk measures are known to be as high as possible ($\Omega(n^2/\log n)$, where n is the bit-length of the input).

Element Distinctness

Element Distinctness is a function $\mathrm{ED}_m : [m^2]^m \to \{0,1\}$ which given a vector $(x_1,\ldots,x_m) \in [m^2]^m$ and outputs 1 if all x_i are distinct and 0 otherwise ($\exists i \neq j$ such that $x_i = x_j$).

Others

Clique, matching, and determinant all have measure $\Omega(n^2/\log n)$ [57].

Random

Finally, and perhaps unsurprisingly, we note that a random function has measure at least $\frac{n(n-2)}{\log n}$) with overwhelming probability (for n large enough). See Appendix B for proof.

3.4 DRE Size Lower Bounds via Nečiporuk

We define a pseudo-min-entropic analogue of Nečiporuk's measure, with an additional non-constantness restriction that is tailored for use in DRE lower bounds.

▶ **Definition 9.** *The* (s,ϵ)-pseudo min-entropy *of a random variable* X*, which we will denote by* $\tilde{H}_\infty^{(s,\epsilon)}(X)$*, is the supremum of* $H_\infty(\tilde{X})$ *over all random variables* $\tilde{X}$ *that are* (s,ϵ)*-indistinguishable from* X*.*

▶ **Definition 10.** *For any function* $f:\{0,1\}^n\to\{0,1\}$ *and any subset* $\emptyset\neq V\subseteq[n]$*, define*

$$\tilde{G}_V^{(s,\epsilon)}(f) \stackrel{\mathsf{def}}{=} \sup\left(\tilde{H}_\infty^{(s,\epsilon)}(f_{V|Z})\right),$$

where the supremum is taken over all $\{0,1\}^{\bar{V}}$*-valued random variables* Z *whose support only consists of values* z *that make* $f_{V|z}$ *non-constant.*

We define $\tilde{G}^{(s,\epsilon)}(f)$ *to be the maximum over all partitions* $[n]=V_1\cup\cdots\cup V_m$ *of* $\sum_{i\in[m]}\tilde{G}_{V_i}^{(s,\epsilon)}(f)$*.*

▶ Remark 11. If not for the non-constantness constraint on $f_{V|Z}$, the measure $\tilde{G}^{(\infty,0)}$ is the same as Nečiporuk's original measure. Reducing s or increasing ϵ only increases this measure. Taking the non-constantness restriction into account, our measure cannot be smaller than Nečiporuk's measure by more than $O(n)$ (so superlinear lower bounds on Nečiporuk's measure imply an asymptotically identical lower bound on our measure).

Beyond a certain threshold, increasing s no longer changes the value of $\tilde{G}^{(s,\epsilon)}$:

▷ Claim 12. For any function $f:\{0,1\}^n\to\{0,1\}$ and any subset $V\subseteq[n]$, we have $\tilde{G}_V^{(\infty,\epsilon)}(f)=\tilde{G}_V^{\left(2^{2^{|V|}},\epsilon\right)}(f)$.

Proof. Any function of n bits can be computed by a circuit of size 2^n. In fact this can be strengthened to $O(\frac{2^n}{n})$ [59, 45], but we prefer the simpler bound 2^n. Apply this to the (s,ϵ)-indistinguishability in the definition of pseudo-min-entropy of $f_{V|Z}$ (which is a truth table of bit length $n=2^{|V|}$). ◁

Our main lower bound is given by the following theorem.

▶ **Theorem 13.** *Let* $f:\{0,1\}^n\to\{0,1\}$ *be a function, and let* $\mathcal{X}$ *be a* $(s^*_{\mathsf{DRE}},\frac{1}{3})$*-secure DRE for* f *with a decoding algorithm of size* s_{Dec}*.*

Then for all $V\subseteq[n]$*, we have*

$$|\mathcal{X}^V|\geq\min\left(\log_2\left(\frac{s^*_{\mathsf{DRE}}}{s_{\mathsf{Dec}}\cdot 2^{|V|}}\right),\tilde{G}_V^{(s^*_{\mathsf{DRE}},\frac{1}{3})}(f)-2\right).$$

Proof. Suppose otherwise – that $|\mathcal{X}^V|<\log_2\left(\frac{s^*_{\mathsf{DRE}}}{s_{\mathsf{Dec}}\cdot 2^{|V|}}\right)$ and $|\mathcal{X}^V|<\tilde{G}_V^{(s^*_{\mathsf{DRE}},\frac{1}{3})}(f)-2$.

Let Z be a $\{0,1\}^{\bar{V}}$-valued random variable that maximizes $\tilde{H}_\infty^{(s^*_{\mathsf{DRE}},\frac{1}{3})}(f_{V|Z})$, supported by values z for which $f_{V|z}$ is non-constant, and let $\tilde{F}_V$ denote a random variable that is $(s^*_{\mathsf{DRE}},\frac{1}{3})$-indistinguishable from $f_{V|Z}$ and satisfies $H_\infty(\tilde{F}_V)=\tilde{H}_\infty^{(s^*_{\mathsf{DRE}},\frac{1}{3})}(f_{V|Z})$. Let Z' be an independent copy of Z.

We first claim that $(\mathcal{X}_Z^{\bar{V}},f_{V|Z})\approx^{(s^*_{\mathsf{DRE}},\frac{1}{3})}(\mathcal{X}_{Z'}^{\bar{V}},f_{V|Z})$. To see why, suppose for contradiction that there is size-s^*_{DRE} circuit $\mathcal{A}$ that distinguishes $(\mathcal{X}_Z^{\bar{V}},f_{V|Z})$ from $(\mathcal{X}_{Z'}^{\bar{V}},f_{V|Z})$ with advantage better than $\frac{1}{3}$. Then in particular there exist $z,z'\in\{0,1\}^{\bar{V}}$ such that $\mathcal{A}$ distinguishes $(\mathcal{X}_z^{\bar{V}},f_{V|z})$ from $(\mathcal{X}_{z'}^{\bar{V}},f_{V|z})$ with the same advantage. Hardwiring $f_{V|z}$ into $\mathcal{A}$, this gives a circuit $\mathcal{B}$ of size[5] $|\mathcal{B}|\leq|\mathcal{A}|$ for distinguishing $\mathcal{X}_z^{\bar{V}}$ from $\mathcal{X}_{z'}^{\bar{V}}$ with the same advantage. But this contradicts the $(s^*_{\mathsf{DRE}},\frac{1}{3})$-indistinguishability that is guaranteed by DRE security.

[5] Recall that the size of a circuit is measured in number of gates, and all gates of $\mathcal{A}$ whose inputs are the hard-wired value $f_{V|z}$ can be simplified or eliminated.

We also know that $(\mathcal{X}_{Z'}^{\bar{V}}, f_{V|Z}) \approx^{(s^*_{\mathsf{DRE}}, \frac{1}{3})} (\mathcal{X}_{Z'}^{\bar{V}}, \tilde{F}_V)$, so together with the previous claim, we have $(\mathcal{X}_{Z}^{\bar{V}}, f_{V|Z}) \approx^{(s^*_{\mathsf{DRE}}, \frac{2}{3})} (\mathcal{X}_{Z'}^{\bar{V}}, \tilde{F}_V)$. However, there is a distinguisher that contradicts this. Specifically, try all possible values of $\left(\mathcal{X}_b^i\right)_{i\in V, b\in\{0,1\}}$ (there are at most $\frac{s^*_{\mathsf{DRE}}}{s_{\mathsf{Dec}}\cdot 2^{|V|}}$ possibilities), and apply the DRE decoding algorithm ($2^{|V|}$ times per possibility) to see whether any possibility "explains" the given truth table.

By correctness of the DRE, there will always exist a value that explains $f_{V|Z}$ given $\mathcal{X}_Z^{\bar{V}}$, but because $H_\infty(\tilde{F}_V) > \log_2 |\mathcal{X}^V| + 2$, the probability that any value explains $\tilde{F}_V$ is at most $\frac{1}{4}$. Hence the distinguisher succeeds with probability $\frac{3}{4} > \frac{2}{3}$, which is a contradiction. ◀

3.5 The Nečiporuk Measure of PRFs

In this section, we prove lower bounds on the Nečiporuk measure of PRFs (of varying security levels), which imply corresponding lower bounds on the size of DREs.

▶ **Proposition 14.** *If $E : \{0,1\}^{\kappa+n} \to \{0,1\}$ is an (s,ϵ)-secure PRF with key length κ and input length n satisfying $s \geq 4$ and $\epsilon \leq \frac{1}{6}$, then for any subset $V \subseteq [\kappa+1, \kappa+n]$ with $|V| \geq 2$, we have $\tilde{G}_V^{(s,\epsilon')}(E) = 2^{|V|}$ for $\epsilon' = 3\epsilon + 2^{-s+1} + 2^{-2^{|V|}+1}$.*

Proof. Let Z' be a $\{0,1\}^{\bar{V}}$-valued random variable whose first κ coordinates are independent and uniformly random, and the rest of whose coordinates are 0. By PRF security, the probability that $E_{V|Z'}$ is constant is at most $\delta \stackrel{\mathsf{def}}{=} \epsilon + 2^{-\min(s,2^{|V|})+1} \leq \epsilon + 2^{-s+1} + 2^{-2^{|V|}+1} \leq \frac{1}{2}$.

Let $\mathcal{A}$ be an arbitrary size-s circuit. Suppose for contradiction that $\mathcal{A}$ distinguishes $E_{V|Z'}$ from a uniformly random truth table with advantage greater than ϵ. Then each input wire of $\mathcal{A}$ can be replaced by an oracle gate to yield a circuit that distinguishes oracle access to $E(K,\cdot)$ (for uniform K) from oracle access to a uniformly random function with the same advantage ϵ. This contradicts (s,ϵ)-security of the PRF. So $E_{V|Z'}$ is (s,ϵ)-indistinguishable from a uniformly random truth table.

Conditioned on $E_{V|Z'}$ being non-constant, the advantage of any $\mathcal{A}$ in distinguishing $E_{V|Z'}$ from a uniformly random truth table can increase to at most

$$\frac{\frac{1}{2}+\epsilon}{1-\delta} - \frac{1}{2} \leq (\frac{1}{2}+\epsilon)\cdot(1+2\delta) - \frac{1}{2} = \epsilon + \delta + 2\epsilon\cdot\delta \leq 3\epsilon + 2^{-s+1} + 2^{-2^{|V|}+1}.$$

Thus if Z denotes the random variable Z' conditioned on $E_{V|Z'}$ being non-constant, we have $\tilde{H}^{(s,\epsilon')}(E_{V|Z}) = 2^{|V|}$ for $\epsilon' = 3\epsilon + 2^{-s+1} + 2^{-2^{|V|}+1}$. ◀

▶ **Corollary 15.** *If $E : \{0,1\}^{\kappa+n} \to \{0,1\}$ is the evaluation algorithm for an (s,ϵ)-secure PRF family with key length κ and input length n satisfying $s \geq 4$ and $\epsilon \leq \frac{1}{6}$, then $\tilde{G}^{(\infty,\epsilon')}(E) \geq \Omega\left(\frac{n\log s}{\log\log s}\right)$ for $\epsilon' = 3\epsilon + 2^{-s+2}$. In particular, if the PRF family is exponentially secure, then $\tilde{G}^{(\infty,\epsilon')}(E) \geq \Omega\left(\frac{n^2}{\log n}\right)$.*

Proof. For every $V \subseteq [\kappa+1, n]$ of size $|V| = \log\log s$, Proposition 14 implies that there exists a random variable Z such that $\tilde{H}^{(s,\epsilon')}(E_{V|Z}) = 2^{|V|} = \log s$ for $\epsilon' = 3\epsilon + 2^{-s+2}$. But by Claim 12, $\tilde{H}^{(s,\epsilon')}(E_{V|Z}) = \tilde{H}^{(\infty,\epsilon')}(E_{V|Z})$.

The lower bound on $\tilde{G}^{(\infty,\epsilon')}(E)$ follows by partitioning $[\kappa+n]$ into $V_0 \cup V_1 \cup \cdots \cup V_{n/\log\log s}$, where $V_0 = [\kappa]$ and each V_i has size $|V_i| = \log\log s$ for $1 \leq i \leq n/\log\log s$. ◀

▶ Remark 16. We obtain a similar result to Corollary 15 in Appendix A that applies to uniformly secure PRFs.

▶ **Corollary 17.** *If E is the evaluation algorithm for an exponentially secure PRF family with input length n, then any statistically secure DRE for E has size at least $\Omega\left(\frac{n^2}{\log n}\right)$.*

3.6 A Truly Quadratic Lower Bound

We observe that for exponentially secure PRFs with n-bit output, even computationally secure DREs require size $\Omega(n^2)$.

▶ **Theorem 18.** *Any computational DRE of an exponentially-secure PRF with n-bits of output must have size $\Omega(n^2)$.*

To prove this theorem we will rely on the following result of Applebaum et al. [9].

▶ **Theorem 19.** *Let $S(k, x, r)$ be a one-time MAC with key k, message x, and randomness r. Let $\ell(n)$ denote the input locality of $S_k(x, r)$ and let $s(n)$ denote the length of a tag, where n is the security parameter. (A function has input locality ℓ if no input bit affects more than ℓ output bits.) Then, there is an efficient attack on $S(k, x, r)$ that succeeds with probability $1/\binom{s(n)}{\ell(n)} \cdot 2^{-\ell(n)}$.*

Proof. Recall that an exponentially secure PRF $f_k : \{0,1\}^n \to \{0,1\}^n$ is also an exponentially secure one-time MAC [42]. Moreover, a DRE of a MAC preserves unforgeability [9]. Because $1/\binom{n}{\ell(n)} \cdot 2^{-\ell(n)} \leq 2^{\ell(n)}$, it follows Theorem 19 that any DRE of an exponentially-secure f_k must have input locality $\Omega(n)$. By decomposability, any such DRE must have size $\Omega(n^2)$. ◀

4 Upper Bounds on DRE Size

In this section we present nearly matching upper bounds for some of the explicit functions to which our lower bounds apply. We explicitly conjecture two variants of the "hidden shift problem" are exponentially secure PRFs and show that they admit nearly quadratic size (efficient, perfect) DREs. Finally, we show a recent *weak* PRF candidate due to Boneh et al. [19], conjectured to be nearly exponentially-secure, admits a linear-size (efficient and perfect) DRE

4.1 Almost Tight Quadratic Upper Bounds

Partial Decomposability

We introduce the notation of a *partially decomposable randomized encoding*, so that later we can construction DRE by composing a randomized encoding and a partially decomposable randomized encoding. A randomized encoding $(\mathsf{Enc}, \mathsf{Dec})$ for a function $f : \{0,1\}^n \times \mathcal{W} \to \mathcal{Y}$ is a *partially decomposable randomized encoding (PDRE)* if every bit of $\mathsf{Enc}(x, w, r)$ is determined by $w \in \mathcal{W}, r \in \mathcal{R}$ and only 1 bit of $x \in \{0,1\}^n$.

▶ **Lemma 20** (Composition of randomized encodings)**.** *Let $\mathsf{Enc} : \{0,1\}^n \times \mathcal{W} \to \{0,1\}^\ell$ be (the encoding function of) a randomized encoding $(\mathsf{Enc}, \mathsf{Dec})$ for function $f : \{0,1\}^n \to \mathcal{Y}$. Let $\mathsf{Enc}' : (\{0,1\}^n \times \mathcal{W}) \times \mathcal{R} \to \{0,1\}^{\ell'}$ be the encoding function of a PDRE $(\mathsf{Enc}', \mathsf{Dec}')$ for function Enc. Then $\mathsf{Enc}' : \{0,1\}^n \times (\mathcal{W} \times \mathcal{R}) \to \{0,1\}^{\ell'}$ is the encoding function of a DRE for function f.*

Proof. The corresponding decoding function is $\mathsf{Dec}''(c) := \mathsf{Dec}(\mathsf{Dec}'(c))$. It's easy to verify $(\mathsf{Enc}', \mathsf{Dec}'')$ is a DRE, as each bit of $\mathsf{Enc}'(x, r, w)$ is determined by (r, w) and only 1 bit of x.

A DRE for Element Distinctness

Choose an $O(\log n)$-bit prime p with $p > \binom{n}{2}$. For all $1 \le i < i' \le n$, define indicator $\delta_{i,i'} \in \{0,1\}$ that captures whether $x_i = x_{i'}$,

$$\delta_{i,i'} := \begin{cases} 1, & \text{if } x_i = x_{i'}, \\ 0, & \text{if } x_i \neq x_{i'}. \end{cases}$$

Sample $a \leftarrow \mathbb{Z}_p \setminus \{0\}$ for the CRS. For all $1 \le i < i' \le n$, sample random $r_{i,i'} \in \mathbb{Z}_p$ from CRS such that $\sum_{1 \le i < i' \le n} r_{i,i'} = 0$. Define $\hat{r}_{i,i'} \in \mathbb{Z}_p$ as $\hat{r}_{i,i'} := r_{i,i'} + a \cdot \delta_{i,i'}$.

Then a DRE for element distinctness is induced by composing the following two claims:

▷ Claim 21. $(\hat{r}_{i,i'})_{1 \le i < i' \le n}$ is a randomized encoding of the functionality output.

Proof. It's obvious that $(\hat{r}_{i,i'})_{1 \le i < i' \le n}$ is a randomized encoding of $a \cdot \sum_{1 \le i < i' \le n} \delta_{i,i'}$. The later is a randomized encoding of the functionality output because: when $(x_i)_{1 \le i \le n}$ are all distinct, $a \cdot \sum_{1 \le i < i' \le n} \delta_{i,i'}$ is zero; when there is a collision, $a \cdot \sum_{1 \le i < i' \le n} \delta_{i,i'}$ is uniformly random in $\mathbb{Z}_p \setminus \{0\}$. ◁

▷ Claim 22. For all $1 \le i < i' \le n$, there exists a PDRE for $\hat{r}_{i,i'}$ of size $O(\log^4 n)$.

Proof. For any $v \in \mathbb{Z}_p$, let $v[k]$ denote the k-th bit of its binary representation. Then the k-th bit of $r_{i,i'}$ can be computed from

$$\begin{aligned} \hat{r}_{i,i'}[k] &= \begin{cases} r_{i,i'}[k], & \text{if } \delta_{i,i'} = 0 \\ (r_{i,i'} + a)[k], & \text{if } \delta_{i,i'} = 1 \end{cases} \\ &= r_{i,i'}[k] \oplus (r_{i,i'}[k] \oplus (r_{i,i'} + a)[k]) \cdot \bigvee_{j=1}^{\log p} (x_i[j] \oplus x_{i'}[j]), \end{aligned}$$

which, as a function of $(x_i, x_{i'})$, is a binary branching program of size $O(\log n)$. Thus there is a PDRE for $\hat{r}_{i,i'}$ of size $O(\log^3 n)$ [8][6]. As $\hat{r}_{i,i'}$ has $\log n$ bits, there exists a PDRE for $\hat{r}_{i,i'}$ of size $O(\log^4 n)$. ◁

4.2 A PRF Candidate With A Nearly Optimal DRE

Now we can present the almost-optimally-garble-able candidate PRF. Modulo a conjecture on its hardness, this simple algebraic PRF candidate admits a (perfect) DRE of size at most a $\log n$ factor from the minimum. Moreover, a simple generalization of this candidate yields linear output length with the same DRE complexity. Thus, if this candidate is exponentially secure, it is indeed optimally-garble-able.

In addition to applications in efficient MPC, this candidate can be conversely interpreted through Razborov and Rudich's natural proof framework as barrier to proving super quadratic bounds on DRE size [53].

[6] For a branching program of size s and has t input bits, there is a DRE for the branching program of size $s^2 t$.

An Exponentially-Secure PRF Candidate

Our starting point is an algebraic object that has received considerable attention in both cryptography and mathematics: Legendre sequences. A Legendre sequence is a sequence of the form:

$$(x+1)^{(p-1)/2}, (x+2)^{(p-1)/2}, (x+3)^{(p-1)/2}, \ldots$$

where all operations are over $\mathbb{Z}_p$ for some prime p.

The pseudorandomness of sequences of quadratic characters have a long history in both cryptography and mathematics [4, 20, 24, 26, 38, 46, 47, 52, 55]. These sequences have been shown to behave as if random with respect to a variety of statistical tests designed for randomness.

Recent work has considered the so-called "hidden shift problems" and their generalizations. In the quadratic character variant of the hidden shift problem, algorithms are given oracle access to a function $\phi_k : \mathbb{Z}_p \to \{-1, 0, +1\}$ where $\phi_k(x) = (k+x)^{(p-1)/2}$ from some $k \in \mathbb{Z}_p$. The task is then to recover k. Efficient quantum algorithms for this problem are known [61, 62, 63, 56, 41]. However, the best classical algorithms to date are still just subexponentional (under an assumption on the density of smooth integers) [20, 56, 43]. Indeed, Dam, Hallgren, and Ip [63] have explicitly conjectured that ϕ_k is a PRF with respect to polytime classical algorithms. Grassi et al. [37] additionally proposed this function specifically as an "MPC-friendly" PRF. Recently, cryptanalytic bounties have been announced on this PRF [31].

With the known attacks in mind, we give a twist on the hidden shift problem restricting evaluation to a short interval. So far as we know this confounds all existing techniques (including quantum algorithms) and the best algorithm[7] runs in $2^{(1+o(1))n}$-time [60].

We actually make two conjectures: (1) restricted hidden shift yields an exponentially-secure PRF with one bit of output, (2) a natural generalization is an exponentially-secure PRF with many bits of output. But first, we define the restricted hidden shift function.

For any $m \in \mathbb{Z}^+$, let $p \equiv 1 \pmod m$ be a prime with $p \geq 2^{2n}$, and let $\langle \zeta_m \rangle$ denote the group of m^{th} roots of unity in $\mathbb{Z}_p^\times$. For $k \in \mathbb{Z}_p$ define

$$\begin{aligned} \mathsf{Char}_k^{p,m,n} &: [0, 2^n - 1] \to \langle \zeta_m \rangle \\ \mathsf{Char}_k^{p,m,n} &: x \mapsto (k+x)^{\frac{p-1}{m}} \pmod p. \end{aligned}$$

Note that $\mathsf{Char}_k^{p,2,n}(x) = 0$ for $k + x = p$. In order to achieve single bit output (just two possible output values) we restrict the key space in addition to the input space, so that this equation cannot be satisfied.

▶ **Conjecture 23.** *Let $p = p(n)$ be any prime sequence satisfying $p \equiv 1 \pmod m$, $p > 2^{n+1}$. Then, $\left\{ \left\{ \mathsf{Char}_k^{p,2,n} \right\}_{k \in \{1,\ldots,2^n\}} \right\}_{n \in \mathbb{Z}^+}$ is, for some $s(n) = 2^{\Omega(n)}$, an $(s(\cdot), s(\cdot)^{-1})$-secure PRF family.*

Next, we present a variant with long output by applying an input restriction to the "hidden power problem" [21] or "hidden root problem" [64]. In this problem, the goal is to recover k using query access to $x \mapsto (k+x)^e$ for more general $e | p-1$ (the shift problem

[7] The algorithm is to simply guess k and test on enough x. However it is worth noting that even this is not known to work, and requires making a conjecture on the distribution of Legendre sequences generated by random k [60]. The best *provable* distinguisher that we know of runs in time $2^{(3/2+o(1))n}$-time by simply exhaustively enumerate all sequences of length $2^{n/2}$ and comparing [60]

discussed above is simply the specific case of $e = \frac{p-1}{2}$). Notably, [21] demonstrated (classical) algorithms for this problem that make $O(1)$ queries and recover k in time $e^{1+\epsilon}\log^{O(1)} p$. With this in mind, we make the following conjecture.

▶ **Conjecture 24.** *Let $p = p(n)$ be any prime sequence and $m = m(n)$ be any positive integer sequence satisfying $p \equiv 1 \pmod m$, $p \geq 2^{2n}$, and $\frac{p}{m} \geq 2^n$. Then $\left\{\left\{\mathsf{Char}_k^{p,m,n}\right\}_{k\in\mathbb{Z}_p}\right\}_{n\in\mathbb{Z}^+}$ is, for some $s(n) = 2^{\Omega(n)}$, an $(s(\cdot), s(\cdot)^{-1})$-secure PRF family.*

An $O(n^2)$ DRE for the Candidate PRF

We now show that there is a DRE for $\mathsf{Char}_{(\cdot)}^{n,m,p}(\cdot)$ of size $O(n^2)$. Assuming the above conjectures, it follows from Corollary 32 that this DRE has essentially optimal size, not just for $\mathsf{Char}_{(\cdot)}^{n,m,p}(\cdot)$, but among DREs for *any* exponentially-secure PRF.

For clarity, we present a DRE for $\mathsf{Char}_k^{p,2,n}$ and note that the construction can easily be extended to the multi-bit output case.

Our starting point is a simple perfect randomized encoding for quadratic residue[8]:

$$\mathsf{Enc} : x \mapsto x \cdot r^2, \text{ for uniformly sampled } r \leftarrow \mathbb{Z}_p$$
$$\mathsf{Dec} : y \mapsto y^{(p-1)/2}$$

Security follows from the fact that any quadratic residue is mapped to a uniformly random quadratic residue, and any non-residue is mapped to a uniformly random non-residue. Note that this randomized encoding has size $O(n)$.

However, we would like a randomized encoding of the quadratic residuosity of $x + k$ and moreover we would like it to be decomposable. This is easily remedied via bit decomposition and the fact that the above encoding is linear with respect to the input.

$$\begin{aligned}\mathsf{Enc} : x_i &\mapsto x_i \cdot 2^{i-1} \cdot r^2 + s_i \\ k_i &\mapsto k_i \cdot 2^{i-1} \cdot r^2 + t_i \\ &\text{where } r, s_1, \ldots, s_n, t_1, \ldots, t_{2n+1} \text{ are drawn uniformly from } \mathbb{Z}_p \\ &\text{such that } s_1 + \cdots + s_n + t_1 + \cdots + t_{2n+1} = 0.\end{aligned}$$
$$\mathsf{Dec} : y_1, \ldots, y_{3n+1} \mapsto \left(\sum y_i\right)^{(p-1)/2}$$

Similarly, correctness and security follow from the fact that an encoding is simply $3n + 1$ random elements, conditioned on the fact that their sum is a random element with the quadratic residuosity of x. Note that because the encoding consists of $3n + 1$ elements, each of bit length $2n + 1$, the size of this DRE is $O(n^2)$.

4.3 A WPRF Candidate With A Nearly Optimal DRE

In this section, we observe that a recent weak pseudorandom function candidate put forward by Boneh et al. admits a DRE of quasi-linear size [19].

Boneh et al. [19] have put forward the following WPRF candidate related to both the learning parity with noise problem (with "deterministic" noise) and learning with rounding problem (over constant-size modulus). Given a key $k \in \{0,1\}^n$, they define

[8] A similar randomization technique for quadratic characters was previously used in related contexts in [30, 5, 1, 37].

Protocol Let S_6 and f act on $\mathbb{Z}_6$ on the right in the natural way. Let $\sigma \in S_6$ denote the permutation that maps x to $x+1$. Let $L = 0 \in \mathbb{Z}_6$.

Randomness

- Sample $r_1, \ldots, r_{n-1} \leftarrow S_6$.
- Define $(R_{1,0}, R_{1,1}) = (L \cdot r_1, L \cdot \sigma \cdot r_1)$
- For $2 \le i \le n-1$, define $(R_{i,0}, R_{i,1}) = (r_{i-1}^{-1} \cdot r_i, r_{i-1}^{-1} \cdot \sigma \cdot r_i)$
- Define $(R_{n,0}, R_{n,1}) = (r_{n-1}^{-1} \cdot f, r_{n-1}^{-1} \cdot \sigma \cdot f)$

Encoding For $1 \le i \le n$, $\mathsf{Enc}_i(z_i, R_i) = M_i = R_{i,z_i}$

Decoding $M_1 \cdots M_n$

Figure 1 A DRE for a function of a sum mod 6 [17].

$$\mathsf{LWR}_k^6 : \{0,1\}^n \to \{0,1\}$$
$$\mathsf{LWR}_k^6(x) = \begin{cases} 0 & \text{if } \langle x, k\rangle \equiv 0,\ 1,\ \text{or } 2 \pmod 6 \\ 1 & \text{if } \langle x, k\rangle \equiv 3,\ 4,\ \text{or } 5 \pmod 6. \end{cases}$$

This candidate was proposed with efficient secure function evaluation protocols in mind; however, the protocol presented in [19] requires two phases of interaction: first it applies a DRE-based subprotocol for computing shares of the mod-6 inner product, and then another subprotocol for rounding. Here we show that LWR^6 has a DRE of size $O(n)$.[9]

Let $\lfloor \cdot \rceil : \mathbb{Z}_6 \to \{0,1\}$ denote the function

$$\lfloor x \rceil = \begin{cases} 0 & \text{if } x \in \{0,1,2\} \\ 1 & \text{otherwise.} \end{cases}$$

We obtain our DRE for LWR_k^6 by composing two DREs ([8, 11]); the first is for a function that maps $(z_1, \ldots, z_n) \mapsto \left\lfloor \sum_i z_i \pmod 6 \right\rceil$ for $z_1, \ldots, z_n \in \{0,1\}$, and the second is for the AND function mapping $(k_i, x_i) \in \{0,1\}^2$ to $k_i \cdot x_i$.

The DRE for the first function is obtained as a special case of a result on symmetric functions due to Beimel et al. [17, Theorem 7.2, Figure 9] that refines a group-based DRE due to Kilian [44]:

▶ **Imported Theorem 25** ([17]). *For any function $f : \mathbb{Z}_6 \to \{0,1\}$, the scheme of Figure 1 is a size-$O(n)$ DRE of the function h that maps $(z_1, \ldots, z_n) \mapsto f\big(\sum z_i \pmod 6\big)$.*

The second function is constant-sized, and thus has a constant-sized DRE by Barrington's theorem [14] and Kilian's rerandomization.

References

1 Shweta Agrawal, Yuval Ishai, Dakshita Khurana, and Anat Paskin-Cherniavsky. Statistical Randomized Encodings: A Complexity Theoretic View. In Magnús M. Halldórsson, Kazuo Iwama, Naoki Kobayashi, and Bettina Speckmann, editors, *Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part I*, volume 9134 of *Lecture Notes in Computer Science*, pages 1–13. Springer, 2015. doi:10.1007/978-3-662-47672-7_1.

[9] In contrast to the PRF candidate proposed above, this WPRF candidate is at most $2^{n/\log n}$-secure. Assuming it is indeed $2^{n/\log n}$ secure, an $O(\lambda \log \lambda)$ size DRE is needed to get 2^λ security.

2 Martin R. Albrecht, Lorenzo Grassi, Léo Perrin, Sebastian Ramacher, Christian Rechberger, Dragos Rotaru, Arnab Roy, and Markus Schofnegger. Feistel Structures for MPC, and More. *IACR Cryptology ePrint Archive*, 2019:397, 2019. URL: https://eprint.iacr.org/2019/397.

3 Martin R. Albrecht, Christian Rechberger, Thomas Schneider, Tyge Tiessen, and Michael Zohner. Ciphers for MPC and FHE. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 430–454. Springer, 2015. doi:10.1007/978-3-662-46800-5_17.

4 Michael Anshel and Dorian Goldfeld. Zeta functions, one-way functions, and pseudorandom number generators. *Duke Math. J.*, 88(2):371–390, June 1997. doi:10.1215/S0012-7094-97-08815-3.

5 Benny Applebaum. Key-Dependent Message Security: Generic Amplification and Completeness. *J. Cryptology*, 27(3):429–451, 2014. doi:10.1007/s00145-013-9149-6.

6 Benny Applebaum. Garbled Circuits as Randomized Encodings of Functions: a Primer. In Yehuda Lindell, editor, *Tutorials on the Foundations of Cryptography*, pages 1–44. Springer International Publishing, 2017. doi:10.1007/978-3-319-57048-8_1.

7 Benny Applebaum, Thomas Holenstein, Manoj Mishra, and Ofer Shayevitz. The Communication Complexity of Private Simultaneous Messages, Revisited. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part II*, volume 10821 of *Lecture Notes in Computer Science*, pages 261–286. Springer, 2018. doi:10.1007/978-3-319-78375-8_9.

8 Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography in NC^0. *SIAM J. Comput.*, 36(4):845–888, 2006. doi:10.1137/S0097539705446950.

9 Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography with Constant Input Locality. *J. Cryptology*, 22(4):429–469, 2009. doi:10.1007/s00145-009-9039-0.

10 Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. From Secrecy to Soundness: Efficient Verification via Secure Computation. In Samson Abramsky, Cyril Gavoille, Claude Kirchner, Friedhelm Meyer auf der Heide, and Paul G. Spirakis, editors, *Automata, Languages and Programming, 37th International Colloquium, ICALP 2010, Bordeaux, France, July 6-10, 2010, Proceedings, Part I*, volume 6198 of *Lecture Notes in Computer Science*, pages 152–163. Springer, 2010. doi:10.1007/978-3-642-14165-2_14.

11 Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. How to Garble Arithmetic Circuits. *SIAM J. Comput.*, 43(2):905–929, 2014.

12 Marshall Ball, Brent Carmer, Tal Malkin, Mike Rosulek, and Nichole Schimanski. Garbled Neural Networks are Practical. *IACR Cryptology ePrint Archive*, 2019:338, 2019. URL: https://eprint.iacr.org/2019/338.

13 Boaz Barak, Iftach Haitner, Dennis Hofheinz, and Yuval Ishai. Bounded Key-Dependent Message Security. In Henri Gilbert, editor, *Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Monaco / French Riviera, May 30 - June 3, 2010. Proceedings*, volume 6110 of *Lecture Notes in Computer Science*, pages 423–444. Springer, 2010. doi:10.1007/978-3-642-13190-5_22.

14 David Arno Barrington. *Width-3 permutation branching programs*. Laboratory for Computer Science, Massachusetts Institute of Technology, 1985.

15 Tugkan Batu, Sanjoy Dasgupta, Ravi Kumar, and Ronitt Rubinfeld. The Complexity of Approximating the Entropy. *SIAM J. Comput.*, 35(1):132–150, 2005.

16 Paul Beame, Nathan Grosshans, Pierre McKenzie, and Luc Segoufin. Nondeterminism and An Abstract Formulation of Nečiporuk's Lower Bound Method. *TOCT*, 9(1):5:1–5:34, 2016.

17 Amos Beimel, Ariel Gabizon, Yuval Ishai, Eyal Kushilevitz, Sigurd Meldgaard, and Anat Paskin-Cherniavsky. Non-Interactive Secure Multiparty Computation. In *CRYPTO (2)*, volume 8617 of *Lecture Notes in Computer Science*, pages 387–404. Springer, 2014.

18 Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Foundations of garbled circuits. In Ting Yu, George Danezis, and Virgil D. Gligor, editors, *the ACM Conference on Computer and Communications Security, CCS'12, Raleigh, NC, USA, October 16-18, 2012*, pages 784–796. ACM, 2012. doi:10.1145/2382196.2382279.

19 Dan Boneh, Yuval Ishai, Alain Passelègue, Amit Sahai, and David J. Wu. Exploring Crypto Dark Matter: - New Simple PRF Candidates and Their Applications. In *TCC (2)*, volume 11240 of *Lecture Notes in Computer Science*, pages 699–729. Springer, 2018.

20 Dan Boneh and Richard J. Lipton. Algorithms for Black-Box Fields and their Application to Cryptography. In Neal Koblitz, editor, *Advances in Cryptology — CRYPTO '96*, pages 283–297, Berlin, Heidelberg, 1996. Springer Berlin Heidelberg.

21 Jean Bourgain, Moubariz Z. Garaev, Sergei Konyagin, and Igor E. Shparlinski. On the Hidden Shifted Power Problem. *SIAM J. Comput.*, 41(6):1524–1557, 2012.

22 Marco L. Carmosino, Russell Impagliazzo, Valentine Kabanets, and Antonina Kolokolova. Learning Algorithms from Natural Proofs. In *Conference on Computational Complexity*, volume 50 of *LIPIcs*, pages 10:1–10:24. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016.

23 Marco L. Carmosino, Russell Impagliazzo, Valentine Kabanets, and Antonina Kolokolova. Agnostic Learning from Tolerant Natural Proofs. In *APPROX-RANDOM*, volume 81 of *LIPIcs*, pages 35:1–35:19. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017.

24 Ivan Damgård. On the Randomness of Legendre and Jacobi Sequences. In *CRYPTO*, volume 403 of *Lecture Notes in Computer Science*, pages 163–172. Springer, 1988.

25 Deepesh Data, Manoj Prabhakaran, and Vinod M. Prabhakaran. On the Communication Complexity of Secure Computation. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part II*, volume 8617 of *Lecture Notes in Computer Science*, pages 199–216. Springer, 2014. doi:10.1007/978-3-662-44381-1_12.

26 Cunsheng Ding. Pattern Distributions of Legendre Sequences. *IEEE Trans. Information Theory*, 44(4):1693–1698, 1998.

27 Itai Dinur, Daniel Kales, Angela Promitzer, Sebastian Ramacher, and Christian Rechberger. Linear Equivalence of Block Ciphers with Partial Non-Linear Layers: Application to LowMC. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part I*, volume 11476 of *Lecture Notes in Computer Science*, pages 343–372. Springer, 2019. doi:10.1007/978-3-030-17653-2_12.

28 Christoph Dobraunig, Maria Eichlseder, Lorenzo Grassi, Virginie Lallemand, Gregor Leander, Eik List, Florian Mendel, and Christian Rechberger. Rasta: A Cipher with Low ANDdepth and Few ANDs per Bit. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part I*, volume 10991 of *Lecture Notes in Computer Science*, pages 662–692. Springer, 2018. doi:10.1007/978-3-319-96884-1_22.

29 Nico Döttling and Sanjam Garg. Identity-Based Encryption from the Diffie-Hellman Assumption. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part I*, volume 10401 of *Lecture Notes in Computer Science*, pages 537–569. Springer, 2017. doi:10.1007/978-3-319-63688-7_18.

30 Uriel Feige, Joe Kilian, and Moni Naor. A minimal model for secure computation (extended abstract). In *STOC*, pages 554–563. ACM, 1994.

31 Dankrad Feist. Legendre PRF bounties. URL: https://legendreprf.org/bounties.

32 Michael J. Freedman, Yuval Ishai, Benny Pinkas, and Omer Reingold. Keyword Search and Oblivious Pseudorandom Functions. In *Theory of Cryptography, Second Theory of Cryptography Conference, TCC 2005, Cambridge, MA, USA, February 10-12, 2005, Proceedings*, pages 303–324, 2005.

33 Rosario Gennaro, Craig Gentry, and Bryan Parno. Non-interactive Verifiable Computing: Outsourcing Computation to Untrusted Workers. In Tal Rabin, editor, *Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings*, volume 6223 of *Lecture Notes in Computer Science*, pages 465–482. Springer, 2010. doi:10.1007/978-3-642-14623-7_25.

34 Oded Goldreich. *The Foundations of Cryptography - Volume 1: Basic Techniques*. Cambridge University Press, 2001. doi:10.1017/CBO9780511546891.

35 Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986. doi:10.1145/6490.6503.

36 Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. One-Time Programs. In David A. Wagner, editor, *Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings*, volume 5157 of *Lecture Notes in Computer Science*, pages 39–56. Springer, 2008. doi:10.1007/978-3-540-85174-5_3.

37 Lorenzo Grassi, Christian Rechberger, Dragos Rotaru, Peter Scholl, and Nigel P. Smart. MPC-friendly symmetric key primitives. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pages 430–443, 2016.

38 Jeffrey Hoffstein and Daniel Lieman. The Distribution of the Quadratic Symbol in Function Fields and a Faster Mathematical Stream Cipher. In Kwok-Yan Lam, Igor Shparlinski, Huaxiong Wang, and Chaoping Xing, editors, *Cryptography and Computational Number Theory*, pages 59–68, Basel, 2001. Birkhäuser Basel.

39 Yuval Ishai. Randomization Techniques for Secure Computation. In *Secure Multi-Party Computation*, volume 10 of *Cryptology and Information Security Series*, pages 222–248. IOS Press, 2013.

40 Yuval Ishai and Eyal Kushilevitz. Randomizing Polynomials: A New Representation with Applications to Round-Efficient Secure Computation. In *41st Annual Symposium on Foundations of Computer Science, FOCS 2000, 12-14 November 2000, Redondo Beach, California, USA*, pages 294–304. IEEE Computer Society, 2000. doi:10.1109/SFCS.2000.892118.

41 Gábor Ivanyos, Marek Karpinski, Miklos Santha, Nitin Saxena, and Igor E. Shparlinski. Polynomial Interpolation and Identity Testing from High Powers Over Finite Fields. *Algorithmica*, 80(2):560–575, 2018.

42 Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography, Second Edition*. CRC Press, 2014.

43 Dmitry Khovratovich. Key recovery attacks on the Legendre PRFs within the birthday bound. *IACR Cryptology ePrint Archive*, 2019:862, 2019.

44 Joe Kilian. Founding Cryptography on Oblivious Transfer. In Janos Simon, editor, *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, pages 20–31. ACM, 1988. doi:10.1145/62212.62215.

45 O B Lupanov. A Method for Synthesizing Circuits. *Radiofizika*, 1958.

46 Christian Mauduit. Finite and infinite pseudorandom binary words. *Theor. Comput. Sci.*, 273(1-2):249–261, 2002.

47 Christian Mauduit and András Sárközy. On Finite Pseudorandom Binary Sequences, VI,(On Sequences). *Monatshefte für Mathematik*, 130(4):281–298, September 2000. doi:10.1007/s006050070028.

48 Moni Naor and Omer Reingold. Number-theoretic Constructions of Efficient Pseudo-random Functions. In *38th Annual Symposium on Foundations of Computer Science, FOCS '97, Miami Beach, Florida, USA, October 19-22, 1997*, pages 458–467, 1997.

49 Eduard Ivanovich Nečiporuk. On a Boolean function. In *Dokl. Akad. Nauk SSSR*, volume 169, pages 765–766, 1966.

50 Ryan O'Donnell. *Analysis of Boolean Functions*. Cambridge University Press, New York, NY, USA, 2014.

51 Igor Carboni Oliveira and Rahul Santhanam. Conspiracies Between Learning Algorithms, Circuit Lower Bounds, and Pseudorandomness. In *Computational Complexity Conference*, volume 79 of *LIPIcs*, pages 18:1–18:49. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017.

52 Rene Peralta. On the distribution of quadratic residues and nonresidues modulo a prime number. *Mathematics of Computation*, 58(197):433–440, 1992.

53 Alexander A. Razborov and Steven Rudich. Natural Proofs. *J. Comput. Syst. Sci.*, 55(1):24–35, 1997.

54 Christian Rechberger, Hadi Soleimany, and Tyge Tiessen. Cryptanalysis of Low-Data Instances of Full LowMCv2. *IACR Trans. Symmetric Cryptol.*, 2018(3):163–181, 2018. doi:10.13154/tosc.v2018.i3.163-181.

55 Joël Rivat and András Sárközy. On Pseudorandom Sequences and Their Application. In Rudolf Ahlswede, Lars Bäumer, Ning Cai, Harout K. Aydinian, Vladimir M. Blinovsky, Christian Deppe, and Haik Mashurian, editors, *General Theory of Information Transfer and Combinatorics*, volume 4123 of *Lecture Notes in Computer Science*, pages 343–361. Springer, 2006. doi:10.1007/11889342_19.

56 Alexander Russell and Igor E. Shparlinski. Classical and quantum function reconstruction via character evaluation. *J. Complexity*, 20(2-3):404–422, 2004. doi:10.1016/j.jco.2003.08.019.

57 John E. Savage. *Models of computation - exploring the power of computing*. Addison-Wesley, 1998.

58 Rocco A. Servedio and Li-Yang Tan. What Circuit Classes Can Be Learned with Non-Trivial Savings? In *ITCS*, volume 67 of *LIPIcs*, pages 30:1–30:21. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017.

59 C. E. Shannon. The synthesis of two-terminal switching circuits. *The Bell System Technical Journal*, 28(1):59–98, January 1949. doi:10.1002/j.1538-7305.1949.tb03624.x.

60 Igor E. Shparlinski. Private Communication, 2019.

61 Wim van Dam. Quantum Algorithms for Weighing Matrices and Quadratic Residues. *Algorithmica*, 34(4):413–428, 2002.

62 Wim van Dam and Sean Hallgren. Efficient Quantum Algorithms for Shifted Quadratic Character Problems. *CoRR*, quant-ph/0011067, 2000. arXiv:quant-ph/0011067.

63 Wim van Dam, Sean Hallgren, and Lawrence Ip. Quantum Algorithms for Some Hidden Shift Problems. *SIAM J. Comput.*, 36(3):763–778, 2006.

64 Frederik Vercauteren. The Hidden Root Problem. In *Pairing*, volume 5209 of *Lecture Notes in Computer Science*, pages 89–99. Springer, 2008.

65 Andrew Chi-Chih Yao. How to Generate and Exchange Secrets (Extended Abstract). In *27th Annual Symposium on Foundations of Computer Science, Toronto, Canada, 27-29 October 1986*, pages 162–167. IEEE Computer Society, 1986. doi:10.1109/SFCS.1986.25.

A PRF Bounds in the Uniform Setting

In this appendix, we give improved lower bounds on the complexity of garbling pseudorandom functions (PRFs). In particular, the attack presented here is uniform, as opposed to non-uniform bounds in Corollary 15. Our results follow from applying the natural proof framework of [53]. However, we achieve improved bounds by demonstrating the existence of a property tester for a relaxation of Nečiporuk's measure. By combining our results with those of Section 3.4 we show any exponentially-secure PRF has DRE size $\Omega(n^2/\log n)$.

We then discuss a candidate PRF with a DRE construction of size almost matching the lower bound.

A.1 PRFs are complex under (average-case) Nečiporuk

Intuitively, because a random function has high measure under Nečiporuk, so should a pseudorandom function.[10] In fact, Servedio and Tan have recently shown how to exactly learn functions with low ($O(n^{1.99})$) measure under Nečiporuk in time $2^{n-n^{\delta}}$ (via membership and equivalence queries) [58]. We show that the much simpler task of simply distinguishing a function with low measure can be done much more quickly (and without equivalence queries, which do not fit into the usual PRF game).

We accomplish this via an average case variant of Nečiporuk. Recall that Nečiporuk is ultimately statement about the number of functions that can be generated under some restriction. Viewed differently, this can be framed as a statement about the *maximum entropy* of the random variable defined by sampling a restricted function uniformly at random. Our observation is that for the special case of distinguishing from a random function it suffices to look at the *Shannon entropy* of the same variable. Consequently, instead of bounding the support size we can focus on much easier task of bounding the entropy.

An "average-case" notion of Nečiporuk

We begin by introducing our average-case variant of Nečiporuk's measure that relies on Shannon entropy as opposed to maximum entropy.

For a function $f : \{0,1\}^n \to \{0,1\}$, and a set $S \subset [n]$, let $Z^{f,S}$ denote the variable distributed according to $f_{S|z}$ for uniformly drawn $z \leftarrow \{0,1\}^{\bar{S}}$. Define,

$$h_S(f) \stackrel{\mathsf{def}}{=} H(Z^{f,S}).$$

Notice that $H_{\max}(Z^{f,S}) = g_S(f)$, thus it follows that $h_S(f) \leq g_S(f)$.

Random functions are complex (under h_S)

Next we observe that random functions have high complexity with respect to the average-case variant of Nečiporuk we defined above.

▶ **Proposition 26.** *For any set $S \subseteq [n]$ and a uniformly random function $F : \{0,1\}^n \to \{0,1\}$,*

$$\Pr[h_S(F) \leq 2^{|S|} - t] < \exp(-\frac{2t^2}{|\bar{S}| + \ln(2)})$$

We can apply the same style of balls/bins argument used for Nečiporuk's original measure again here.

[10] Statements of this form indeed were at the heart of Razoborov and Rudich's natural proof framework and its recent extensions [53, 22, 23, 51]. Unfortunately, because Nečiporuk's measure seems to behave poorly under known pseudorandom function generators, its not clear how to apply their framework here to get strong bounds on pseudorandom generators with simple DREs.

Proof. First, we bound $\mathbb{E}[H(Z^{F,S})]$ from below. We will omit S from the superscript in this proof ($Z^F = Z^{F,S}$). Additionally, we will take Z^F_ϕ to denote $\Pr_F[Z^F = \phi]$. Note that for any ϕ, $\mathbb{E}[Z^F_\phi] = 1/\#\{\phi : \{0,1\}^{|S|} \to \{0,1\}\} = 2^{-2^{|S|}}$.[11]

$$\begin{aligned}
\mathbb{E}_F[H(Z)] &= \mathbb{E}_F\left[\sum_\phi Z^F_\phi \log(1/Z^F_\phi)\right] \\
&= \sum_\phi \mathbb{E}_F[Z^F_\phi \log(1/Z^F_\phi)] \\
&\geq \sum_\phi \mathbb{E}_F[Z^F_\phi] \log(1/\mathbb{E}_F[Z^F_\phi]) \\
&= 2^{2^{|S|}} \cdot \frac{1}{2^{2^{|S|}}} \log(2^{2^{|S|}}) \\
&= 2^{|S|}
\end{aligned}$$

Note that the third line follows from Jensen's inequality.

Next, we show concentration around the mean in the standard way. Consider $H(Z^F)$ as a Doob martingale on the independent random variables $F_{S|z}$ for $z \in \{0,1\}^{\bar{S}}$. Clearly, if F and F' only differ on single restriction of f to z, then $|H(Z^F) - H(Z^{F'})| \leq \frac{\log(2^{|\bar{S}|})+\ln(2)}{2^{|\bar{S}|}}$. Moreover, because F is random, these variables are independent. So, we can apply McDiarmid/Azuma's inequality to get, for any $t > 0$:

$$\Pr_F[\mathbb{E}[H(Z^F)] - H(Z^F) \geq t] \leq \exp(\tfrac{2t^2}{|\bar{S}|+\ln(2)}).$$ ◀

Plugging $|S| = \log n$ and $t = n/2$ into the above proposition we immediately get the following corollary.

▶ **Corollary 27.** *For any set $S \subseteq [n]$ such that $|S| = \log n$, if $F : \{0,1\}^n \to \{0,1\}$ is a uniformly random function, then*

$$\Pr[h_S(F) \leq n/2] < \exp(-\frac{n^2}{2(n - \log n + \ln(2))}) < \exp(-n/2).$$

A.2 Low Nečiporuk measure can be distinguished from random

Next, we use the above to show that any function with Nečiporuk measure that is slightly less than maximal can be distinguished from a random function in time $O(2^{n/10})$. It immediately follows that none of the classes whose functions have bounded Nečiporuk measure can contain exponentially-secure PRFs.

The following theorem is implicit in Batu et al. [15].

▶ **Imported Theorem 28.** *There is an algorithm that given sample access to a distribution X supported on $[N]$, promised to either have "high" entropy (at least $N/2$) or "low" entropy (at most $N/21$), runs in time $\tilde{O}(N^{1/100})$ and distinguishes which is the case with overwhelming probability.*

[11] In more detail: Let $M = \#\{0,1\}^{\bar{S}}$ (number of balls) and $N = \#\{0,1\}^{\{0,1\}^S}$ (number of bins). Then, for $k \in \mathbb{N}$ we can see that $\Pr[Z^F_\phi = k/M]$ is the probability that exactly k out M balls (or restrictions $z \in \{0,1\}^{\bar{S}}$) hit the bin ϕ (which happens with probability $1/N$). Thus, $\Pr[Z^F_\phi = k/M] = \binom{M}{k} N^{-k}(1 - \frac{1}{N})^{M-k}$. Because this is simply a rescaled binomial distribution it follows that $\mathbb{E}[Z^F_\phi] = \frac{1}{M} \cdot \frac{M}{N} = \frac{1}{N}$.

▶ Remark 29. Batu et al. actually show how to multiplicatively approximate entropy within a factor of $(1+2\epsilon)\gamma$ ($\gamma > 1, \epsilon \in (0, 1/2]$) given sample access in time $O(N^{1/\gamma^2}/\epsilon^2 \log n)$ with constant failure probability when the distribution has entropy at least $\Omega(\gamma/\eta)$ for some small constant η ([15, Theorem 2]).

To apply this to the low entropy case, it suffices to suffices to show min-entropy is greater than the constant assumed above. For these parameters, empirical estimates are more than efficient enough. In fact, [15, Lemma 2] says just that. Finally, correctness of these estimates can be amplified by taking the median/majority after $\operatorname{poly}\log n$ repetitions.

▶ **Theorem 30.** *There is an algorithm running in time $\tilde{O}(2^{n/100})$ that given oracle access to either a random function $F : \{0,1\}^n \to \{0,1\}$ or any $f : \{0,1\}^n \to \{0,1\}$ such that $G(f) < \frac{n^2}{21 \log n}$ can distinguish between the two cases with overwhelming advantage.*

Proof. Note that if $G(f) < \frac{n^2}{21 \log n}$, then in particular $\sum_{V_i} g_{V_i}(f) < \frac{n^2}{21 \log n}$ for the partition $(V_1, \ldots, V_{n/\log n})$ of $[n]$ into consecutive $\log n$-bit blocks. Moreover, there must be some V_i such that $h_{V_i}(f) \le g_{V_i}(f) < n/21$.

In contrast, Corollary 27 implies that for a uniformly random F, it holds with overwhelming probability that for all i, $h_{V_i}(F) \ge n/2$.

Additionally, for any i, it is possible to efficiently sample Z^{f,V_i} by simply drawing $z \leftarrow \{0,1\}^{|\bar{V}_i|}$ uniformly at random and evaluating $f_{V_i|z}$ on all $x \in \{0,1\}^{V_i}$. Because $|V_i| = \log n$, this procedure takes time $\operatorname{poly}(n)$.

It follows that we can run the procedure from Imported Theorem 28 on all V_i in time $\tilde{O}(2^{n/100})$. If the procedure outputs "High" on all V_i, then output "F." Otherwise, output "f." By Theorem 28 and the above observations, the procedure described will err with at most negligible probability. ◀

▶ Remark 31. We note that for $\epsilon > 0$ the above distinguisher can be modified to test on the partition $V = (V_1, \ldots, V_m)$ where each V_i is a block of size $\epsilon \log n$ ($m = \frac{n}{\epsilon n}$) and again distinguish entropy that differs by constant factor in any block from $n^\epsilon/2$, taking time $O(2^{n^\epsilon})$ overall. By Proposition 26 a random function will have Nečiporuk measure $h_{V_i}(f) \ge n^\epsilon/2$ for all V_i with high probability. It follows that an $O(2^{n^\epsilon})$-secure PRF must have DRE complexity $\Omega(n^{1+\epsilon}/\log n)$.

PRFs have high complexity

From Theorem 30, it almost immediately follows that there can be no exponentially-secure PRFs in any class to which Nečiporuk applies. This yields a host of lower bounds on PRF complexity that, to our knowledge, were not known before now.

▶ **Corollary 32.** *No exponentially-secure PRF has*

- *Decomposable Randomized Encodings of size $o(n^2/\log n)$,*
- *Binary formulas of size $o(n^2/\log n)$ over arbitrary basis,*
- *Deterministic branching programs of size $o(n^2/\log^2 n)$,*
- *Switching networks of size $o(n^2/\log^2 n)$,*
- *Non-deterministic branching programs of size $o(n^{3/2}/\log n)$,*
- *Parity branching programs of size $o(n^{3/2}/\log n)$,*
- *Span programs of size $o(n^{3/2}/\log n)$,*
- *Switching-and-rectifier networks of size $o(n^{3/2}/\log n)$.*

B Deferred Proofs

▶ **Proposition 33.** *For any set* $S \subseteq [n]$ *and a random function* $f : \{0,1\}^n \to \{0,1\}$, $\Pr_f[2^{g_S(f)} \leq 2^{n-|S|} - t] < \exp(-\frac{2t^2}{2^{n-|S|}})$

This follows from a standard balls & bins argument, reproduced here for completeness.

Proof. Recall that $2^{g_V(f)} = \#\{f_{S|z} : z \in \{0,1\}^{\bar{S}}\}$. If we let Y_ϕ for $\phi : \{0,1\}^S \to \{0,1\}$ be the indicator random variable such that

$$Y_\phi := \begin{cases} 1 & \text{if } \exists z \in \{0,1\}^{\bar{S}} : f_{S|z} = \phi \\ 0 & \text{otherwise} \end{cases}$$

Then we can rewrite the above as,

$$2^{g_S(f)} = \sum_{\phi:\{0,1\}^S \to \{0,1\}} Y_\phi.$$

By linearity of expectation,

$$\mathbb{E}[2^{g_S(f)}] = \mathbb{E}[\sum_\phi Y_\phi] = \sum_\phi \mathbb{E}[Y_\phi] = 2^{2^{|S|}} \cdot \frac{2^{|\bar{S}|}}{2^{2^{|S|}}} = 2^{n-|S|}.$$

Finally, we consider $2^{g_S(f)}$ as a doob martingale on the independent random variables $f_{S|z}$ for $z \in \{0,1\}^{\bar{S}}$. Clearly, if f and f' only differ on single restriction of f to z, then $|g_S(f) - g_S(f')| \leq 1$. Moreover, because f is random, these variables are independent. So, we can apply McDiarmid/Azuma's inequality to get, for any $t > 0$:

$$\Pr_f[\mathbb{E}[2^{g_S(f)}] - 2^{g_S(f)} \geq t] \leq \exp(\tfrac{2t^2}{2^{|\bar{S}|}}).$$ ◀

In particular, if we take $|S| = \log n$ and $t = 2^{n-\log n-1}$, then $\Pr_f[g_S(f) \leq n - \log n - 1] \leq \exp(-2^{n-\log n-1})$. This yields the following corollary via a union bound.

▶ **Corollary 34.** *For a random function* f, $\Pr_f[G(f) \leq n^2/\log n - 2n] \leq \frac{n}{\log n} \cdot \exp(-2^{n-1})$.